UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI
DISCIPLINA BAZE DE DATE PROIECT

Gestiunea unei Platforme de Tip Forum

Coordonator,
Cătălin Mironeanu

Student,
Cîrjă Ioan
GRUPA 1308B

Iaşi, 2023

TITLU PROIECT: Gestiunea unei Platforme de Tip Forum

Proiectul se va concentra pe analizarea problemei administrării unei platforme online, unde utilizatorii pot crea postări, sau reacționa la postări ale altor utilizatori. Se urmărește proiectarea și implementarea unei baze de date care să permită modelarea fluxului de postări printr-o astfel de platformă.

Testare a fost realizată, de asemenea, print-o aplicație web ce foloseste Node.js, Express.js, MySql Database, HTML, CSS, Javascript.

DESCRIEREA CERINȚELOR ȘI MODUL DE ORGANIZARE AL PROIECTULUI:

DATELE DE INTERES PRINCIPALE

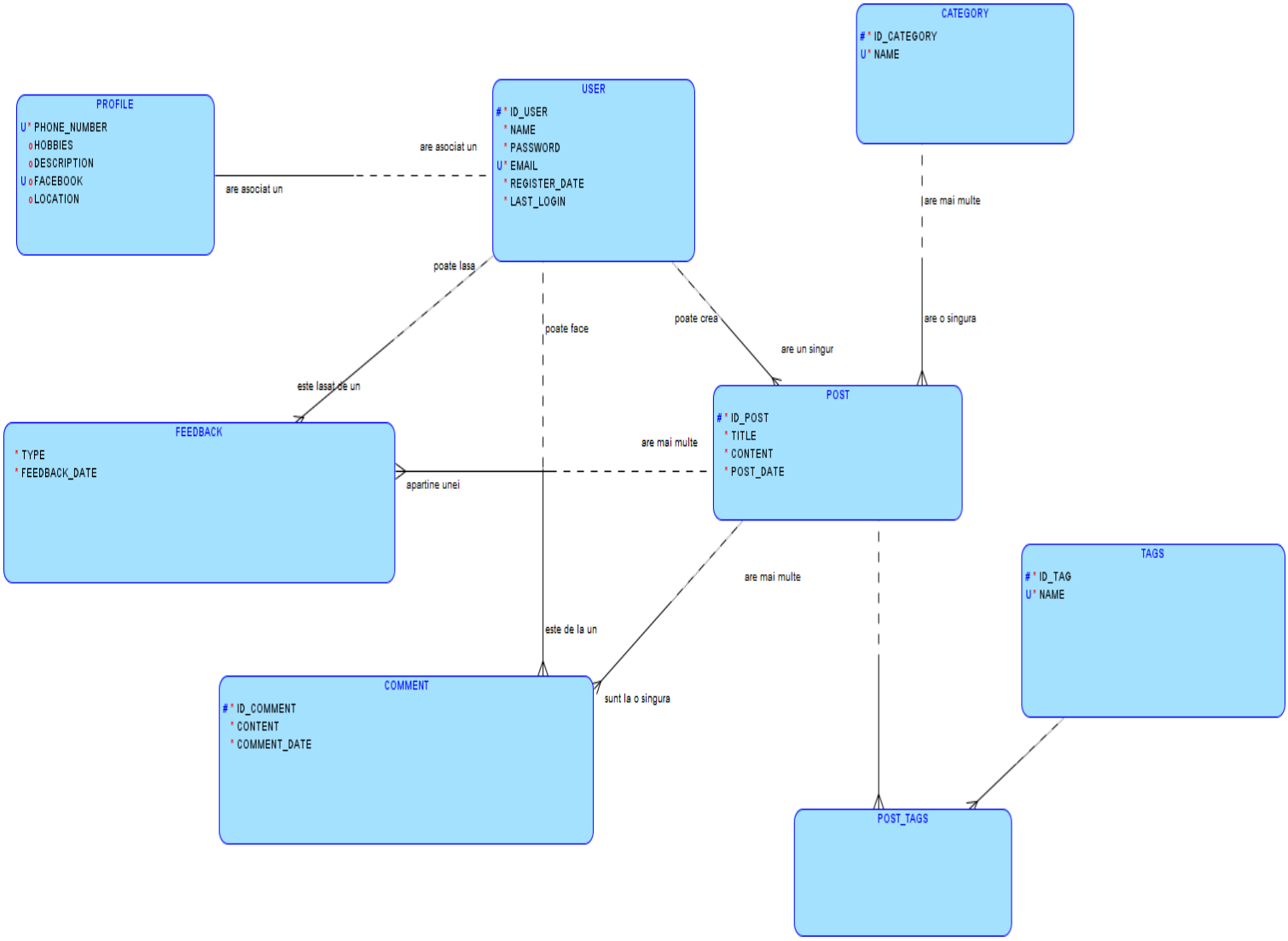
Acestea sunt gestionate de baza de date și sunt legate de:

- **USER**: Trebuie să păstrăm o listă a utilizatorilor înregistrați pe platformă pe baza e-mailului, a parolei contului, a numelui (stocate în tabela USER), În baza mai multor tabele, profilul unui utilizator este completat cu date personale, dar care nu au un caracter obligatoriu. Astfel, ei își pot personaliza platforma, cu descrieri, locații, date de contact pentru alți utilizatori.

- **POSTĂRI**: În mod inevitabil, trebuie să păstrăm date despre postarea efectivă. Monitorizarea acestora se va face înmai multe tabele, fiecare gestionând diferite aspecte, precum, aprecieri, comentarii, conținutul postării,etc. Utilizatorii vor fi capabili să își creeze propriile postări, cu conținut după bunul plac, aparținând diferitelor categorii valabile. Aceste postări pot primi comentarii, întrebări, sau opinii, dar și un simplu like/dislike. Pentru o transmitere a mesajului mai concretă, sau pentru o sortare ce nu depinde de categorie, utilizatorii pot să-și adauge tag-uri la postări, similar unei postări de pe facebook (ex. #hardwork).

DESCRIEREA DETALIATĂ A MODELULUI LOGIC ȘI A ENTITĂȚILOR

DIAGRAMA MODELULUI LOGIC:



ENTITĂȚI ȘI CONSTRÂNGERI FOLOSITE:

- **USER**- entitate folosită în gestionarea utilizatorilor
  - ID_USER - NUMBER(4)
    - Cheie primară
    - Nu poate fi NULL
  - NAME - VARCHAR2(50)
    - Nu poate fi NULL
    - Conține o verificare de lungime de nume
    - Nu conține o verificare de tip regexp, în ideea că utilizatori își pot denumi contul sub orice acronim, ce nu trebuie să fie numele real, asemeni platformelor de astăzi
  - PASSWORD - VARCHAR2(50)
    - Nu poate fi NULL
    - Trebuie să conțină și litere mari, și litere mici
    - Lungime minimă impusă
  - EMAIL - VARCHAR2(50)
    - Nu poate fi NULL
    - Validare a email-ului folosind regexp
    - Unic
  - REGISTER_DATE – DATE
    - Nu poate fi NULL
    - Nu pot fi puse valori peste data curentă
  - LAST_LOGIN – DATE
    - Nu poate fi NULL
    - Nu pot fi puse valori peste data curentă

- **PROFIL** – conține informații adiționale despre utilizator, la opțiunea acestuia
  - PHONE_NUMBER - VARCHAR2(10)
    - Nu poate fi NULL
    - Validat prin regexp
    - Unic
  - HOBBIES - VARCHAR2(100)
    - Nu conține validări în ideea personalizării
  - DESCRIPTION - VARCHAR2(100)
    - Nu conține validări în ideea personalizării
  - FACEBOOK - VARCHAR2(50)
    - Validare a linkului către adresa de facebook folosind regexp
    - Unic
  - LOCATION - VARCHAR2(80)
  - ID_USER
    - Cheie străină către tabela USER
    - Cheie primară

- **POST**- tabelă folosită pentru gestionarea postărilor
  - ID_POST - NUMBER(4)
    - Cheie primară
    - Nu poate fi NULL
  - TITLE - VARCHAR2(50)
    - Nu poate fi NULL

- Conține o verificare de lungime

- CONTENT - VARCHAR2(1000)
  - Nu poate fi NULL
  - Lungime minimă impusă

- POST_DATE – DATE
  - Nu poate fi NULL
  - Nu pot fi puse valori peste data curentă

- ID_USER
  - Cheie străină către tabela USER

- ID_CATEGORY
  - Cheie străină către tabela CATEGORY

➢ **CATEGORY**- tabelă folosită pentru gestionarea categoriilor postărilor
- ID_CATEGORY - NUMBER(4)
  - Cheie primară
  - Nu poate fi NULL

- NAME - VARCHAR2(100)
  - Nu poate fi NULL
  - Conține o verificare de lungime

➢ **TAGS**- tabelă folosită pentru gestionarea tag-urilor
- ID_TAG - NUMBER(4)
  - Cheie primară
  - Nu poate fi NULL

- NAME - VARCHAR2(50)
  - Nu poate fi NULL
  - Conține o verificare de lungime, împreună cu un regexp ce se asigură că tag-ul începe cu #

➢ **POST_TAGS**- tabelă folosită pentru legătura M-M dintre tag-uri și postări
- ID_TAG
  - Cheie străină către TAGS
- ID_POST
  - Cheie străină către POST

➢ **FEEDBACK**- tabelă folosită pentru gestionarea reacțiilor la postări
- TYPE – VARCHAR2(8)
  - Nu poate fi NULL
  - Poate fi doar pozitiv/negativ
- FEEDBACK_DATE – DATE
  - Nu poate fi NULL
  - Nu pot fi puse valori peste data curentă
- ID_USER
  - Cheie străină către USER
- ID_POST
  - Cheie străină către POST

➢ **COMMENT** - tabelă folosită pentru gestionarea comentariilor
- ID_COMMENT
  - Cheie primară
  - Nu poate fi NULL
- TYPE – VARCHAR2(8)
  - Nu poate fi NULL
  - Poate fi doar pozitiv/negativ

- CONTENT – VARCHAR2(300)
  - Nu poate fi NULL
  - Verificare de lungime

- COMMENT_DATE – DATE
  - Nu poate fi NULL
  - Nu pot fi puse valori peste data curentă

- ID_USER
  - Cheie străină către USER

- ID_POST
  - Cheie străină către POS

ÎN PROIECTAREA BAZEI DE DATE S-AU STABILIT URMĂTOARELE TIPURI DE RELAȚII:

➔ **1:1**

- USER-PROFIL, deoarece un profil este asociat unui singur utilizator, în mod logic.

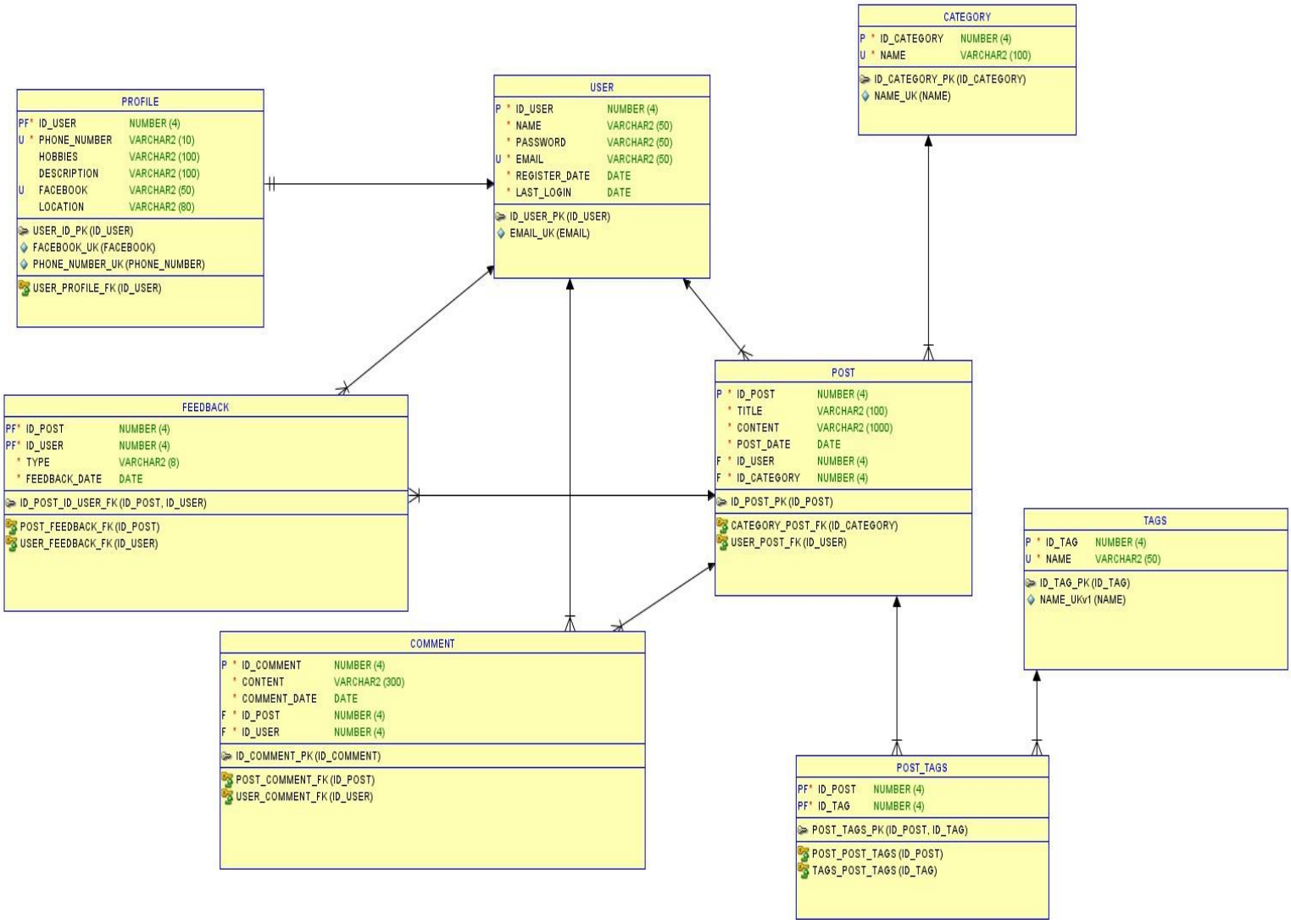
➔ **1:n**

- USER-POST, deoarece un user poate crea mai multe postări, dar o postare aparține unui singur user.
- USER-FEEDBACK, deoarece un user poate da mai multe like-uri/dislike-uri la postări diferite, dar un feedback aparține unui singur user
- USER-COMMENT, similar cu feedback, un user poate lasa mai multe commentarii, dar un comentariu aparține unui singur user
- CATEGORY-POST, deoarece o postare ține de o singă categorie, dar o categorie are mai multe postări
- POST-FEEDBACK, o postare are mai multe feedback-uri, dar un feedback ține de o singură postare
- POST-COMMENT, un comentariu ține de o singură postare, dare o postare poate avea mai multe comentariii.

➔ **N:N**

- POST-TAGS, deoarece o postare poate avea mai mule tag-uri, iar un tag apare la mai multe postări, acest lucru se realizează prin tabela POST_TAG

DESCRIEREA MODELULUI RELAȚIONAL:



ASPECTE LEGATE DE NORMALIZARE:

Normalizarea a fost folosită la nivelul entităților prin folosirea foreign key-urilor pentru a le descompune în entități mai mici care stochează aceleasi date ca și entitatea inițială astfel încât să fie eliminate redundanță în date.

ALTE FUNCȚIONALITĂȚI FOLOSITE:

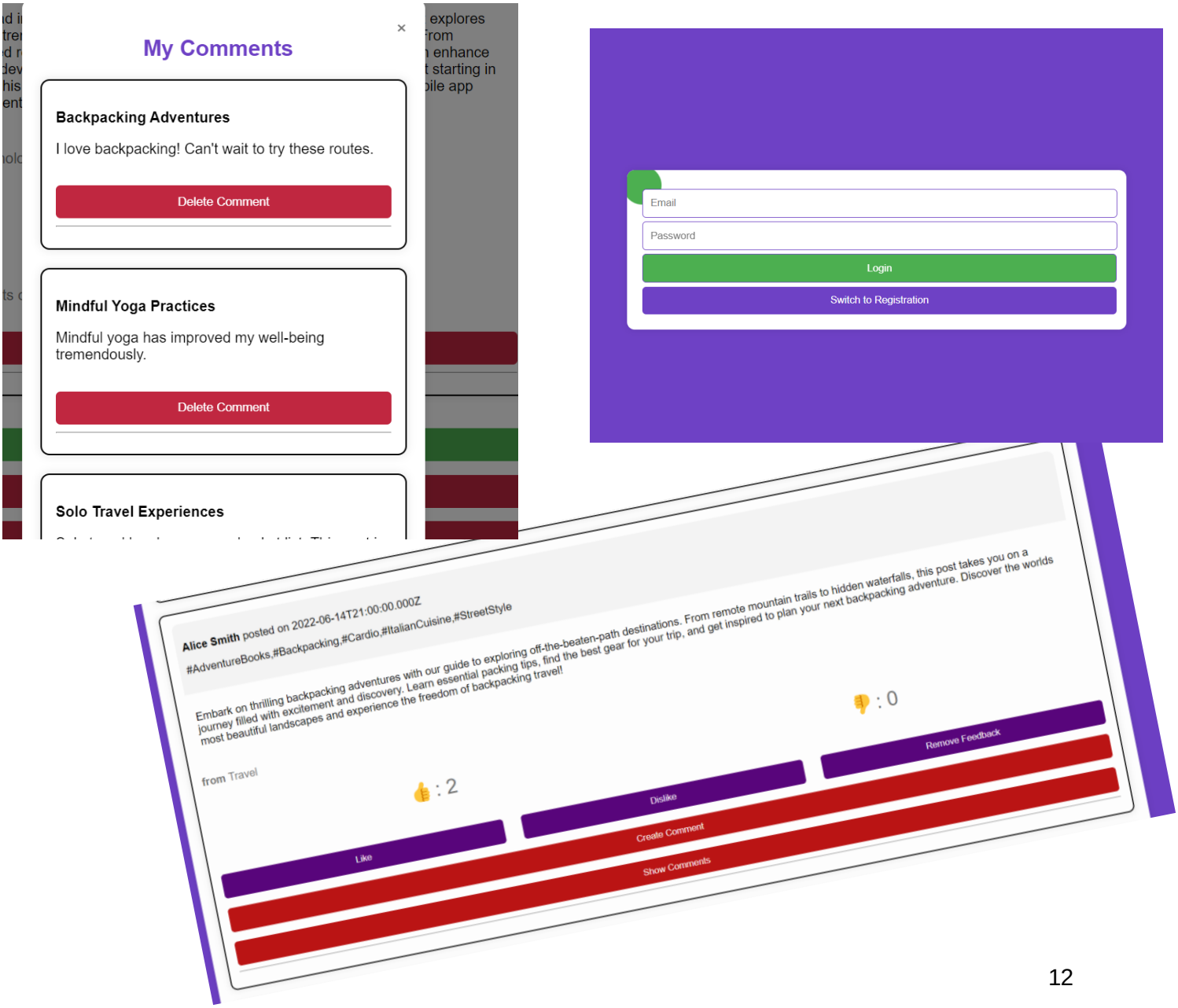
- AUTOINCREMENT, folosit pentru creșterea automată a id-urilor ce sunt chei primare, sau fac parte din tabela părinte într-o relație ce implică foreign key.

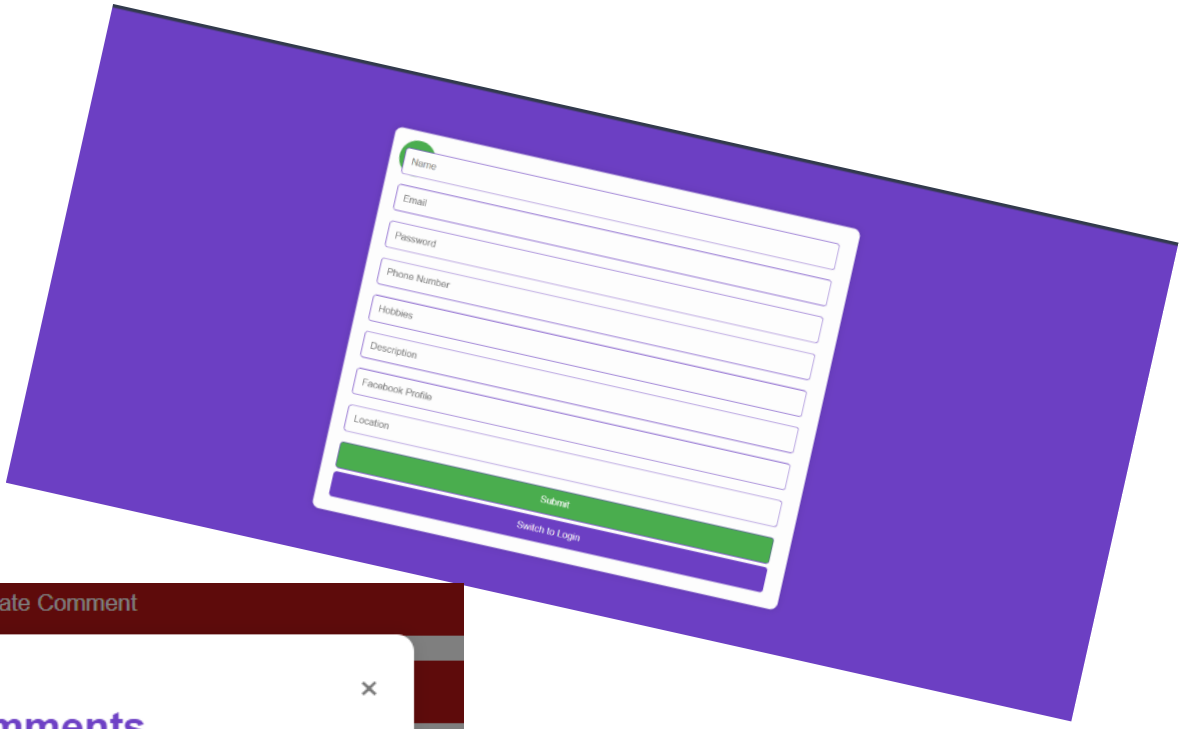
- TRIGGERS, folosite pentru verificarea datelor de tip DATE.

PROIECTAREA APLICAȚIEI WEB:

TEHNOLOGII FOLOSITE:

Frontend→ Javascript, HTML, CSS;
Backend→Node.js, Express.js, cu librării aferente;




Name
Email
Password
Phone Number
Hobbies
Description
Facebook Profile
Location
Submit
Switch to Login


Create Comment
Comments
I love backpacking! Can't wait to try these routes.
I have to try backpacking after reading this post!
ur nutrition with our collection of delicious and nutritious smoothie re

EXEMPLE DE COD:

→afisarea datelor unei postari din tabele MySQL

```
app.get('/posts', function (req, res) {
  var userIdToExclude = req.session.userId;
  console.log(userIdToExclude);

  var sql = `
    SELECT p.id_post, p.title, p.content, p.post_date,
    u.name as user_name, c.name as category_name,
    COUNT(DISTINCT f1.id_user) as positive_feedbacks,
    COUNT(DISTINCT f2.id_user) as negative_feedbacks,
    GROUP_CONCAT(DISTINCT t.name) as tags,
    MAX(cmt.content) as comment_content,
    MAX(cu.name) as comment_user
  FROM post p
  JOIN USER u ON p.id_user = u.id_user
  JOIN category c ON p.id_category = c.id_category
  LEFT JOIN feedback f1 ON p.id_post = f1.id_post AND f1.type = 'positive'
  LEFT JOIN feedback f2 ON p.id_post = f2.id_post AND f2.type = 'negative'
  LEFT JOIN post_tags pt ON p.id_post = pt.id_post
  LEFT JOIN tags t ON pt.id_tag = t.id_tag
  LEFT JOIN COMMENT cmt ON p.id_post = cmt.id_post
  LEFT JOIN USER cu ON cmt.id_user = cu.id_user
  WHERE p.id_user <> ?
  GROUP BY p.id_post, p.title, p.content, p.post_date, u.name, c.name
  ORDER BY p.post_date DESC;
  `;

  con.query(sql, [userIdToExclude], function (err, result) {
    if (err) {
      console.error('Error fetching posts:', err);
      res.status(500).json({ error: 'Internal Server Error' });
    } else {
      res.json(result);
    }
  });
```

→Conectarea propriu-zisă la baza de date a presupus creare unui server local MySQL, utilizand MySQL Workbench and MySQL Configurator;
→Aplicatia se conecteaza la utilizatorul root;
→Conexiunea se face folosind librariile express, mysql, nodemon, etc. din Node.js

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

var con = mysql.createConnection(
    {
        host:"localhost",
        user:"root",
        password:"Twins020406!",
        database:"forum",
    }
)
con.connect(function(err){
    if(err) throw err;
    console.log("We connected to DB!");
})
```