

Programul de studii :

Informatică Aplicată

Lucrare de licență

**Platformă de Social Media**

|  |  |
| --- | --- |
| Autor: | **Ștefan Adriana** |
| Coordonator ştiințific: | **Lector Dr. Răzvan Bocu** |

Braşov, 2021

Contents

[1. Introducere 4](#_Toc75343926)

[1.1. Contextul proiectului 4](#_Toc75343927)

[1.2. Motivația 5](#_Toc75343928)

[2. Obiectivele Proiectului 6](#_Toc75343929)

[2.1. Obiectivul principal 6](#_Toc75343930)

[2.2. Obiective secundare 6](#_Toc75343931)

[3. Tehnologii folosite 8](#_Toc75343932)

[3.1. Java 11 8](#_Toc75343933)

[3.1.1. Utilitate 8](#_Toc75343934)

[3.1.2. Spring Boot 9](#_Toc75343935)

[3.1.3. Apache Maven 9](#_Toc75343936)

[3.1.4. Hibernate 10](#_Toc75343937)

[3.1.5. MySQL 11](#_Toc75343938)

[3.1.6. JWT 11](#_Toc75343939)

[3.1.7. Angular 11](#_Toc75343940)

[4. Arhitectura sistemului 12](#_Toc75343941)

[5. Arhitectura componentei backend 13](#_Toc75343942)

[5.1. Detalierea componentelor 14](#_Toc75343943)

[5.1.1. Controller 15](#_Toc75343944)

[5.1.2. Service 16](#_Toc75343947)

[5.1.3. Repository 18](#_Toc75343948)

[5.1.4. Model 20](#_Toc75343949)

[5.1.5. Configuration 22](#_Toc75343950)

[5.1.6. Security 24](#_Toc75343951)

[5.1.7. DTO 25](#_Toc75343952)

[5.1.8. Util 26](#_Toc75343953)

[6. Arhitectura componentei frontend 27](#_Toc75343954)

[6.1. Pachetul component 28](#_Toc75343955)

[6.1.1. Componenta Header 31](#_Toc75343956)

[6.1.2. Componenta Profile-Page 34](#_Toc75343957)

[6.1.3. Componenta Login 39](#_Toc75343958)

[6.1.4. Componenta Home-Pages 42](#_Toc75343959)

[6.1.5. Componenta Post-Pages 49](#_Toc75343960)

[6.1.6. Componenta Chat-Pages 50](#_Toc75343961)

[6.2. Pachetul Model 52](#_Toc75343962)

[1.1. Pachetul Service 53](#_Toc75343963)

[1.2. Pachetul Pipe 55](#_Toc75343964)

[7. Structura bazei de date 55](#_Toc75343965)

[8. Concluzii 60](#_Toc75343966)

[8.1. Obiective propuse 60](#_Toc75343967)

[8.2. Idei de dezvoltare ulterioare 60](#_Toc75343968)

[Bibliografie 62](#_Toc75343969)

# **Introducere**

## Contextul proiectului

Platformele de social media permit oamenilor să se conecteze cu cei dragi și facilitează comunicarea între aceștia. Totodată ajută la creșterea comunicării și stabilirii conexiunilor cu oamenii din întreaga lume, permit creeare și schimbul de informații și idei și formează comunitații virtuale de oameni.

Social media este interacțiunea comună între oameni în care creează, împărtășesc său schimbă informații și idei în comunitățile virtuale. A devenit nevoia de bază și calitatea ființelor umane de a fi social. Evoluțiile spectaculoase în comunicații și tipurile de divertisment inovatoare și uimitoare au dat acces la informații și capacitatea de a da voce oamenilor care nu ar fi fost auziți niciodată. Actuala generație este destul de norocoasă să asiste la unele dintre cele mai uimitoare evoluții tehnologice din istorie.

Oamenii doresc întotdeauna să se conecteze cu societatea într-un fel său altul. În zilele anterioare, modurile de comunicare erau limitate. Oamenii au socializat cu ceilalți în propriile lor moduri. Mai devreme, socializarea a fost restrânsă la vizitarea reciprocă a locului, având întâlniri mari, întâlniri în cluburi, parcuri și alte locuri publice. Acum timpul s-a schimbat. Oamenii și-au minimizat viața socială din cauza vieții agitate și a creșterii distanței geografice și a preocupărilor economice.

Odată cu venirea tehnologiei, site-urile și aplicațiile de rețele sociale au anunțat o revoluție în lume. A adus într-adevăr oameni din întreaga lume mai aproape prin crearea, schimbul său schimbul de informații și idei în comunități și rețele virtuale. Aceste site-uri de rețele sociale se bazează pe tehnologii web și creează platforme extrem de interactive. A căpătat impuls la nivel global datorită caracteristicilor săle mai bune, accesului, frecvenței, imediatității, utilizabilității și permanenței. A fost recunoscută atât de larg și utilizarea a crescut atât de incredibil astăzi încât s-a mutat de la computerele desktop la laptopuri la telefoanele mobile. Platforma de social media este, fără îndoială, ușor de obținut și accesibilă. Astăzi, fiecare persoană este dependentă de rețelele sociale și asta cu o viteză fulgerătoare.

Avantajele unei astfel de platforme enumera:

* Vizează un public larg, făcându-l un instrument util și eficient pentru societate.
* Acesta ajunge la oameni chiar și în zonele îndepărtate, iar informațiile sunt răspândite rapid.
* Distanța nu mai este o limitare din cauza rețelelor sociale. Poți fi în permanență actualizat cu cele mai recente știri și evenimente din societate și mediu prin intermediul site-urilor de socializare.
* Site-uri și bloguri precum Facebook, Twitter, Instagram, LinkedIn și multe altele au devenit un instrument pentru conectarea oamenilor de pe tot globul. Oamenii pot participa la discuții live său sesiuni live său prelegeri care se desfășoară oriunde în lume în timp ce stau acasă.
* Profesorii și profesorii pot preda pe diferite teme din locuri îndepărtate.
* Rețelele sociale permit companiilor să utilizeze aceste site-uri ca rețea pentru a genera conștientizare despre produsul lor, pentru a-și promova marca și, prin urmare, pentru a-și crește vânzările. Economisește costurile de marketing și publicitate.
* Aceste site-uri de rețea de pe rețelele de socializare oferă o platformă cuprinzătoare pentru tinerii care aspiră la artiști pentru a-și prezenta pasiunea și abilitățile.
* Liderii politici folosesc platforma rețelelor sociale pentru răspândirea comunicării sociale în masă. În aceste zile, candidații politici comunică și cu alegătorii prin intermediul rețelelor sociale.

Dar aceste platforme au de asemenea și dezavantaje. Aceste dezavantaje sunt reprezentate de:

* Utilizatorii rețelelor sociale devin victime ale unor escrocherii frauduloase și online care par a fi autentice.
* Deschide posibilitatea hackerilor de a comite fraude și de a lansă atacuri virale
* Productivitatea oamenilor este îngreunată din cauza utilizării și îngăduinței extreme pe aceste site-uri de socializare.
* Și studenții sunt extrem de activi pe site-urile de socializare în aceste zile, ceea ce îi limitează de la activități în aer liber.
* Unele site-uri sunt utilizate pentru a exprima furia său disputa personală din cauza căreia se creează mult haos și confuzie.

## Motivația

Moțivatia care a stat în spatele acestei alegeri de tema pentru proiectul de licenta a fost dorinta de a aprofunda notiunile tehnice invatate pe parcusul celor trei ani de studii ce vizeaza partea de dezvoltare web. Tematica aplicației a starnit interes deoarece notiunile abordate au reprezentat o provocare pe parcusul facultatii, notiuni precum lucrul cu o baza de date complexa, folosirea securitatii pentru asigurarea integritatii aplicației și lucrul cu o arhitectura cat mai bine definita a proiectului.

# **Obiectivele Proiectului**

## Obiectivul principal

Scopul principal al proiectului este reprezentat de dezvoltarea unei platforme social media cu ajutorul căreia se va facilita comunicarea și împarașirea de informații între utilizatori. Această platformă va fi folosită de către utilizatori, cărora la momentul înregistrării li se vor atribui un profil cu care se vor putea face identificați în momentul căutării de către ceilalți utilizatori.

## Obiective secundare

Scopurile secundare ale proiectului urmăresc oferirea posibilității de a împartași opiniile într-o manieră rapidă și facilă prin intermediul platformei:

* Exprimarea părerilor față de o postare prin apreciere și prin comentariu

Utilizatorul își poate exprima aprecierea față de postarea unui alt utilizator folosind un buton destinat special acestei acțiuni, buton sub forma de inimă, și își poate retrage aprecierea în cazul în care acesta consideră că nu este cazul său că s-a produs o greșeală. De asemenea un utilizator poate lăsă comentarii la o postare.

* Notificarea în momentul în care se produce acțiune ce îl vizează pe utilizatorul logat

Dacă utilizatorul logat va primi o apreciere, i se va comenta la o postare, va incepe să fie urmarit de un alt utilizator său primește un mesaj, atunci platforma îi va trimite automat o notificare pentru a-l înștiința. Ideea din spatele acestei funcționalități este de a putea raspunde cursiv și instant la acțiunile ce îl privesc pe acesta.

* Căutare altor utilizatori după username său numele complet

Această funcționalitate urmărește să permită utilizatorilor posibilitatea de a căuta alți utilizatori în mod dinamic, în funcție de username-ul acestora său după numele complet. Un câmp de căutare va trimite request-uri constant cu modificările aduse în câmp pentru a aduce din baza de date cele mai bune variante în urma căutării.

* Adăugarea de informații personale în profil

Utilizatorul logat are posibilitatea de a-și îmbogăți profilul personal prin adăugarea unei poze personale, a unui nume complet personalizat și a unei descrieri prin care acesta dorește să fie remarcat. De asemenea în detaliile utilizatorului vor aparea informații cu privire la numărul de personae urmărite, numărul de personae ce urmăresc utilizatorul și la numărul de postări adaugate de acesta.

* Crearea de postări

Un utilizator își poate exprima gândurile prin crearea unei postări personalizate ce va conține neaparat o poză, dar este posibilă și adaugarea unei descrieri corespunzatoare postării cât și crearea de etichete pentru utilizatorii existenți în platformă, aceste etichete au ca scop divertismentul cât și o împărtășire mai ușoară a posibilelor conexiuni cu alte profile în cazul ca acestea nu exista.

* Urmărirea utilizatorilor ce nu se afla deja în lista de urmariți ai utilizatorului logat

Utilizatorul are posibilitatea de a-și reînnoi lista de urmăriți folosindu-se de o listă special filtrată unde el poate vizualiza și adăuga la urmăriți utilizatorii pe care acesta nu îi urmărește deja.

* Vizualizarea detaliilor unei postări

Această funcționalitate dorește crearea unei ferestre speciale unde utilizatorul poate vizualiza în de-aproape o postare cu toate detaliile acesteia precum: cine a creat-o, cine este etichetat în ea, descrierea acesteia, câte aprecieri și comentarii are, cât și comentariile existente la postare.

* Vizualizarea detaliilor unui profil

Utilizatorul logat va putea să vizualizeze detaliile profilului său: poză de profil, username-ul, numele complet, descrierea, numărul de postări, numărul de personae urmărite și numărul de urmăritori cât postările create de acesta și postările create de-a lungul timpului.

* Posibilitatea de a modifica sau șterge o postare

Dacă cel care crează o postare va dori să modifice persoanele etichetate și descrierea postării atunci acesta va avea această posibilitate. Poză postării nu va putea fi modificată, postarea în sine fiind definite de poză creată. De asemena utilizatorul are posibilitatea de a-și șterge postarea.

* Posibilitatea de a șterge un comentariu

Utilizatorul logat va avea posibilitatea de a-și șterge propriile comentarii lasăte la postări cât și comentariile lăsate de ceilalți utilizatori la propriile postări.

# **Tehnologii folosite**

Pentru realizarea lucrării am optat pentru back-end în Spring Boot configurat pentru Java 11, iar pentru front-end am ales să folosesc Angular.

## Java 11

Java este un limbaj de programare obiect-orientat, cu un scop general și destinat lucrului cu clase, special creat pentru a avea cât mai puține dependințe la implementare. Acest limbaj este destinat dezvoltării de aplicații, prin urmare, este rapid, sigur și de încredere, fiind astfel utilizat la scară largă pentru dezvoltarea de diverse aplicații dintr-o gama largă de proiecte cu diferite cerințe.

Java a fost dezvoltat de Sun Microsystems în anul 1995. James Gosling este cunoscut că tatăl Java. Înainte de Java, numele său era Oak. Deoarece Oak era deja o companie înregistrată, așa că James Gosling și echipa să au schimbat numele Oak în Java.

### Utilitate

Unul dintre cele mai mari motive pentru care Java este atât de popular este independența platformei. Programele pot rula pe mai multe tipuri diferite de calculatoare; atâta timp cât computerul are instalat un mediu Java Runtime Environment (JRE), poate rula un program Java. Prin urmare este foarte ușor de creat un mediu de dezvoltare în care s epoate utiliza acest limbaj de programare.Java este utilizat în diferite domenii precum:

i) Crearea de aplicațîi Android

Android este o platformă software open source ce permite dezvoltatorilor să scrie cod gestionat folosind Java pentru a gestiona și controla dispozitivul Android. Aplicațiile Android pot fi dezvoltate utilizând limbajul de programare Java și Android SDK. Deci, familiarizarea cu elementele de bază ale limbajului de programare Java este o condiție prealabilă pentru programarea pe platforma Android. Deși există și alte modalități de a crea aplicațîi Android, majoritatea aplicațiilor sunt scrise în Java folosind API-ul Android Google.

ii) Createa de aplicații Web

Java este unul dintre cele mai utilizate limbaje de programare pentru dezvoltarea aplicațiilor web dinamice. O aplicație web este un software de calculator care utilizează browserul web și tehnologiile pentru a efectua sărcini pe internet. O aplicație web este implementată pe un server web.

Java oferă câteva tehnologii precum Servlet și JSP care ne permit să dezvoltăm și să implementăm cu ușurință o aplicație web pe un server. De asemenea, oferă câteva cadre precum Spring, Spring Boot care simplifică activitatea și oferă un mod eficient de a dezvoltă o aplicație web. Acestea reduc efortul dezvoltatorului.

Multe departamente guvernamentale, de asistență medicală, asigurări, educație și apărare au aplicațiile web construite în Java. Un exemplu semnificativ în acest sens este Gmail Google.

### Spring Boot

Spring Boot este un framework pentru crearea de aplicații independente și este o extensie a framework-ului Spring. Configurarea este mai ușoară, reducând timpul de dezvoltare și crescând astfel productivitatea.

Funcțiile principale sunt configurarea automată a Spring, existența fișierului pom.xml pentru configurația Maven, integrarea cu Tomcat, Jetty, înlocuirea fișierelor XML cu comentarii etc.

Avantajele utilizării Spring Boot enumeră dezvoltarea aplicațiilor ușor de înțeles și de dezvoltat, modularitatea codului și reutilizara acestuia, productivitatea sporită și reducerea timpului efectiv de dezvoltare.

Motivarea alegerii acestei tehnologii a constat în faptul că această permite configurarea ușoară a proiectului și a dependintelor necesăre, conectarea ușoară la baza de date, existența anotarilor ce permit manipularea ușoară a codului.

### Apache Maven

Maven este un sistem de build și administrare a proiectelor, scris în Java. Face parte din proiectele găzduite de Apache Software Foundation. Funcționalitățile săle principale sunt descrierea procesului de build al software-ului și descrierea dependențelor acestuia. Proiectele sunt descrise printr-unul său mai multe fișier XML, denumite POMuri (Project Object Model), dar au o structură implicită, ceea ce încurajează structurarea similară a proiectelor.

## Hibernate

Hibernate este un framework Java care simplifică dezvoltarea aplicației Java pentru a interacționa cu baza de date. Este un instrument open source, ușor, ORM (Object Relational Mapping). Hibernate implementează specificațiile JPA (Java Persistence API) pentru persistența datelor.

Următoarele sunt avantajele cadrului de hibernare:

* Open Source și ușor

Cadrul Hibernate este open source sub licență LGPL și ușor.

* Performanță rapidă

Performanța cadrului de hibernare este rapidă, deoarece memoria cache este utilizată intern în cadrul de hibernare. Există două tipuri de cache în cache-ul de primul nivel și în cel de-al doilea nivel al cadrului de hibernare. Memoria cache de primul nivel este activată în mod implicit.

* Interogare independentă a bazei de date

HQL (Hibernate Query Language) este versiunea orientată obiect a SQL. Generează interogări independente din baza de date. Deci nu este nevoie să scrieți interogări specifice bazei de date. Înainte de Hibernate, dacă baza de date este modificată pentru proiect, trebuie să schimbăm și interogarea SQL care duce la problema de întreținere.

* Crearea automată a tabelelor

Cadrul Hibernate oferă facilitatea de a crea automat tabelele bazei de date. Deci, nu este nevoie să creați manual tabele în baza de date.

* Simplifică Complex Join

Preluarea datelor din mai multe tabele este ușoară în cadrul hibernării.

* Oferă statisticile interogărilor și starea bazei de date

Hibernate acceptă Query cache și oferă statistici despre interogare și starea bazei de date.

Arhitectura Hibernate include multe obiecte, cum ar fi obiectul persistent, fabrica de sesiuni, fabrica de tranzacții, fabrica de conexiuni, sesiunea, tranzacția etc.

Aplicația de hibernare poate fi creată cu adnotare. Există multe adnotări care pot fi folosite pentru a crea aplicații de hibernare, cum ar fi @Entity, @Id, @Table etc.

### MySQL

Este un sistem de gestionare a bazelor de date relaționale, open-source, ideal atât pentru aplicații mici, cât și pentru aplicații mari, foarte rapid, fiabil, scalabil și ușor de utilizat, multiplatform și în conform cu standardul ANSI SQL

### JWT

Este un standard JSON pentru securitatea aplicațiilor, în special pentru autentificarea utilizatorilor în aplicații. Standardul definește un model compact și autonom pentru transmiterea sigură a informațiilor prin obiecte JSON. De fiecare dată când este apelat punctul final, acesta verifică dacă există un simbol valid și, dacă este invalid, utilizatorul nu va putea obține autorizație în aplicație. JWT poate fi semnat cu o cheie pentru verificarea și utilizarea jetoanelor.

Datorită dimensiunii săle relativ mici, un JWT poate fi trimis printr-un URL, printr-un parametru POST său în interiorul unui antet HTTP și este transmis rapid. Un JWT conține toate informațiile necesăre despre o entitate pentru a evita interogarea unei baze de date de mai multe ori. De asemenea, destinatarul unui JWT nu trebuie să apeleze un server pentru a valida simbolul.

### Angular

Angular este o platformă și un cadru pentru crearea de aplicații client cu o singură pagină folosind HTML și TypeScript. Angular este scris în TypeScript. Implementează funcționalități de bază și opționale ca un set de biblioteci TypeScript pe care le importați în aplicațiile dvs.

# **Arhitectura sistemului**

Arhitectura client / server este un model de calcul în care mai multe componente funcționează în roluri strict definite pentru a comunica. Serverul găzduiește, livrează și gestionează majoritatea resurselor și serviciilor care trebuie consumate de client. Acest tip de arhitectură de resurse partajate are unul său mai multe computere client conectate la un server central printr-o rețea său conexiune la internet.

Arhitectura client / server funcționează atunci când computerul client trimite o cerere de resursă său proces către server prin conexiunea la rețea, care este apoi procesătă și livrată clientului. Un computer server poate gestiona mai mulți clienți simultan, în timp ce un client poate fi conectat la mai multe servere simultan, fiecare furnizând un set diferit de servicii.

Cum funcționează modelul client-server?

Client: Când vorbim cuvântul Client, înseamnă să vorbim despre o persoană său o organizație care utilizează un anumit serviciu. În mod similar, în lumea digitală, un client este un computer (gazdă), adică capabil să primească informații său să utilizeze un anumit serviciu de la furnizorii de servicii (servere).

Servere: în mod similar, atunci când vorbim cuvântul Servere, înseamnă o persoană său un mediu care servește ceva. În mod similar, în această lume digitală, un server este un computer la distanță care oferă informații (date) său acces la anumite servicii.



Figura 1 Model al arhitecturii de tip Client-Server

# **Arhitectura componentei backend**

Diagram

Description automatically generated În vederea devzoltarii componentei backend a aplicației am optat pentru alegerea tehnologiei Spring Boot, codul fiind scris în limbajul de programare Java 11. Dezvoltarea acestei componente a urmat săblonul pus la dispoziție de arhitectură pe trei nivele cunoscută și sub denumirea de Three Layers. Această arhitectură presupune separarea în trei nivele generale a aplicației , după cum este ilustrat și în figura 2:

Figura 2 Model al arhitecturii de tip Three Layers

* Nivelul UI sau de prezentare se ocupă de gestionarea request-urilor făcute de utilizator pe interfață și trimiterea unui răspuns browser-ului. Gestionarea acestor requesturi se face de regulă în clasele de tip Controller, unde sunt definite funcii speciale ce reprezintă endpoint-uri cărora li se vor trimite informații de pe frontend. Acest nivel este specializat pe găsirea celei mai bune funcții unde se va trece la următorul nivel.
* Nivelul de Business Logic se ocupă de partea de logică aplicată request-ului creat de utilizator la nivelul UI și aplicarea de diverse operații asupra acestora. Aceste modificări pot fi reprezentate de algoritmi de sortare, de filtrare, de crearea de obiecte DTO (Dată Transfer Object) și se ocupă de crearea legăturii cu următorul nivel. Pentru acest nivel, clasele de tip Service sunt cele mai reprezentative.
* Nivelul de Data Access, cunoscut și ca Persistence Layer, execută operații CRUD asupra obiectelor anterior gestionate în nivelul de Business Logic. Clasele de tip Repository sunt responsăbile pentru de aducere a datelor, modificarea, ștergerea și inserarea acestora.Aceste clase ușurează cu mult lucrurl cu bazele de date, ele conțînând funcții ce reprezintă query-uri de tip SQL ce se pot personaliza. în acest proiect am optat pentru interfetele de tip CrudRepository, pentru executarea acțiunilor asupra bazei de date, cât și pentru PagingAndSortingRepository pentru aducerea unui număr limitat de inserții din baza de date.

Motivele principale pentru alegerea acestui model de arhitectură sunt scalabilitatea îmbunătățită a aplicației , astfel se pot dezvoltă independent unele de altele cele trei nivele și posibilitatea de a dezvoltă rapid aplicația.

## Detalierea componentelor

**Graphical user interface, text, application, chat or text message

Description automatically generated**

Figura 3 Arhitectura proiectului

Dezvoltarea independența a acestor componentă este posibilă datorită responsăbilitatilor diferite a fiecăreia dintre acesteia, responsăbilitatile fiind foarte specifice fiecărui tip de nivel, prin urmare:

### Controller

### Clasele de tip Controller se ocupă cu gestionarea fiecărui request primit prin intermediul url-urilor. Aceste clase sunt adnotate cu@RestController, adnotare ce permite crearea de end-point-uri care vor transmite un răspuns automat în momentul în care o funcție va da return, nemaifiind nevoie de adnotarea @ResponseBody. Aceste răspunsuri sunt în mod automat serializate sub formă de JSON în HttpResponse.

### Un exemplu pentru un end-point creat într-o clasă de tip Controller se află în următoarea figura 4:

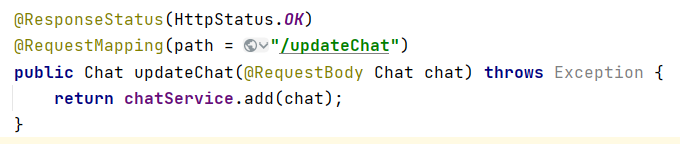
****

Figura 4

* @ResponseStatus este responsabilă de trimiterea unui HttpResponse în momentul în care se va executa cu succes modificarea în baza de date a unui obiect existent de tip Chat.
* @RequestMapping are scopul de a crea un endpoint disponibil pentru noi request-uri.
* În parametrii funcției se va cere explicit un @RequestBody de tip Chat, astfel funcția așteaptă ca numai obiecte de tip Chat vor fi trimise pentru a fi gestionate.
* În înteriorul funcției se face direct return cu răspunsul oferit de Service-ul instanțiat cu denumirea de chatService, în acest caz se va face update la un obiect deja existent din baza de date.

Pentru utilizarea serviciilor direct injectate în interiorul unui Controller acesta va fi decorat cu adnotarea @AllArgsConstructor ce permite utilizarea acestuia fără a crea un constructor în care să se initializeze toate serviciile necesăre. Adnotarea @CorssOrigin marchează metodă său tipul adnotat că permițând requesturi de tip Cross Origin.

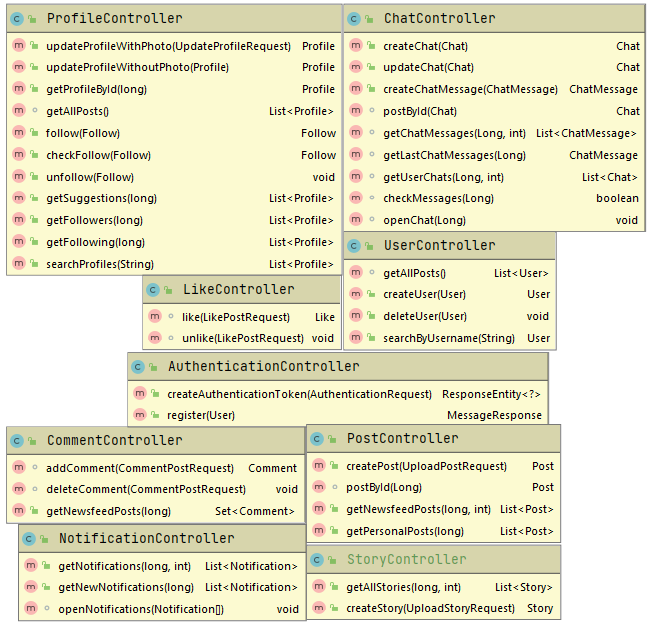
 În pachetul com.licență.socialmedia.controller am inclus clasele: ProfileController, ChatController, LikeController, UserController, AuthenticationController, PostController, CommentController, NotificationController și StoryController, clase ilustrate și în figura 5:

Figura 5

### Service

Clasele de tip Service reprezintă nivelul Business Bogic al aplicației , unde sunt aplicate diverse operații asupra obiectelor primite în Controllere și transmise la acest nivel. Acesta definește funcționalitățile pe furnizate în aplicație, modul în care acestea sunt accesăte și oferă modularitatea codului.

Aceste clase implementeza funcții defintite în interfețe ce sunt special create pentru această operație și sunt decorate cu adnotarea @Service pentru a putea fi recunoscute de aplicație că servicii. Următorul exemplu reprezintă o funcție implementată din interfață IPostService în serviciul PostService:



Figura 6

În cadrul funcției din figura 6 putem observa:

* Parametrii asupra cărora se vor aplica regulile de bussines, parametrii ce reprezintă un obiect de tip postare și o listă de obiecte de tip Profile.
* In cadrul funcției se verifica dacă exista o postare cu acelasi ID în baza de date, dacă identificatorul unic al acestei postări este reprezentat de numărul 0, atunci se va sălva obiectul exact cum a fost trimis în urma request-ului. După executarea blocului de cod conditional se va seta atributul photo cu o versiune de biti carora li se aplicat o decompresie. Dacă o astfel de postare nu exista, atunci tuturor utilizatorilor din lista de profile li se va trimite o notificare cu noua postare creată. în final se va returna obiectul de tip post, asupra carora s-au aplicat toate operatiile prezentate anterior.
* In cadrul serviciului PostService s-a injectat un câmp de tip IpostRepository, folosindu-se adnotarea @AllArgsConstructor pentru a simplifica implementarea.

În plus față de logica aplicației , aceste clase se ocupă și de entități mapate din baza de date prin intermediul obiectelor de transfer de date (DTO). De exemplu, dacă o metodă GET specifică obține o listă de entități din baza de date, această clasă va transforma răspunsul și va trimite lista obiectelor de transmisie la stratul de prezentare. Lista va fi recuperată de componenta front-end și va apărea în interfața cu utilizatorul.

În pachetul com.licenta.socialmedia.service am inclus clasele: NotificationService, FollowService, PostService, ChatMessăgeService, ChatService, UserService, ProfileService, CommentService, StoryService, LikeService, SecurityService, MyUserDetailsService.

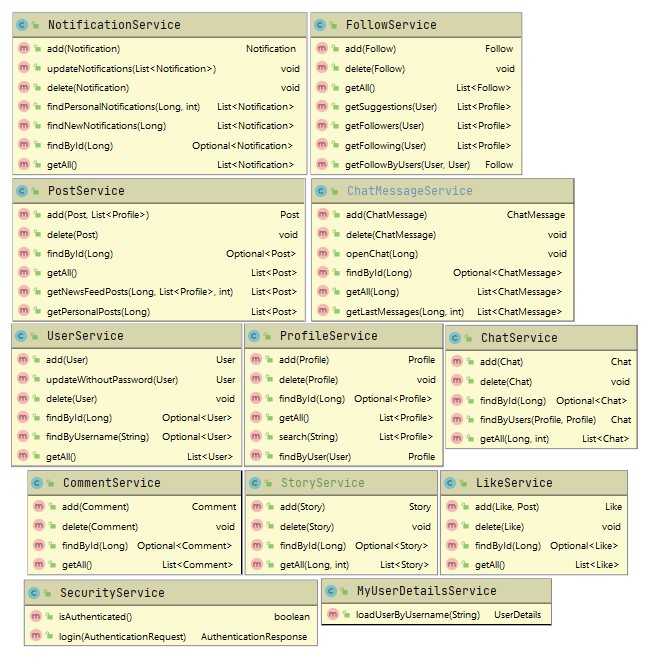


Figura 7

### Repository

Acest tip de clase reprezintă nivelul de Dată Access, cunoscut și că Persistance Layer. Aceste clase sunt responsăbile pentru comunicare în mod direct cu baza de date și crearea de query-uri automate său personalizate. Toate operațiile aplicabile bazei de date se efectuează folosind funcțiile puse la dispoziție de claselele Repository. Pentru a crea un astfel de tip de clasă este necesăra implementare funcțiilor din clasele specializate pentru gestionarea informației păstrate în baza de date.

În proiectul de față am ales să moștenesc clasele CrudRepository și PagingAndSortingRepository. Următorul exemplu ilustrează o interfață creată:

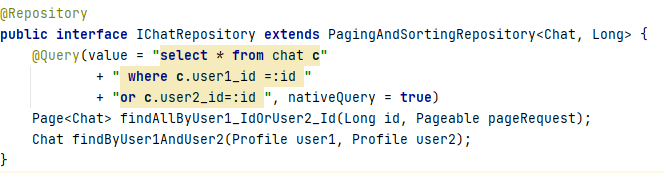


Figura 8

* @Repository are ca scop înștiințarea aplicației cu privirea la repository-ul declarat
* Fragmentul „extends PagingAndSortingRepository<Chat,Long> specifică faptul ca acest repository va crea funcții ce vor returna liste de obiecte de un anumit număr și pe o pagină specifică.
* Adnotarea @Query permite crearea unui query personalizat de sintaxă SQL, astfel se vor selecta toate obiectele de tip chat din baza de date unde oricare dintre utilizatorii ce sunt inregistrati într-o conversație au id-ul specificat în parametrii funcției.
* Tipul returnat de funcție este reprezentat de pagina de obiecte Chat, pagină cu atribute specificate de parametrul Pageable pageRequest.
* Denumirea funcției corespune acțiunii query-ului personalizat, astfel numai din nume se poate deduce rezultatul returnat.
* Parametrii reprezintă id-ul după care se face căutarea, și un obiect de tip Pagable care specifică numărul paginii ce va fi rezultată și numărul de obiecte de afișat. Astfel dacă pageRequest = PageRequest.of( 0 , 5) atunci se vor afișa primele 5 elemente, iar dacă pageRequest = PageRequest.of( 1 , 5) atunci se vor afișa urmatoarele 5 elemente după ce se va sări peste primele 5

În pachetul com.licenta.socialmedia.repository am inclus clasele: IFollowRepository, IChatRepository, IPostRepository, IStoryRepository, IChatMessăgeRepository, RoleRepository, ICommentRepository, IProfileRepository, IUserRepository, INotificationRepository, ILikeRepository.

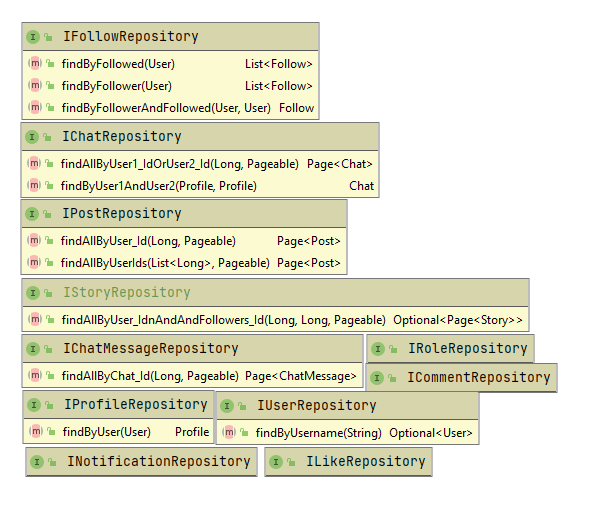


Figura 9

### Model

Aceste clase sunt create pentru generarea în baza de date a entităților. O entitate reprezintă un tabel din baza de date, iar fiecare obiect reprezintă o linie din tabela. Atributele din aceste clase reprezintă la rândul lor coloanele tabelelor.

Pentru crearea tabelei este necesăra decorarea clasei cu anotatea @Entity. Această permite maparea cu entitatea din baza de date. Această adnotare va cere în mod explicit că entitatea să conțînă un identificator unic declarat cu adnotarea @Id, căreia i se va asigura auto-incrementearea acestei chei primare folosind @GeneratedValue(strategy = GenerationType.**AUTO**). Relațiile de agregare cu alte entități se vor declara folosind adnotări ce vor define relațiile stabilite. Exemple de relații sunt @ManyToOne, @OneToOne și @OneToMany.

Un exemplu de astfel de clasă este ilustrat în figura 10:

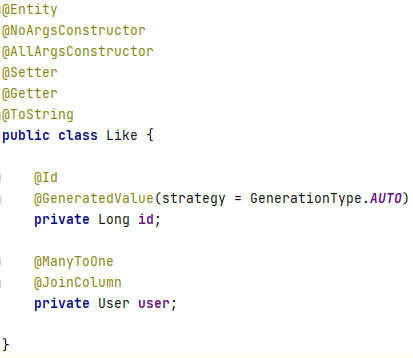


Figura 10

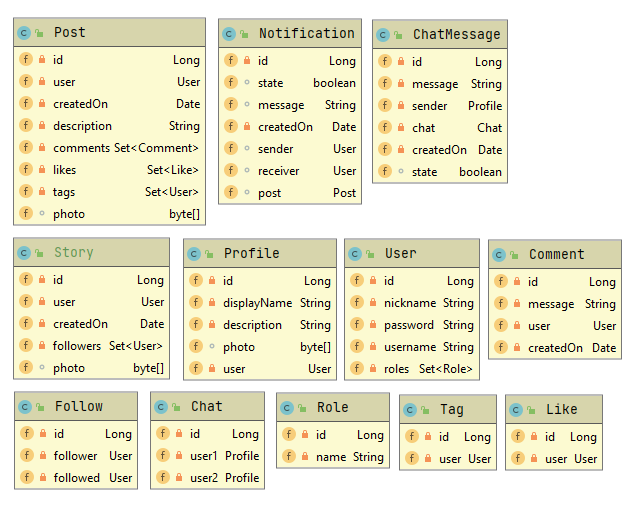
* @Entity este folosit pentru crearea tabelei
* @NoArgsConstructor, @AllArgsConstructor permit instanțierea unui obiect de tip Like folosind oricare dintre cele două tipuri de constructori.
* @Setter și @Getter ajută la scurtarea timpului de implementare, astfel se autogenerează funcții de Get și Set pentru fiecare atribut.
* Cheia unică este decorată cu adnotările @Id și @GeneratedValue
* Se stabilește o relație de tip @ManyToOne cu entitatea User.

Figura 11

În pachetul com.licenta.socialmedia.model am inclus clasele: Chat, ChatMessage, Comment, Follow, Like, Notification, Post, Profile, Rolem Story, Tag, User.

### Configuration

Pachetul configuration conține clase ce implementează funcții responsăbile cu configurarile necesăre aplicației . Astfel în acest pachet este creată clasă WebSecurityConfigurer ce este responsabilă cu restricționarea accesului la endpoint-urile create în controllere a utilizatorilor neautorizați, următoare funcție fiind cea care se ocupă de această acțiune:



Figura 12

* Variabila http va gestiona sesiunile de logare astfel pentru oricare din url-rile ce contin „/register”, „/”, „/socket/\*\*” și „/auth” se va permite accesul și fără verificarea header-ului request-ului. Orice alt request la backend va necesita autentificarea utilizatorului.
* Funcția de http.addFilterBefore(authenticationJwtTokenFilter(),UsernamePasswordAuthenticationFilter.**class**) va verifica creditentialele userului autentificat prin decodarea obiectului JWT și căutarea rezultatului în baza de date.

Clasă WebSocketConfigurer este responsabilă pentru inregistrarea url-urilor specific Web Socket, astfel ea implementeaza funcții care se ocupa cu configurarea aplicației . Urmatoarele funcții sunt esentiale în setarea unei astfel de conexiuni:

* + 1. Funcția registerStompEndpoints

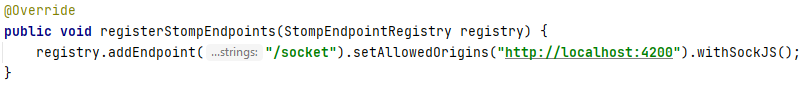


Figura 13

* Aceasta funcție este responsabilă de înregistrarea prefixului la care se vor face request-uri de către Web Socket.
* Permite requesturi de la url-ul **"http://localhost:4200"**
  + 1. Funcția configureMessageBroker

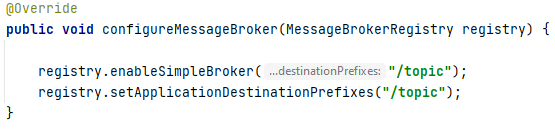


Figura 14

* Înregistrează prefixul end-point-urilor ce vor transmite informații

În pachetul com.licenta.socialmedia.configuration am inclus clasele: WebSecurityConfigurer și WebSocketConfigurer.

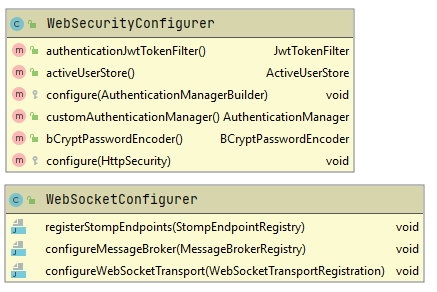


Figura 15

### Security

Acest pachet conține clase care gestionează partea de securitate a aplicației . Rolul acestor clase este de a implementa partea de securitate a punctului final prin tokenul JWT.

* JwtUtil este însărcinat cu analizarea token-ului în obiectul User și generarea token-ului din obiectul User.
* JwtEntryPoint este concepută pentru a trimite o eroare atunci când simbolul este nevalid
* JwtTokenFilter este punctul de intrare în procesul de autentificare JWT; filtrul extrage token-ul JWT din headerele requestului și delegă autentificarea către AuthenticationManager-ul injectat. Dacă jetonul nu este găsit, se aruncă o excepție care oprește requestul de la procesăre. De asemenea, avem nevoie de o suprascriere pentru autentificarea cu succes, deoarece fluxul Spring implicit ar opri lanțul de filtrare și ar continuă cu o redirecționare.
* UserDetailsImpl se ocupă de procesărea token-ului transmis și stabilirea dacă există său nu în baza de date un utilizator cu creditentialele obținute în urmă decodarii token-ului.

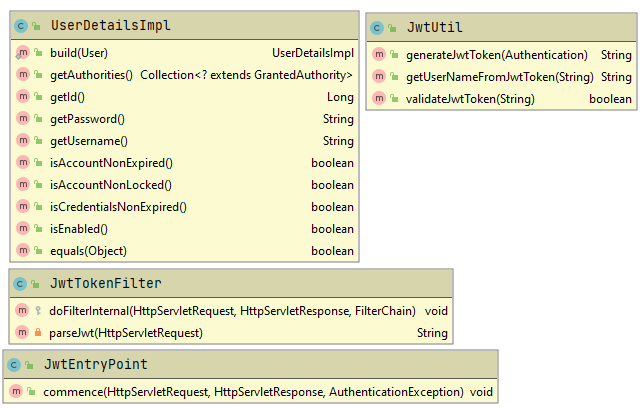


Figura 16

### DTO

Acest pachet contine obiecte folosite pentru a transfera informatii, obiecte cunoscute și ca DTO (Data Transfer Object). Acest pachet este structurat pe doua categorii de DTO: Response și Request.

Obiectele DTO de tip Response sunt folosite în Controllere pentru a transmite informatii cat mai complete.

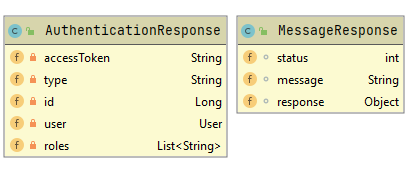


Figura 17

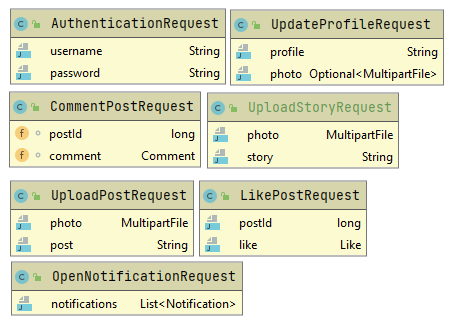
 Obiectele DTO de tip Request sunt create în scopul de a conține informațiile a mai multe obiecte, sau informatii ce nu pot fi convertite direct cu @ResponseBody intr-un anumit tip de entitate. Acest pachet contine clasele ilustrate în figura 18:

Figura 18

### Util

Acest pachet conține clase folosite pentru diverse scopuri precum conversia imaginilor său crearea de strîng-uri pentru endpoint-uri.

Clasă PhotoUtils este special creată pentru a facilita lucrul cu imagini în baza de date. La momentul înserării unei imagini în baza de date, această este convertitata într-un vector de biți. Pentru a optimiza procesul de adăugare și de aducere a imaginii, conversia biților este o soluție optimă. Astfel în momentul înserării se fă apela funcția de compressBytes:

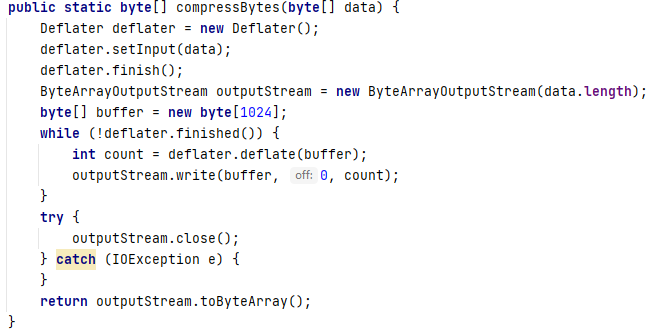


Figura 19

* Această funcție este responsabilă de micșorarea lungimii vectorului fără a pierde din calitatea imaginii

La momentul aducerii imaginii din baza de date, aceasta vine în forma să modificata, astfel este nevoie o decompresie a imaginii. De aceasta actiune este responsabilă funcția decompressBytes.

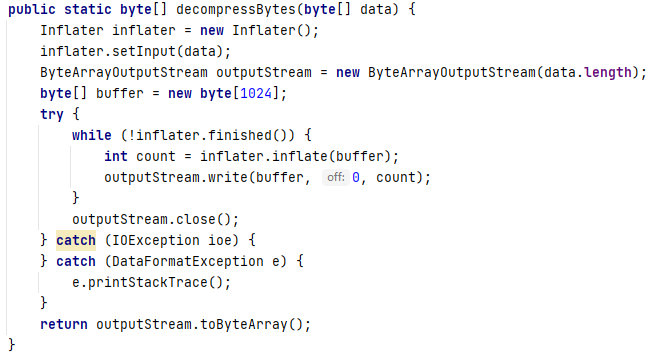


Figura 20

# **Arhitectura componentei frontend**

Componenta frontend a proiectului a fost dezvoltată cu ajutorul tehnologiei Angular. Paginile sunt create cu ajutorul HTML și CSS, iar funcționalitatea logică a fost scrisă în TypeScript.

Figura 20

Angular urmărește arhitectura bazată pe componente. în acest tip de arhitectura se urmărește modularizarea codului și divizarea acestuia pe cat mai multe pachete, cunoscute și sub denumirea de module, pentru a avea usor acces la fiecare componenta și pentru a dezvolta independent unele de altele fiecare componenta. Prin urmare, o aplicație mare este divizată (decuplată) în componente funcționale și logice, cu un cod scris intr-o maniera curata și care să respecte principiile MVC. Prin urmarirea principiului MVC se doreste crearea de componente cu responsăbilitati cat mai specifice și limitate, de scriere de cod specializat pe o serie limitata de actiuni. Un modul în aplicația Angular este un grup de componente, directive, pipe-uri și servicii, care sunt legate de aplicație și construiesc în mod colectiv o funcționalitate comună.

Angular oferă o mulțime de module de construcție, cum ar fi modulul http (pentru a efectua apeluri http din aplicație. Este obligatoriu ca fiecare aplicație Angular să aibă un modul root. Fiind o aplicatie web cu o singura pagina, Componenta AppComponent reprezinta de fapt containerul în care se vor folosi pe rand, în funcție de actiunile și alegerile utiliatorului, partile funcționale și vizuale ale componentelor definite. Aceasta componenta este de fapt componenta principala a aplicației , care se află în fișierul app.component.ts.

Fiecare componentă, serviciu, directivă, pipe și model utilizate în aplicație trebuie declarate în fișierul app.modules.ts, altfel va fi aruncată o eroare în consola browserului. Fișierul app.routing.modules.ts conține AppRoutingModule care configurează routerul de rutare. Ruta îi spune routerului ce vizualizare trebuie afișată pentru fiecare legătură utilizată în aplicație, specifică adresă URL din cale, iar în componentă specifică componentele pe care routerul trebuie să le creeze atunci când navighează la calea specificată în cale. Imaginile folosite în aplicație sunt în „src / assets / images”.

Pentru a avea o mai buna evidență a locației componentelor, proiectul este structurat pe pachetele: component, service, pipe, model.

## Pachetul component

Acest pachet conține toate interfetele grafice ale aplicației , o componentă reprezentând de fac o colecție logică de template-uri HTML, de stiluri asociate și modele care ar putea să opereze într-un template folosind îmbinările/legăturile din Angular.

O componentă în Angular este un element de construcție al UI și constă dintr-un fișier de clasă, un template și un fișier CSS pentru stilizarea componenței. Clasă este scrisă în limbaj TypeScript și este decorată cu adnotarea @Component, că în figura 21, ce este utilizată pentru a specifică metadata clasei.

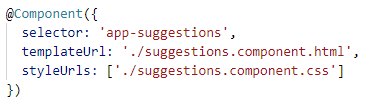


Figura 21

Decoratorul @Component () specifică următoarele informații specifice Angular:

* Un selector care definește modul în care este utilizată componentă într-un alt template. Acest selector are o denumire unică, astfel în momentul în care în interiorul unui fișier HTML se va crea un element, spre exemplu, de tipul „<app-suggestions/>”, aplicația va știi că este necesăra înlocuirea spațiului alocat cu componentă specificată.
* Un fișier HTML care instruiește Angular cum să redea componentă. Cu acest template se construiește partea vizuală a componenței ce poate conține informații în tabele, diferite tipuri de elemente de introducere a datelor precum casuțe de scriere, date-picker-uri, butoane etc. Tot în această componentă se pune în valoare funcția de dată binding pe care Angular-ul o pune la dispoziție, astfel eliminându-se mult cod. Sincronizarea elementelor cu variabilele și funcțiile din clasă componenței se face dinamic.
* Un set opțional de fișiere CSS care definesc aspectul elementelor HTML ale tempalte-ului.

Componentele pot implementa funcții ce ajută la manipularea informației și modificarea template-ului precum funcția “ngOnInit()”. Această este responsabilă de aplicarea de diverse acțiuni asupra atributelor utilizate în Dată Binding înainte că fișierul HTML să construiască elementele vizuale.

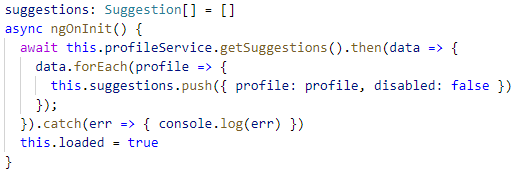
În figura 22:

Figura 22

* În momentul inițializării componenței, lista “suggestions” nu conține niciun element. Această fiind setată cu o lista goală.
* În interiorul funcției “ngOnInit()” se apelează funcția asincronă getSuggestions() din serviciul ProfileService, ce va adduce din baza de date informațiile dorite.
* Cu ajutorul funcției “forEach()” se parcurge fiecare element din rezultatul obținut și se va adaugă în lista “suggestions”
* Funcția “ngOnInit()” este declarată că fiind asincronă deoarece este necesăra folosirea cuvântului cheie “await” pentru a aștepta executarea funcției asincrone “getProfiles()”, o abordare diferită, în care nu se va folosi perechea “async-await” poate genera probleme grave aplicației dacă viteză de aducere a datelor nu este una mare.

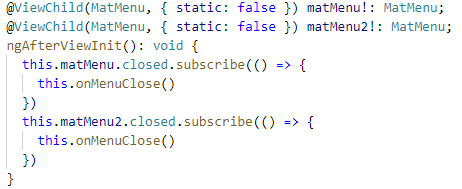
Funcția “ngAfterViewInit()” se ocupă cu modificarea atributelor după momentul creării componentelor UI.

Figura 23

* În interiorul funcției se va stabili comportamentul elementului de UI “MatMenu”, care la momentul închiderii va apela funcția “onMenuClose()” ce va produce modificări asupra componenței.

De asemenea fiecare componentă conține câte un constructor ce este responsăbil de instanțierea atributelor din cadrul clasei.Aceste componente sunt reutilizabile, prin urmare pot fi utilizate în orice altă parte a aplicației . De asemenea ele sunt independente, prin urmare pot fi testate independent, această arhitectură face codul foarte testabil.

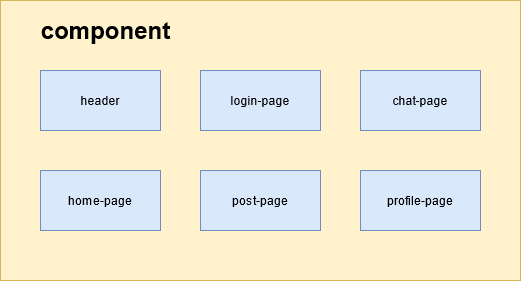
In pachetul component am definit componente pentru fiecare pagina de care am avut nevoie în platforma Social Media.

Figura 24

### Componenta Header

Această componentă conține butoane și elemente necesăre peste tot în aplicație.

Butonul de „home” redirecționează aplicația către componentă unde se pot vizualiza postările create de ceilalți useri, sugestiile de urmărire și story-urile. Redirecționarea la componentă Home se face adăugând în template-ul HTML „routerLink="/home”” în declararea elementului img ce continue și locația imaginii.

Un câmp special de căutare permite utilizatorilor să caute alți ultilizatori ai aplicației prin introducerea numelui complet al acestora său a username-ului cu care sunt înregistrat. Pentru a oferi sugestii pe măsură ce noi caractere sunt introduse, am folosit elemental

Autocomplete din Angular Material, care pe măsură ce se modifică șirul de caractere introdus, va sugera o serie de recomandări ce corespund căutării. Pentru implementarea acestei funcționalități a fost necesăra adăugare unui element HTML de tip input, care în momentul modificării va apela funcția "onSearchChange()". De asemenea, crearea elementului de Autocomplete necesită o structura aparte, după cum ilustrează și figura 25:

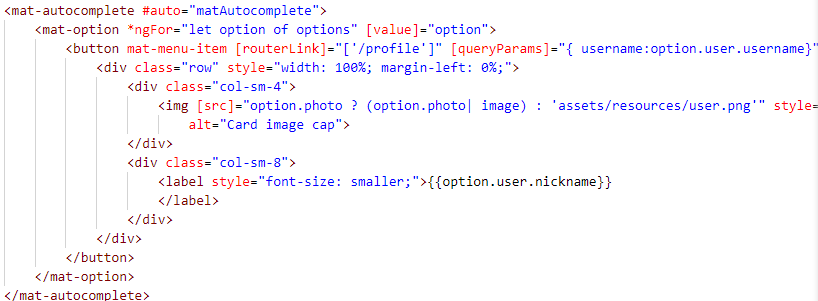


Figura 25

* Secvența “\*ngFor="let option of options"“ este responsabilă de parcurgerea tuturor elementelor din lista de obiecte “options”, lista filtrată în momentul tastarii în câmpul de căutare.
* [value]="option" atribuie valorii selectate obiectul option, astfel se vor reține în întregime toate atributele aflate în option.
* Structura button are că scop afisărea datelor despre utilizatorul căutat, iar în momentul în care este selectat, apansănd click, una dintre sugestii, atunci se va naviga către componentă “ProfileComponent” unde se va descifra obiectul trimis prin “[queryParams]="{ username:option.user.username}"”

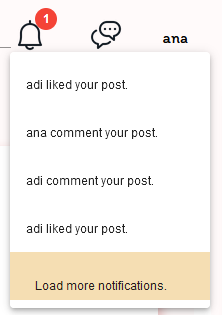
O componentă pentru notificări este de asemenea adăugată în header. Această contine un badge ce, în momentul în care se crează noi notificări destinate utilizatorului logat, se modifca. Această componentă conține și o lista cu notificări filtrate pe utilizatorul logat și pe dată la care acestea au fost create.

Figura 26

După cum este ilustrat în figura 26, lista de notificari contine un număr limitat de notificari, iar în momentul în care se apasă pe butonul de “Load more notifications.”, se vor aduce urmatoarele notificari, filtrate după date crearii.Pentru identificarea noilor notificari, componenta HeaderComponent se aboneaza în constructor la metoda „findNew()” din serviciul NotificationService ce mentine constanta comunicarea cu backendul pentru end-point-ul „/topic/notifications/new/”, careia i se concatenteaza și id-ul utilizatorului logat. Acest endpoint este inregistrat pentru a deservi la conexiunea realizata de serviciul WebSocketService.

Similar cu componentă de notificări este creată și componentă de chat, această fiind notificată de fiecare dată când se crează un nou mesaj destinate utilizatorului logat. Ultimele mesaje sunt filtrate astfel încât utilizatorul să vadă în ordine descrescătoare dații la care au fost create toate mesajele. Mesajele necitite sunt evidențiate. Iar utilizatorul are posibilitatea de a accesă pagină destinate chat-ului unde și poate selecta mesajele pe care vrea să le deschidă.

După cum este ilustrat în figura 26, lista de notificări conține un număr limitat de notificări, iar în momentul în care se apasă pe butonul de “Load more notifications.”, se vor aduce următoarele notificări, filtrate după date creării.Pentru identificarea noilor notificări, componentă HeaderComponent se abonează în constructor la metodă „findNew()” din serviciul NotificationService ce menține constanța comunicarea cu backendul pentru end-point-ul „/topic/notifications/new/”, căreia i se concatenteaza și id-ul utilizatorului logat. Acest endpoint este înregistrat pentru a deservi la conexiunea realizată de serviciul WebSocketService.

Tot în componentă Header, utilizatorul are posibiliatatea de a-și accesă profilul personal folosing elementele de UI illustrate în figura 27:

Figura 27

Aceste elemente folosesc secvență routerLink="/profile" [queryParams]="{username:profile.user.username}" pentru a naviga pe pagină utilizatorului logat, unde se pot vizualiza mai multe detalii.

Cea din ultima acțiune ce se poate execută folosind elementele prezentate în componentă Header este delogarea din sesiune actuală. Butonul de delogare, ilustrat în figura 28, apelează funcția ce este responsabilă cu ștergerea itemului “currentProfile” din localStorage și resetarea token-ului de autentificare.

logout() {

 localStorage.removeItem('currentProfile');

Figura 28

this.authenticationService.logout()

}

### Componenta Profile-Page

Componenta Profile-Page conține subcomponente specializate pentru afișarea anumitor detalii despre profilul accesat. Aceste subcomponente sunt reprezentate de componentele ProfileInformationComponent, ProfilePostsComponent, EditProfileComponent și FollowDetailsComponent. Aceste componente sunt combinate în ProfileComponent, acesta reprezentând componentă principala.

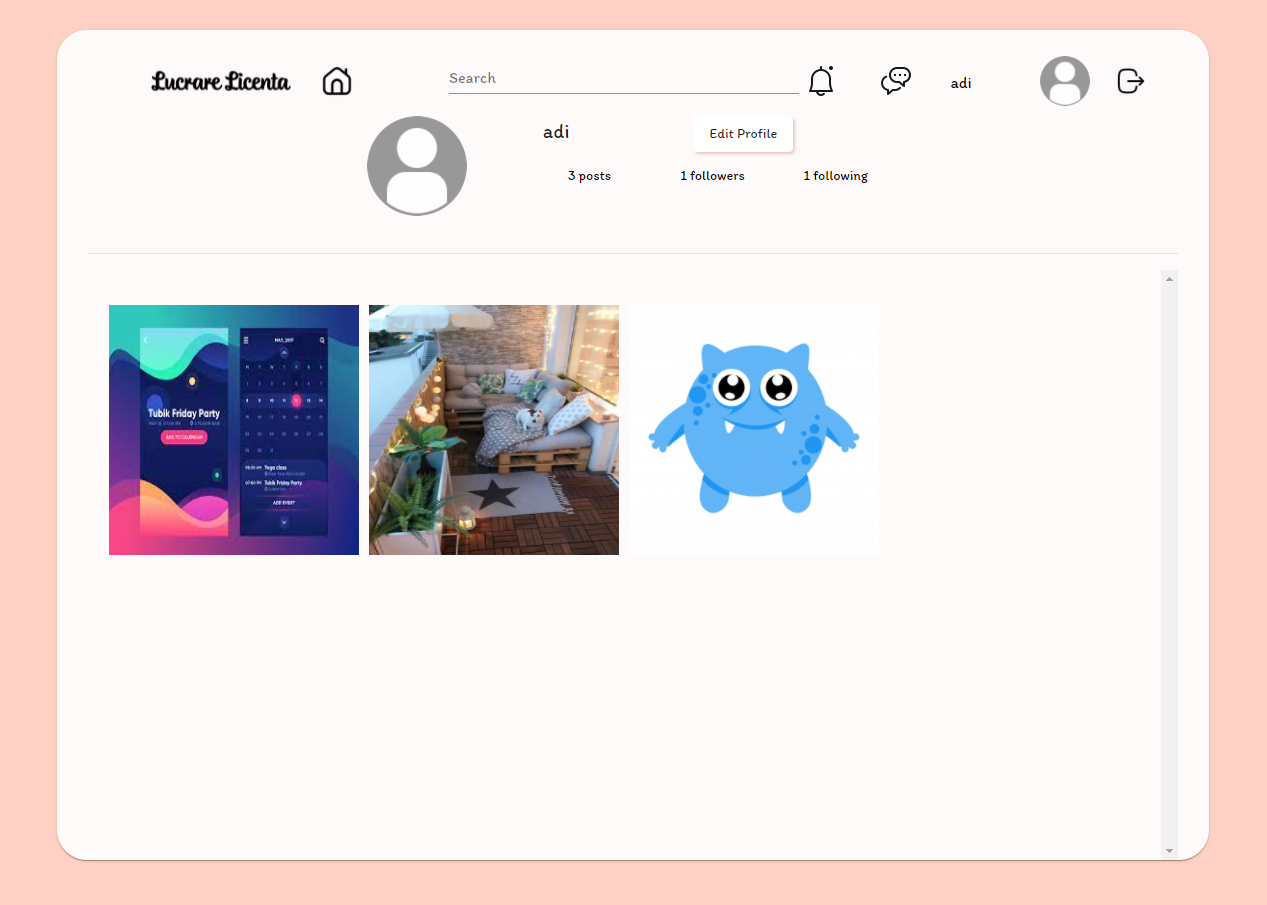
În figura 29 este prezentată interfață vizuală a componenței profile, după cum se poate observă, componentă Profile conține informații referitoare la profilul accesăt precum: usernamel user-ului, fotografie de profil al acestuia, numărul de postări create, numărul de persoane urmărite și de urmăritori, numele complet și o descriere personalizată. Utilizatorul își poate modifică date profilului dacă profilul accesăt este cel personal, în caz contrat atunci poate trimite un request de urmarie apasănd butonul „Follow” și poate vizualiza toate postările create de utilizatorul cu profilul accesăt. în cele ce urmează voi discuta despre fiecare sub-componentă din componentă Profile:

Figura 29

#### ProfileInformationComponent

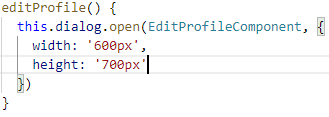
Această componentă primește username-ul trimis prin queryParams la navigare și inițializează componenta UI astfel încât atributele să corespundă profilului accesat. În funcția „ngOnInit()” se verifica dacă utilizatorul logat este același cu cel trimis prin queryParams.

Figura 30

Dacă verificarea are un rezultat pozitiv atunci atributul “profile” va fi intializat utilizandu-se funcția “getPersonalProfile ()” din ProfileService, iar variabilă ce este responsabilă cu afisărea butonului de “Edit Profile”, variabilă “isPersonalProfile, se va seta cu valoarea true. Dacă această valoare este adevărată atunci utilizatorul își va putea modifică detaliile profilului apelând funcția din figura 30 ce va deschide o fereastră cu componentă EditProfileComponent.

În cazul în care variabilă isPersonalProfile este setată că fiind false butonul de “Edit Profile” va fi înlocuit de butonul “Follow”, a cărui funcționalitate am enunțat-o anterior, iar atributul “profile” se va inițializă cu profilul rezultat în urmă apelării funcției “getProfile(User)” ce va returna profilul accesăt. Se va mai face o verificare pentru a stabili dacă profilul accesăt este deja urmărit de utilizatorul logat, iar în cazul în care se stabilește că această verificare este adevărată atunci butonul de “Follow” va fi înlocuit cu un buton de “Unfollow”.

Tot în această componentă putem deschide liste de profile urmărite sau profilele urmăritorilor cu ajutorul funcției din figura 31:

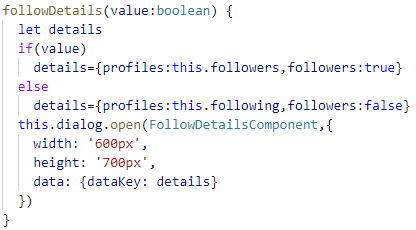


Figura 31

* Parametrul „value” primește din template-ul HTML tipul listei de afisăt, dacă utilizatorul solicită să vizualize lista de persoane ce îi urmăresc profilul atunci parametrul va fi setată cu valoarea true, în caz contrat acesta va primi valoarea false.
* În cadrul funcției se verifică valoarea parametrului „value” iar în funcție de rezultat se inițializeaza variabilă locală „details” cu lista de obiecte de tip Profile solicitată și variabilă „followers” cu valoarea true în cazul în care se dorește vizualizarea listei de urmăritori său false în caz contrar.
* Funcția ‚”this.dialog.open()” se ocupă cu deschiderea unei ferestre ce conține componentă FollowDetailsComponent, căreia îi este injectată informația creată în variabilă „details”.

#### ProfilePostsComponent

În această componentă se va stabili profilul acesăt, în aceeași manieră cum s-a stabilit și pentru ProfileInformationComponent. în cadrul funcției „ngOnInit()” se vor încarcă datele necesăre din PostService pentru profilul accesăt și se va intializa lista de obiecte de tip NewsFeedPost, posts, cu ajutorul funcției „getPersonalPosts()” ce va returna exclusiv postările create de utilizatorul dețînător al profilului.

După intializarea listei posts, această va fi parcursă element cu element în cadrul ngFor din fișierul HTML unde fiecare postare va fi încadrată folosind „mat-grid-list”, componentă ce permite afisărea tabulara a imaginilor. Pe fiecare dintre elemente le vom putea vizualiza într-o fereastră separată, apasănd pe imaginea reprezentativă postării. în urmă acestei acțiuni se va apela funcția din figura 32 ce este responsabilă cu deschiderea unei ferește ce conține detalii despre postare.

Figura 32

#### EditProfileComponent

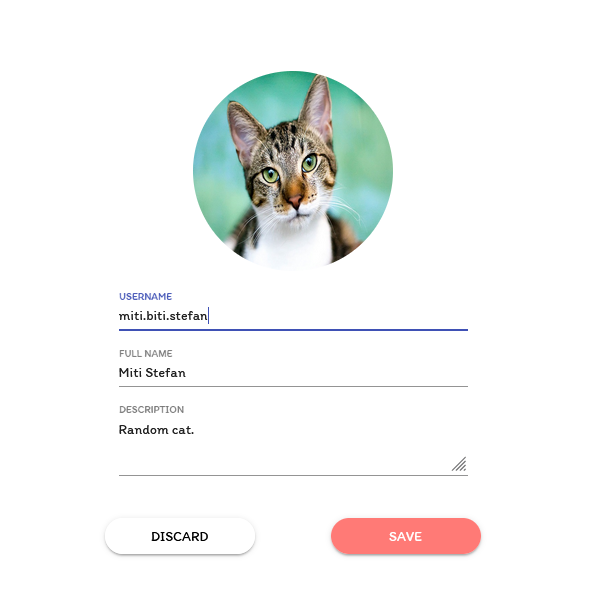
Componenta responsabilă cu modificarea detaliilor existente despre profilul utilizatorului, detalii precum fotografia de profil, username-ul, numele complet și descrierea.

Figura 33

Interfața grafică a acestei componente este ilustrată în figura 33. Setarea imaginii se face cu ajutorul unui câmp de tip input ce este restricționat să primească doar fișiere de tip imagine.

În momentul în care este selectata o imagine din browser-ul deschis se vă apela funcția de „preview()” ce vă modifica calea imaginii actuale cu noua cale a imaginii convertite în base64. Variabila „selectedFile” este reinitializata cu noul fișier selectat.

În momentul apasării butonului „Save” se va apela funcția „saveProfile()” ce este responsabilă de trimiterea parametrilor de tip Profile și File către metoda “updateProfile” din ProfileService. Butonul “Discard” închide fereastra fară a executa operația de modificare a profilului.

#### FollowDetailsComponent

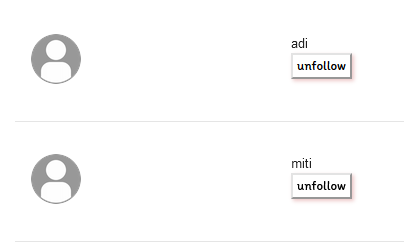
Prin această componentă, utilizatorul poate vedea atât lista de urmăritori cât și lista de profile urmărite, tipul listei se stabilește în ProfileInformationComponent. În figura 34 este ilustrată o lista de persoane urmărite, cărora utilizatorul logat le poate aplica acțiunea de „unfollow”. Iar în figura 35, profilelor din lista utilizatorul le poate da „remove”.

Figura 34

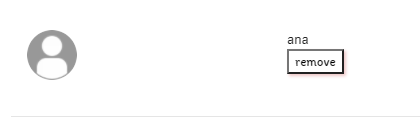


Figura 35

### Componenta Login

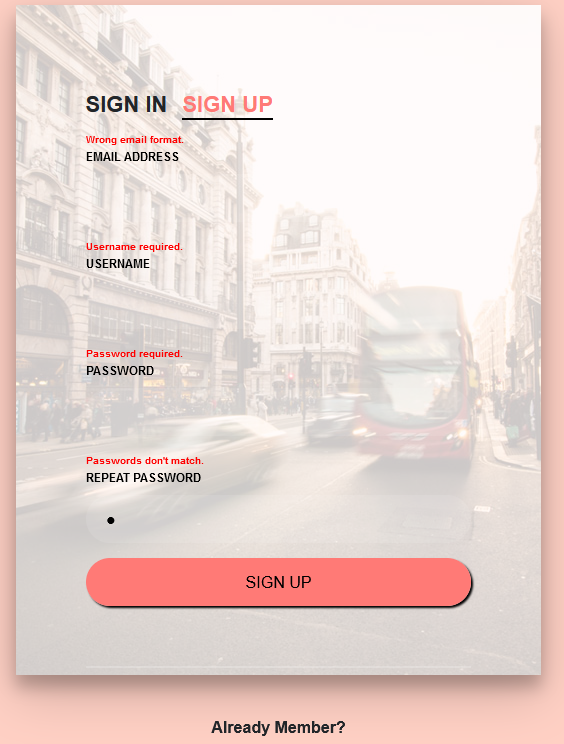
Această componentă este responsabilă de înregistrarea și autentificarea utilizatorului. Această componentă este cea care este apelata în momentul deschiderii aplicației . În interiorul constructorului se verifica dacă este deja creată o sesiune de logare pentru utilizator, verificându-se dacă funcția „getCurrentUser()” din AuthenticationService returnează său nu o valoare. În cazul în care rezultatul funcției este diferit de „undefined” atunci se consideră că există o sesiune de logare încă valabilă și se vă naviga spre componenta HomeComponent. În caz contrar utilizatorul are puse la dispoziție următoarele scenarii:

Figura 36

Utilizatorul își poate crea cont în secțiunea „SIGN UP”:

* Utilizatorul trebuie să-și introducă un email, un username (care vă fi echivalentul nickname-ului în aplicație), trebuie să introducă o parola și să o repete.
* Parolele trebuie să corespunda.
* Toate câmpurile menționate sunt obligatorii, în cazul în care un câmp este omis, nu se vă putea trimite request-ul responsăbil de înregistrarea utilizatorului după cum se poate observa și în figura 36.
* S-a folosit un FormGroup pentru crearea câmpurilor necesăre înregistrării și pentru aplicarea validatorilor.
* În momentul completării corecte a câmpurilor, Authentication Service va apela funcția „register()” ce va primi ca parametru datele din registerForm. Prin abonarea la această funcție se așteapta un răspuns de la server, în cazul în care răpsunsul are statusul 201, atunci se va face logarea automată cu noul cont creat.

Utilizatorul se poate autentifica în secțiunea „SIGN IN”:

* Pentru crearea câmpurilor de introducere a e-mailului și a parolei am folosit un FormGroup denumit loginForm. Acesta este responsabil de validarea câmpurilor și nu permite apelarea funcției de „login()” a AuthenticationService dacă datele introduse nu sunt valide.
* Dacă loginForm este valid, atunci se permite apelarea funcției „login()”, iar dacă răspunsul server-ului este folosit pentru înregistrarea în localStorage a detaliilor referitoare la utilizatorul logat și se va naviga către componenta HomeComponent.



Figura 37

### Componenta Home-Pages

Această componentă este divizata în alte șase subcomponente ce sunt apelate din componenta principală HomeComponent. În cele ce urmează voi aborda individual fiecare componentă:

#### HomeComponent

Prin componenta HomeComponent se pun la dispoziție trei componente după cum ilustrează și figura 38, această componentă deservind drept container pentru a permite utilizatorului să poată vizualiza o varietate de elemente diferite precum sugestii de urmărire, postările create de persoanele urmărite, posibilitatea de a crea postări și poveștile adăugate de utilizatorii urmăriți. Puse împreună, aceste componente alcătuiesc imaginea din figura 38:

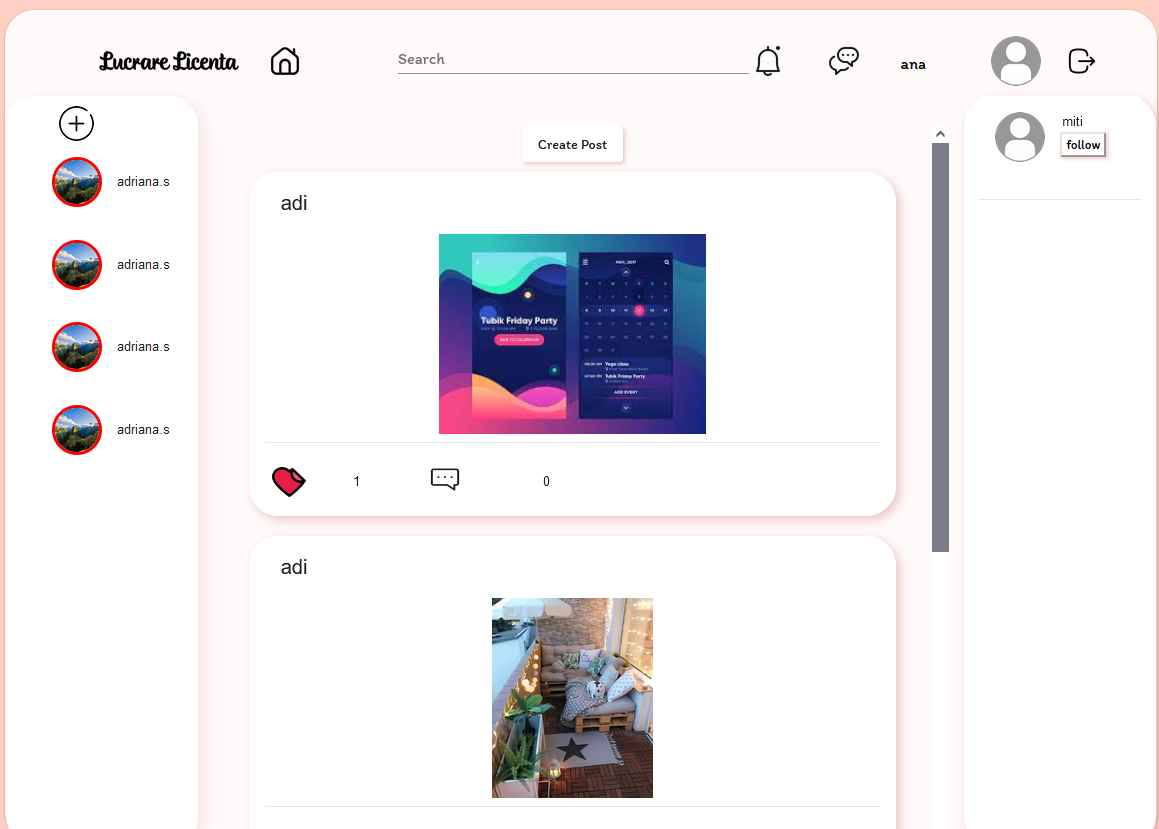


Figura 38

#### NewsFeedComponent

Această componentă este responsabilă de aducerea celor mai recente postări adaugate de utilizatorii pe care utilizatorul logat îi urmărește. Aici utilizatorul are posibilitatea de a crea noi postări, de a aprecia postările de pe pagina și de a încărca mai multe postări pe măsură ce acesta navighează printre postările existente. În cele ce urmeaza, voi enumera cele mai importante funcționalități din această componentă.

* Butonul „Create Post” este responsabil de deschiderea unei ferestre ce conține componenta CreatePostComponent, unde utilizatorul are posibilitatea de a adauga o nouă postare. De deschiderea acestei ferestre este responsabilă funcția „createPost()”, ilustrată în figura 39:

Figura 39

* Afișarea postărilor create de utilizatorii urmăriți se face prin apelarea funcției „findAll()” din cadrul serviciului PostService, unde se va face request către endpoint-ul responsabil de aducerea postărilor solicitate. Prin abonarea la aceasta funcție se așteaptă răspunsul de la backend, răspuns care va fi convertit și adăugat în lista „posts”. Funcția responsabilă de convertire, verifică dacă postarea este apreciată de utilizator, stabilește numărul de aprecieri și de comentarii și crează un obiect de tip NewsFeedPost.
* Utilizatorul poate aprecia o postare apăsând butonul în forma de inima care va apela funcția „onLike” din figura 40, în cadrul funcției se va incrementa numărul de aprecieri după ce se va face abonarea la funcția „likePost”, responsabilă de trimiterea request-ului, ce conține informații despre apreciere, către backend.

Figura 40

* În cazul în care utilizatorul vrea să-și retragă aprecierea atunci acesta poate apasă pe butonul în forma de inima, ce este colorat cu rosu, și se va apela funcția „onUnlike”, a cărei funcționalitate este similară cu cea de „onLike”, diferențele constant în faptul ca se va decrementa numărul de aprecieri, iar request-ul responsăbil de ștergerea aprecierii va apela un endpoint specializat pe aceasta actiune.
* In momentul în care utilizatorul ajunge la finalul listei de postări, prin bottom-scroll, atunci va putea observa un buton „Load more”, care în momentul apasării va face un request catre backend ce va returna urmatoarele postări, în ordinea lor cronologica. Prin apasărea butonul se va apela funcția „loadMorePosts” care va incrementa și variabila „loadMorePostsRequestNumber”, variabila utilizata pentru paginarea postărilor în backend.
* Dacă un utilizator urmarit adauga o noua postare, atunci în aceasta pagina va aparea un buton ce permite incarcarea celor mai recente postări. Aceasta funcționalitate este posibila prin abonarea la sunctia „onPost()” din PostService, unde serviciu de Web Socket inregistreaza endpoint-ul '/topic/posts/created/', căruia I se concateneaza și id-ul utilizatorului logat. Cu acest endpoint se comunica din backend și se notifica în momentul în care un utilizator urmarit creaza o noua postare.
* Prin apasărea pe imaginea unei postări se va deschide o fereastra cu componenta PostPopupComponent, unde sunt prezentate detalii despre postare.

#### SuggestionsComponent

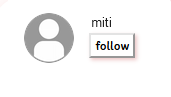
Aceasta componenta este responsabilă de afisărea utilizatorilor pe care utilizatorul logat nu îi urmărește. în funcția „ngOnInit()” este apelata funcția „getSuggestions()” din ProfileService, iar prin abonarea la aceasta funcție se va folosi rezultatul pentru a initializa lista suggestions. în momentul apasării butonului „follow”, ilustrat în figura 41 se va apela funcția „follow()”.

Figura 41

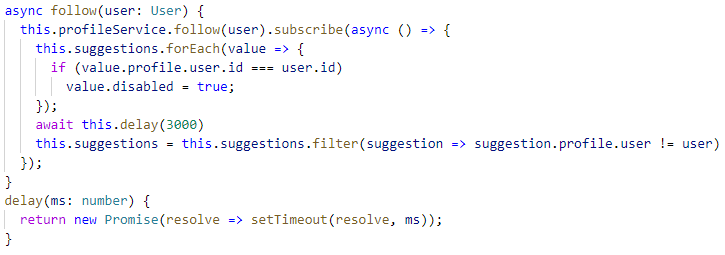
Funcția „follow()” are urmatoarea structura:

Figura 42

* Prin abonarea la funcția „follow()” din ProfileService, ce primeste ca parametru utilizatorul ce urmeaza să fie urmarit, se va dezactiva butonul de „follow” asociat utilizatorului selectat. După trei secunde se va refiltra filtra de sugestii astfel incat să nu contina utilizatorul selectat.

#### CreatePostComponent

Aceasta componenta este utilizata din componenta NewsFeedPostComponent și este folosita pentru crearea unei noi postări, continand un câmp input de tip file, un câmp pentru descriere și un autocomplete cu ajutorul căruia se pot eticheta utilizatori în postare.. Interfata vizuala este ilustrată în figura 43, și pe baza acesteia voi explica funcționalitatile existente în componenta:

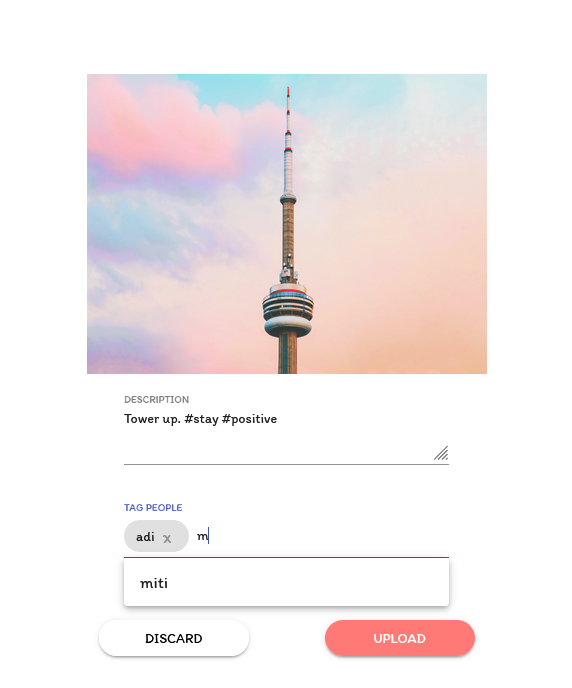
* Adaugarea de imagini se face cu ajutorul unui câmp input ce poate primi doar fișiere de tip imagine. Inainte de selectarea unei imagine, sursă componentei img va contine o imagine săblon, astfel incat utilizatorul să-si dea seama cu usurinta de faptul ca trebuie să apese pe imagine pentru a selecta un fișier. în momentul în care se apasă pe imagine se va apela funcția „preview()”, care este responsabilă de modificarea sursei elementului „img” și modifica valoarea atributului „selectedFile” cu noul fișier selectat.

Figura 43

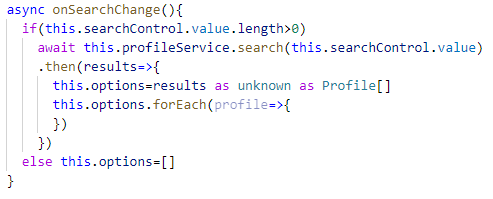
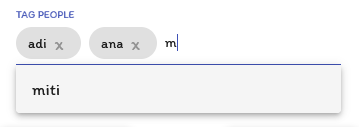
* Câmpul pentru descriere este reprezentat de un element „textarea” ce reprezinta un câmp de tip input cu o inalțime marita.
* Utilizatorul are posibilitatea de a eticheta persoane folosind câmpul indicat de labelul „TAG PEOPLE”. Acest câmp este o combinatie intre un element de tip Autocomplete și un element de tip MatChipList. Autocomplete-ul este responsabil de filtrarea și aducerea sugestiilor, reprezentate de alți utilizatori, pe măsură ce în câmpul input se tasteaza. Filtrarea datelor se face cu ajutorul funcției din figura 44, astfel; se va reinițializa de fiecare dată lista „options” cu răspunsul request-ului creat prin funcția „search” din serviciul ProfileService. În momentul în care se selectează utilizatorii ce se doresc a fi etichetați se va adauga în lista „chipBoxUsers” valoarea selecției. Parcurgând aceasta listă în elementul „mat-chip” se pot observa persoane etichetate, după cum ilustreaza figura 45:

Figura 45

Figura 44

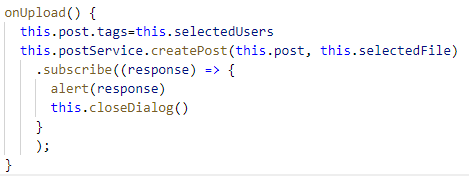
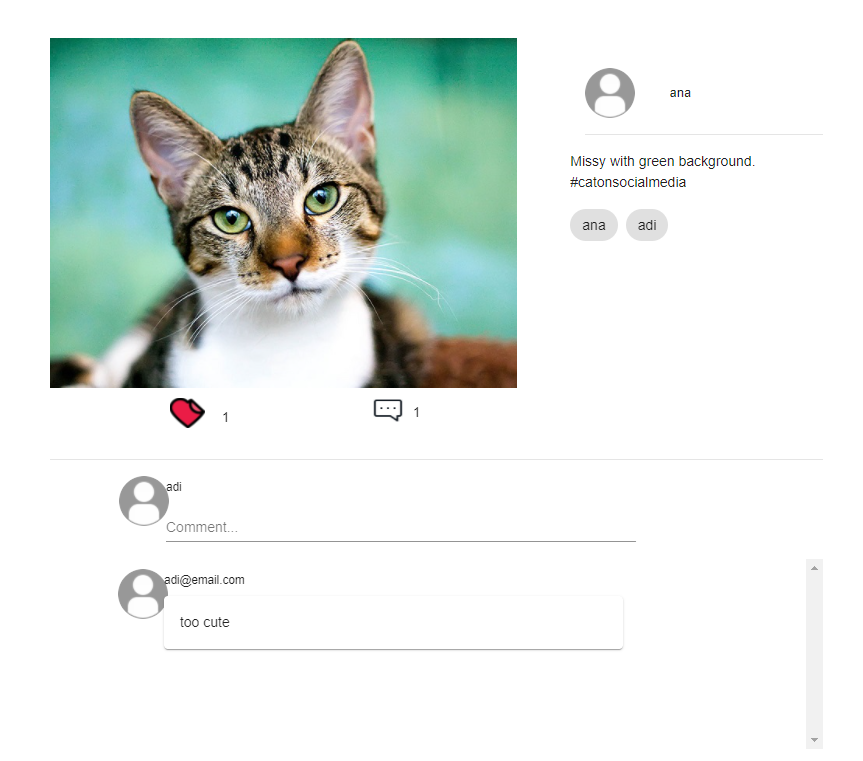
* Utilizatorul are posibilitatea de a șterge persoanele etichetate folosind semnul „X” de pe etichetă, vizibil și în figura 45, care va apela funcția de „remove”, responsabilă de ștergerea etichetei din lista „chipBoxUsers”.
* Utilizatorul poate renunța la crearea postării, apasănd butonul ‚”DISCARD” ce va inchide fereastra.
* Salvarea postării se face cu ajutorul butonului „UPLOAD” ce va apela funcția din figura 46 ce va trimite catre funcția „createPost”, postarea nou creată și fișierul selectat.

Figura 46

Figura 47

### Componenta Post-Pages

Cu această componentă utilizatorul poate observa toate detaliile legate de o postare, după cum ilustrează și figura 47. În aceasta componentă sunt prezentate urmatoarele elemente:

* Poza și nickname-ul utilizatorului care a creat postarea, apansând pe oricare dintre cele două se va naviga spre profilul acestui utilizator.
* Descrierea postării și persoanele etichetate
* Poza incarcată postării
* Numărul de aprecieri și comentarii, iar iconița în forma de inima este colorată după ce se verifică dacă această postare este apreciată sau nu de utilizatorul logat.
* Un câmp unde utilizatorul poate adauga comentarii postării. Comentariile sunt trimise catre server, care va notifica creatorul postării în legatura cu aceasta acțiune, și se va incrementa numărul de notificari nedeschise. De asemenea, comentariile apar în timp real, astfel utilizatorii pot să vada comentariile proaspat adaugate și să continue șirul de comentarii.
* Lista de comentarii ce au fost adăugate

### Componenta Chat-Pages

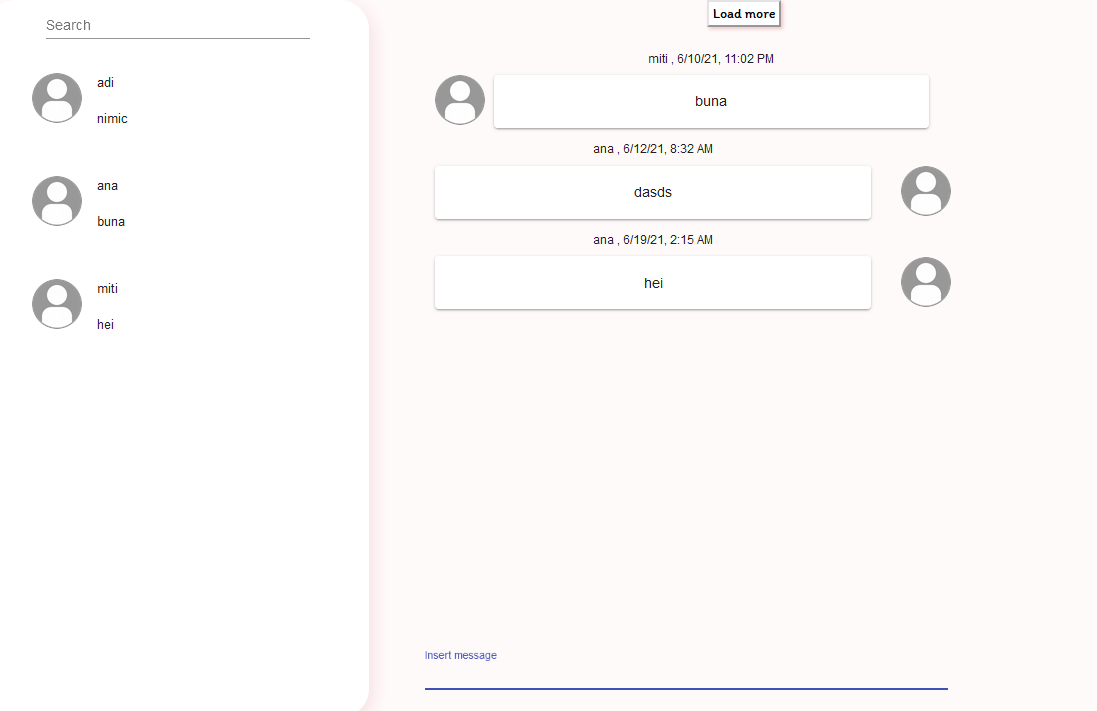
Această componentă este alcatuită din componentele ActiveChatsComponent și CurrentChatComponent și împreună formează vizual pagina ilustrată în figura 48

Figura 48

#### ActiveChatsComponent

Această componentă reprezintă partea din dreapta din figură , și conține un câmp de căutarea ce implementeaza o componentă Autocomplete și o lista de obiecte de tip DisplayChats.

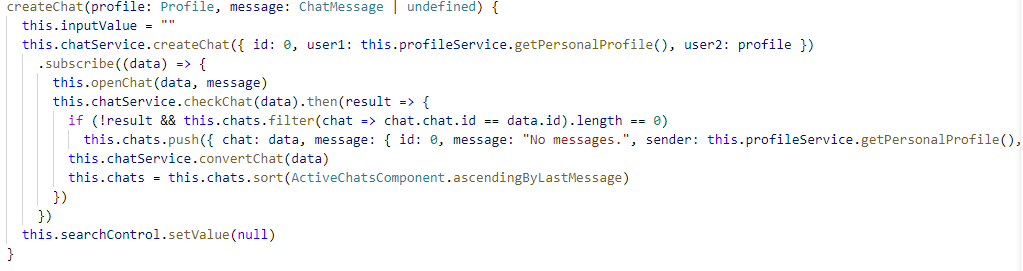
* Autocomplete-ul este responsabil de căutarea tuturor utilizatorilor a căror nickname sau numele complet corespund valorii adaugate. În momentul selecției unei sugestii din lista generată de componentă, se apeleaza funcția „createChat()”, ilustrată în figura 49, în aceasta funcție se verifica dacă există deja creată o conversație între cei doi utilizatori și crează o conversație după un șablon dacă nu există. Se sortează conversațiile astfel încat conversația selectată să fie prima în lista.

Figura 49

* În momentul selectării unei conversații, fie ea din lista deja existentă sau în urma selectarii unei sugestii din Autocomplete se apeleaza funcția „openChat” din figura 50, ce este responsabilă de trimiterea catre componenta parinte, ‚”CurrentChatComponent” a informațiilor despre conversația selectată. Se va crea o copie a conversației selectate folosind succesiunea de funcții „JSON.parse(JSON.stringify(chat))”, astfel se va evita modificarea nedorită a conversației selectate.

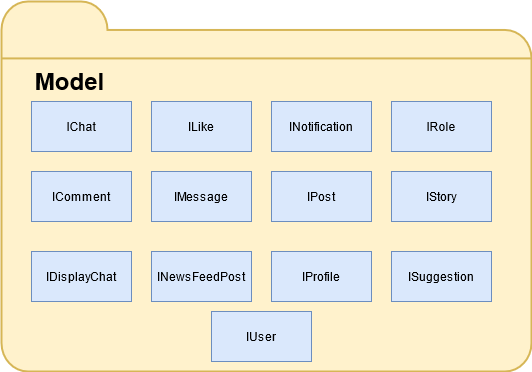
Figura 50

* Folosind serviciul WebSocketService, s-a realizat abonarea la endpoint-ul responsabile de notificarea utilizatorului în momentul în care se trimite către acesta un nou mesaj și se vor resporta conversațiile în ordine descrescatoare, după câmpul „createdOn” al ultimului mesaj trimis.
* Toate conversațiile a carui ultim mesaj este nedeschis de utilizatorul căruia îi este destinat sunt evidențiate fată de restul. În momentul deschiderii acestora se va trimite către server ultimul mesaj din conversație, modificat cu atributul state setat pe „true”, astfel în baza de date se va actualiza mesajul și se va considera în continuare ca fiind citit.
* În momentul afișării componentei, aceasta va solicita server-ului afișarea unui număr finit de conversații. În cazul în care utilizatorul dorește să parcurgă din ordine descendentă conversațiile, acesta poate opta să încarce mai multe conversații, folosind butonul „Load more”.

## Pachetul Model

În acest pachet sunt create interfețe folosite pentru transferul de date cu backend-ul. Aceste interfețe sunt identice entităților create în backend deoarece în momentul transmiterii de obiecte către endpoint, corpul request-ului trebuie să conțînă obiecte ce pot fi deserializate instant în tipul cerut în parametrii. Astfel, dacă în backend am o entitate definită de un id de tip Long și o denumire de tip Strîng, interfață declarată în pachetul Model corespunzătoare acestei entități trebuie să conțînă de asemenea aceleași atribute de același tip. De asemenea acest pachet conține și interfețe ce au scop de a ajută în prezentarea într-o manieră mai ușoară a anumitor modele, cu cât mai multe detalii. Situație întâlnită la interfață NewsFeedPost ce conține un atribut de tip Post, dar și numărul de aprecieri, numărul de comentarii și verificarea dacă postarea este apreciată, său nu, de utilizatorul logat.

Figura 51

****Acest pachet conține fișierele chat.ts, comment.ts, displaychat.ts, like.ts, message.ts, newsfeedpost.ts, notification.ts, post.ts, profile.ts, role.ts, story.ts, suggestion.ts, user.ts. Interfețele sunt ilustrate și în figura 51:

## Pachetul Service

Acest pachet conține toate componentele concepute pentru a apela endpoint-urile din backend. Rolul majorității serviciilor este de a apela endpoint-urile de la componente back-end pentru a citi, adăuga, edita și șterge date.

Printre servicii, se remarcă serviciile din pachetul auth-service ce se ocupă de partea de autentificare și securitate a interfeței. Serviciul de autentificare “AuthenticationService” verifică adresă de e-mail și parolă utilizatorului pe pagină de autentificare. .În cadrul funcției “login” din acest serviciu se face verificarea creditentialelelor, dacă acestea sunt corecte, va crea o nouă sesiune în care va fi sălvat simbolul JWT pentru a verifică apelul către backend, e-mail, utilizator de notificare ID și rol și se va atasă în localStorage o nouă valoare pentru token-ul de acces.

 Pentru că backendul să verifice faptul că utilizatorul este autentificat cu succes în aplicație, se vor trimite headerele necesăre autorizării utilizatorului, folosindu-se serviciul de HttpInterceptor, ce implementează interfață cu același nume. în cadrul funcției “intercept”, prezența și în figura 52, se va atasă request-ului interceptat un nou header de autorizare ce va conține token-ul de logare. Acest header este verificat în backend, iar pe baza rezultatului, aplicația va permite finalizarea request-ului la endpoint-ul trimis.

Figura 52

Serviciul AuthGuard este responsabil de redirecționarea la componentă de “LoginComponent” în momentul în care utilizatorul nu este logat. Această clasă implementează funcția canActivate din interfață “CanActivate”, ilustrată în figura 53: , ce verifică token-ul sălvat în TokenService, iar dacă acesta nu este gol atunci se permite accesărea url-ului. Dacă rezultatul este opus, atunci se va redirecționa utilizatorul pe pagină de logare.

Această clasă este atasăta rutelor dorite din AppRoutingModule pentru a restricționa accesul persoanelor neautorizate. Sintaxa este următoarea: “{ path: 'home', component: HomeComponent, canActivate: [AuthGuard]}”.

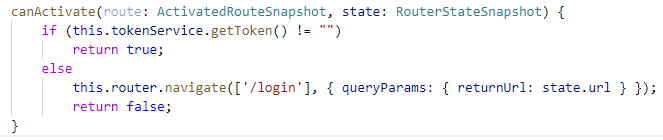


Figura 53

Un alt serviciu important din acest pachet este WebSocketService. Cu acest serviciu se stabilește conexiunea la url-ul de web socketing ce se conectează la API-ul create folosind atât SockJs cât și STOMP.js. Websocket este, de asemenea, un protocol de comunicare care permite implementarea de canale de comunicare bidirecționale între un server și client. Odată, conexiunea WebSocket este instanțiată între server și client, ambele pot face schimb de informații la nesfârșit până când conexiunea este închisă de oricine din cauza acestui WebSocket este preferat față de HTTP unde clientul și serverul schimbă informații la frecvență ridicată și cu latență scăzută deoarece conexiunea HTTP este închisă odată ce o cerere este servită de server și pentru a deschide din nou o conexiune HTTP, există o constrângere de timp. Websocket este bidirecțional, deoarece clienții se pot abona la un eveniment, iar serverele pot publică evenimente.

Pentru conectarea la API este folosită funcția „connect()”, ilustrată în figura 54, care va schimbă Subiectul comportamentului de la ATTEMPTING la CONNECTED.

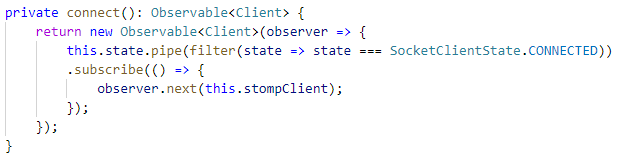


Figura 54

Folosind funcția “subscribeToNotifications()” se permite abonarea la mai multe endpoint-uri ale WebSocket-ului, folosind aceeași conexiune la API. Funcția de “disconnect()” este responsabilă de încheiarea conexiunii clientului Stomp cu backend-ul.

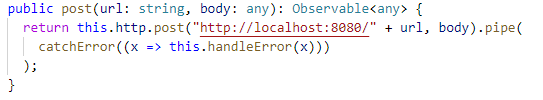
Tot în acest pachet se gasește și serviciul RestService, a cărui scop este a facilita trimiterea de request-uri la endpoint-uri din aplicația client. Astfel am creat funcții standard precum cea din figura 55, ce este apelată doar cu un șir de carcatere, ce reprezintă url-ul apelat, și corpul request-ului.

Figura 55

## Pachetul Pipe

Acest pachet conține clase ce implementează interfața PipeTransform. Aceste clase au o structură simplă, ele implementând doar funcția transform, și sunt responsabile cu transformarea și/sau formatarea datelor din template-urile HTML, acestea primind o valoare input și returnând o valoare output ce reprezinta valoarea primită transformată. Acest tip de clase sunt utile, deoarece se pot utiliza în întreaga aplicație, în timp ele sunt declarate o singură dată. Pentru a formata sursă imaginilor primite, ce reprezintă imaginea codificata în base64, dar care nu conține prefixul necesar decodificării, am folosit un pipe personalizat ce atașează sursei imaginii prefixul „data:image/png;base64,” și este folosit direct în fișierul HTML astfel: “profile.photo| image”.

# **Structura bazei de date**

Pentru a stoca un număr foarte mare de informație și pentru a asigura persistarea acesteia am optat pentru folosirea unei baze de date relaționale, și anume MySql. Un avantaj vital al utilizării unui astfel de sistem de gestionare a bazei de date este faptul că relațiile definite între entități sunt ușor de accesăt cu ajutorul acțiunilor de combinare prin chei străine a tabelelor. Această particularitate specifică bazelor de date relaționale are că rezultat rapiditate în căutare și accesărea ușoară a informației dorite, adăugarea, modificarea și ștergea datelor nu necesită mult cod de scris.

O baza de date normalizată reprezintă o baza de date căreia i s-au aplicat procese de normalizare, ce au rezultat în structurarea cât mai logică a tabelelor minimizarea anomaliilor la adăugare, ștergere și modificare și minimizarea datelor redundante. Baza de date creată este în faza a treia a normalizării deoarece îndeplinește primele două forme ale normalizării, a această conțînând numai valori atomice ce nu mai pot fi decompuse și sunt dependete de cheia primară, iar dependințele funcționale tranzitive au fost înlocuite. Pentru proiectul de față am ales crearea următoarelor tabele principale:

User – cuprinde informațiile necesăre la logare ale utilizatorului:

·       id - tip de date:integer; reprezintă cheia primară

·       nickname - tip de date:varchar(255); reprezintă porecla care va fi afisăta în aplicație pentru utilizator

·       username - tip de date:varchar(255); reprezintă username-ul

·       password - tip de date:varchar(255); reprezintă parolă încriptată

·       role\_id - tip de date:bigint; reprezintă cheia străînă spre tabela Role

Role – cuprinde informațiile necesăre la logare ale utilizatorului:

·       id - tip de date:integer; reprezintă cheia primară

·       name - tip de date:varchar(255); reprezintă denumirea rolului

Profile – cuprinde informațiile despre utilizator:

·       id - tip de date:integer; reprezintă cheia primară

·       display\_name: tip de date:varchar(255); reprezintă numele complet

·       description: tip de date:varchar(255); reprezintă descrierea profilului

·       photo: tip de date:longblob; reprezintă fotografia de profil

·       user\_id: tip de date:bigint; reprezintă cheia străînă spre tabela User

Chat- cuprinde informațiile despre o conversație creată între doi utilizatori:

·       id - tip de date:integer; reprezintă cheia primară

·       user1\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul care a inițiat pentru prima oară o conversație între cei doi utilizatori.

·       user2\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul cu care user1 a ales să discute.

Chat\_message- cuprinde informațiile despre un mesaj trimis în cadrul unui chat:

·       id - tip de date:integer; reprezintă cheia primară

·       sender\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul care creat un nou mesaj.

·       created\_on - tip de date:datetime; reprezintă momentul în care a fost creat mesajul.

·       message - tip de date:vachar(255); reprezintă conținutul mesajului.

·       state - tip de date:bit(1); reprezintă statusul mesajului trimis, această este setată inițial cu valoarea 0, adică false, iar în momentul deschiderii mesajului această valoare va deveni 1.

·       chat\_id: tip de date:bigint; cheia străînă spre tabela Chat și reprezintă conversația în cadrul căreia utilizatorul a trimis mesajul.

Comment- cuprinde informațiile despre un comentariu adăugat unei postări:

·       id - tip de date:integer; reprezintă cheia primară

·       user\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul care creat un nou comentariu.

·       created\_on - tip de date:datetime; reprezintă momentul în care a fost creat mesajul.

·       message - tip de date:vachar(255); reprezintă conținutul mesajului.

Like- cuprinde informațiile despre o nouă apreciere adăugată unei postări:

·       id - tip de date:integer; reprezintă cheia primară

·       user\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul care creat o nouă apreciere.

Tag- cuprinde informațiile despre o nouă eticheta adăugată unei postări:

·       id - tip de date:integer; reprezintă cheia primară

·       user\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul care este etichetat.

Follow - cuprinde informațiile despre o urmărire:

·       id - tip de date:integer; reprezintă cheia primară

·       followed\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul începe să fie urmărit.

·       follower\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul ce a ales să-l urmărească pe utilizatorul reprezentat de „followed\_id”.

Notification- cuprinde informațiile despre o notificare nou creată și destinată unui utilizator:

·       id - tip de date:integer; reprezintă cheia primară

·       created\_on - tip de date:datetime; reprezintă momentul în care a fost creată notificarea.

·       sender\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul a cărui acțiune a inițial creare unei noi insertiii în această tabela.

·       message - tip de date:vachar(255); reprezintă conținutul mesajului notificării.

·       state - tip de date:bit(1); reprezintă statusul notificării trimise, această este setată inițial cu valoarea 0, adică false, iar în momentul deschiderii notificării această valoare va deveni 1.

·       receiver\_id: tip de date:bigint; cheia străînă spre tabela User și reprezintă utilizatorul căruia îi este destinată notificarea.

Post – cuprinde informațiile despre o postare:

·       id - tip de date:integer; reprezintă cheia primară

·       created\_on - tip de date:datetime; reprezintă momentul în care a fost creată postarea.

·       description: tip de date:varchar(255); reprezintă descrierea postării

·       photo: tip de date:longblob; reprezintă fotografia adăugată

·       user\_id: tip de date:bigint; reprezintă cheia străînă spre tabela User

Story – cuprinde informațiile despre o nouă poveste adăugată:

·       id - tip de date:integer; reprezintă cheia primară

·       created\_on - tip de date:datetime; reprezintă momentul în care a fost creată povestea.

·       photo: tip de date:longblob; reprezintă fotografia adăugată.

·       user\_id: tip de date:bigint; reprezintă cheia străînă spre tabela User.

# Concluzii

Acest capitol cuprinde concluziile finale la care am ajuns în urma implementării platformei de social media propuse.

## Obiective propuse

Am dorit implementarea unei platforme sociale ce să conțină cel puțin funcționalitățile de bază astfel incât un utilizator să poată reuși să creze conexiuni cu persoanele căutate, iar acest lucru a fost făcut posibil. De asemenea s-au implementat și funcționalități mai ample ce îi permit utilizatorului să:

* Își poată crea cont
* Poată edita profilul personal
* Poată adauga, edita și șterge postări
* Poată adauga și șterge aprecieri
* Poată adauga și șterge comentarii
* Poată vizualiza postările persoanelor urmărite în ordine descendentă
* Trimită mesaje
* Fie notificat cand i se apreciază și/sau comentează o postare
* Fie notificat cand incepe să fie urmărit
* Fie adauge story-uri
* Fie fie notificat cand i se trimit mesaje
* Fie vizualize datele personale ale unui profil

## Idei de dezvoltare ulterioare

Platforma poate fi imbunătațită având în vedere că în momentul de fața conține funcționalități de bază, astfel pe viitor aceasta ar putea fi dezvoltată astfel încăt să conțină urmatoarele funcționalități:

* Adaugare de multiple poze pentru o postare
* Adaugarea de poze pentru story-uri
* Verificarea e-mailului
* Permiterea raportării postărilor pe baza criteriilor
* Creare funcționalității părții pentru admin
* Evaluarea postărilor
* Sugerarea conexiunilor pe baza de criterii mai complexe
* Încarcarea de videoclipuri

În concluzie platforma reușește să ofere utilizatorului, pe langa funcționalitățile de bază, posibilitatea de a comunica prin intermediul mesajelor cu cei dragi, de a comenta la postări și de a fi notificat cu privirea la diverse acțiuni. De asemenea vizează și dezvoltarea ulterioară a funcționalităților pentru îmbunătățirea experienței în cadrul acesteia.

# Bibliografie

* Alex Antonov, Spring Boot Cookbook: <https://books.google.ro/books?hl=ro&lr=&id=1dpOCwAAQBAJ&oi=fnd&pg=PP1&dq=spring+boot&ots=RpBS0Ckstk&sig=zomt8MlMW3_CbrUX_g6WAb4SKcw&redir_esc=y#v=onepage&q&f=false>
* Craig Walls, Spring Boot IN ACTION : <https://doc.lagout.org/programmation/Spring%20Boot%20in%20Action.pdf>
* Anton Moiseev, Angular Development with TypeScript : <https://books.google.ro/books?hl=ro&lr=&id=1TgzEAAAQBAJ&oi=fnd&pg=PT20&dq=angular+framework+typescript&ots=yfFWTbDJA-&sig=cW9B0O3OTTVAm_DIxLObMEo-cJA&redir_esc=y#v=onepage&q=angular%20framework%20typescript&f=false>
* R. Shaik, Spring Boot Security + JSON Web Token Tutorial:

<https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world>

* Steve Suehring, MySql Bible: <http://justpain.com/eBooks/Databases/MySQL/MySQL%20Bible.pdf>