**Documentație- Flaskblog**

**Echipa:**

Tudose Bogdan

Cochintele Andreea

Bisoceanu Mihai

Ioniță Adriana

**Grupa**: 231

**Cuprins**

Capitolul I: Descrierea netehnică

* 1. Descriere comportament……..……………………………..………………………..pg 3
  2. Lista features…………………………………………………………………………pg 4
  3. User stories……………...……………………………………………………………pg 4
  4. Descriere aspect…………………………………………...…………………………pg 5

Capitolul II: Descrierea tehnică:

2.1 Diagrama uml referitoare la features și schema design……………………………pg

2.2 Descriere tehnică…………………………………………………………………..pg

2.2.1 Introducere- descriere pe larg……………………………………………….pg

2.2.2 Descrierea funcțiilor/fișierelor/claselor…………………..…………………pg

2.2.3 Testare……………………………………………………………………….pg

**Cap. I: Descrierea netehnica**

* 1. **Descrierea comportamentului:**

Blog-ul nostru este unul cu o interfță simplă și accesibilă pentru oricine. Pentru a putea scrie articole, utilizatorul trebuie să se logheze, dacă are deja un cont creat sau să se înregistreze, în cazul contrar. După logare, ne apare în bara de meniu butonul de “new post”, care ne redirecționează către formularul pentru scrierea unei postări noi, unde trebuie să completăm titlul articolului și conținutul. De asemenea, mai târziu, putem edita sau șterge postarea, doar apăsând click pe ea și pe butonul corespunzător acțiunii noastre. Pentru a vedea toate articolele pe care le-am scris sau pe care le-au scris alți utilizatori, este suficient să apăsăm pe numele utilizatorului și vom fi redirecționați pe o pagină care conține, în ordine cronologică toate postările scrise. În partea dreapta, avem un calendar în care putem adaugă evenimente sau putem edita/șterge un eveniment deja existent.

În bara de meniu, mai găsim și butonul “home”, care ne redirecționează către pagină principala, dar și butonul “about” care ne duce către pagina în care am scris câteva cuvinte despre conceptul blog-ului.

**1.2 Lista de features:**

-Pagina de home ( unde poți vedea postările userilor)

- Secțiune de postare articol ( adăugarea articolului propriu)

-Opțiunea de update ( modificarea informațiilor din articolul propriu)

-formular înregistrare ( creerea unui cont)

-login/logout( intrarea/ieșirea din cont)

-actualizare informații cont

- calendar evenimente( adăugarea, ștergerea și editarea unui eveniment)

* 1. **User stories:**

În calitate de utilizator al unui blog, îmi doresc ca acesta să aibă un design minimalist, în culori calde pentru a nu obosi privirea atunci când citesc sau scriu articole mai consistente și să fie ușor de navigat și folosit.

Epic1:

Ca utilizator vreau să folosesc un blog cu o interfață accesibilă

Story1 : Vreau să am un meniu ușor de accesat și la îndemână

Strory2: Vreau să pot găsi rapid postările

Epic2:

În calitate de utilizator al unui blog, doresc ca procesul de a postare al unui articol si cel de logare/înregistrare să fie unul cât mai simplu.

Story1: Vreau să pot posta în cât mai puțini pași

Story2: Doresc să mă pot înregistra/loga rapid și ușor (să nu fiu nevoit să trec prin mulți pași)

Epic3:

Îmi doresc să pot însemna deadline-urile pentru articolul pe care îl voi scrie undeva accesibil.

Story1: Vreau văd evenimentele marcate în mod constant

Strory2: Vreau să pot edita evenimentele adăugate, în cazul în care intervin modificări

* 1. **Descriere design:**

Pentru interfața blog-ului nostru am ales să folosim un design minimalist, cu o paletă de culori calde, pentru a nu obosi vederea utilizatorului. În cadrul blog-ului veți observa:

-Meniul (butoane pentru: redirecționare pe pagina de home/pagina contului/ pagina de about, crearea unei postări noi, logout ), poziționat în partea de sus a ecranului, pentru a ne putea deplasa pe pagină de care avem nevoie rapid.

- Titlul blog-ului “Flaskblog”, afișat pe bara de meniu, în partea stânga

-Calendar pentru evenimente: apare în partea dreaptă a ecranului, tot timpul, indiferent pe ce pagină suntem, ca utilizatorul să aibă constant evenimentele marcate în aria sa vizuală (fiindu-i astfel mai ușor să editeze informațiile legate de eveniment)

-Aranjate în textbox- uri, pe pagina de home, avem articolele postate de alți utilizatori sau chiar de noi înșine, ordonate în funcție de data postării

-În pagina de “new post”, avem două textbox-uri în care introducem titlul, respectiv conținutul articolului. De asemenea, tot aici avem butonul “post”, pe care îl folosim pentru a ne publica articolul.

-Dacă intrăm pe pagina contului nostru (“account”), în partea de sus sunt afișate informațiile contului și poza de profil

-Tot în pagina contului, ne putem edita informațiile (username-ul și e-mailul) și putem încărca o nouă poză de profil, cu ajutorul butonului “Choose file”

-În pagina de “about” găsim câteva informații despre crearea acestui blog și o imagine ce ne ilustrează bucuria de a scrie.

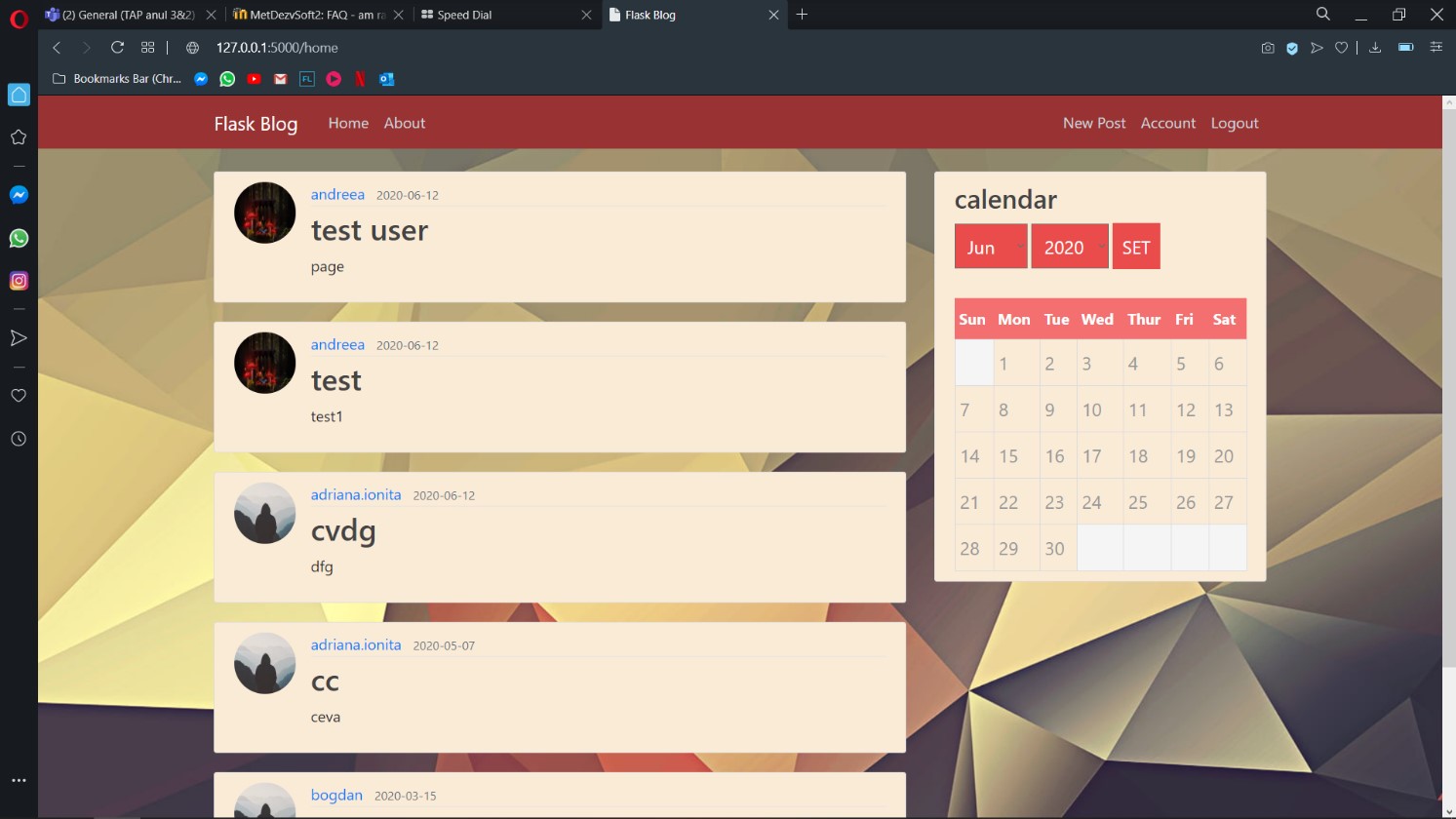


Figure 1.Home

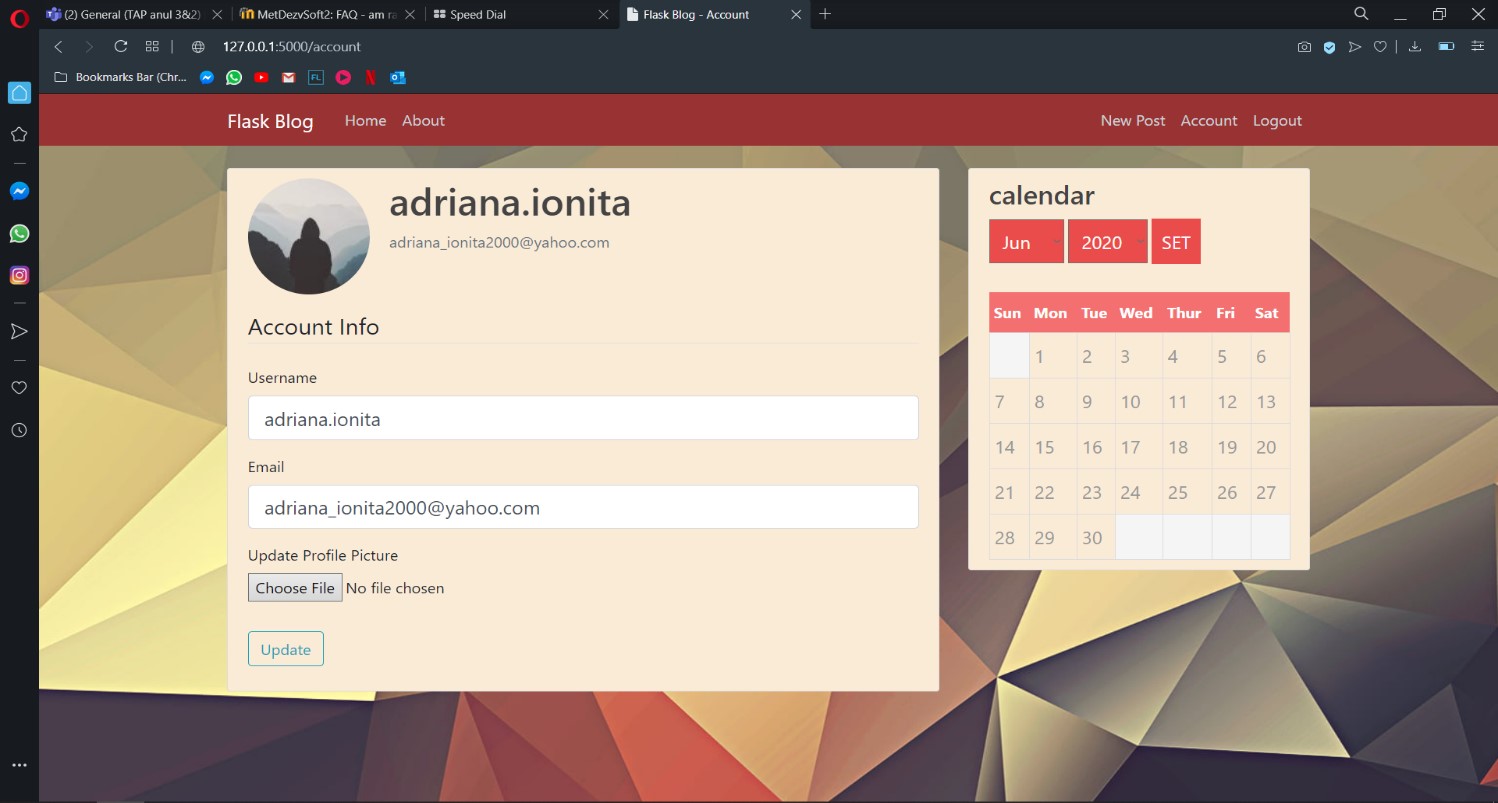


Figure 2.Account page

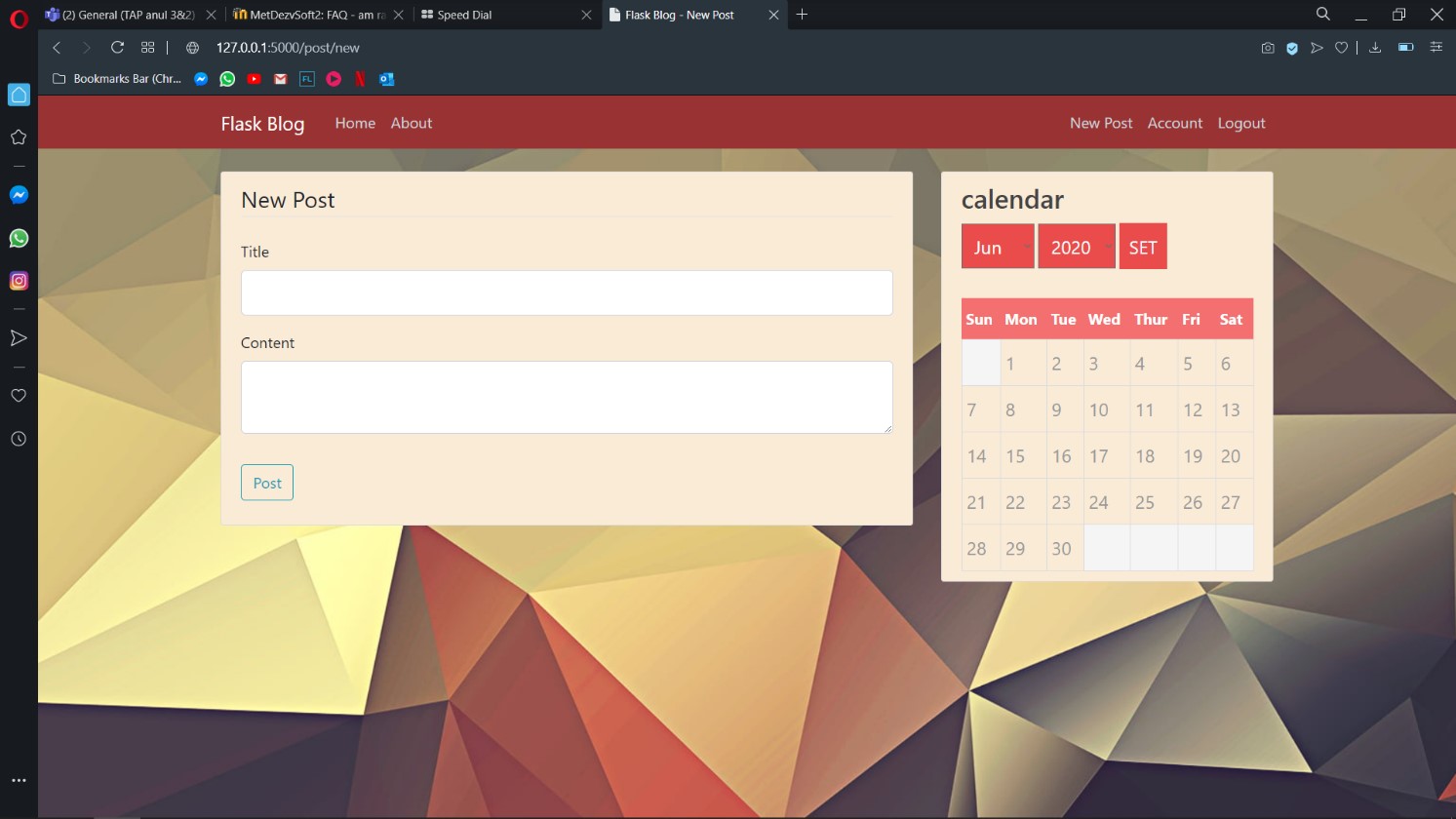


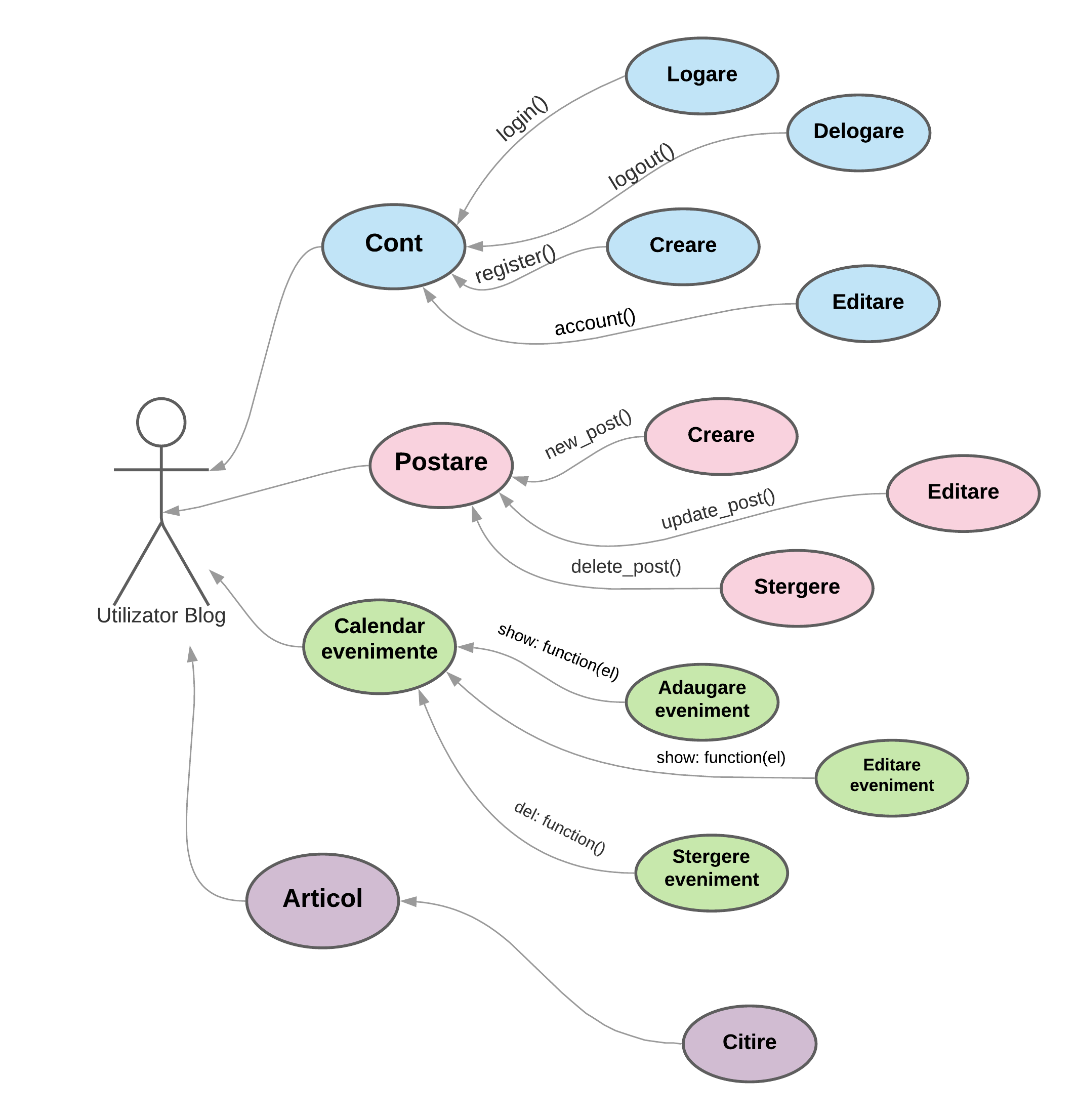
Figure 4.About

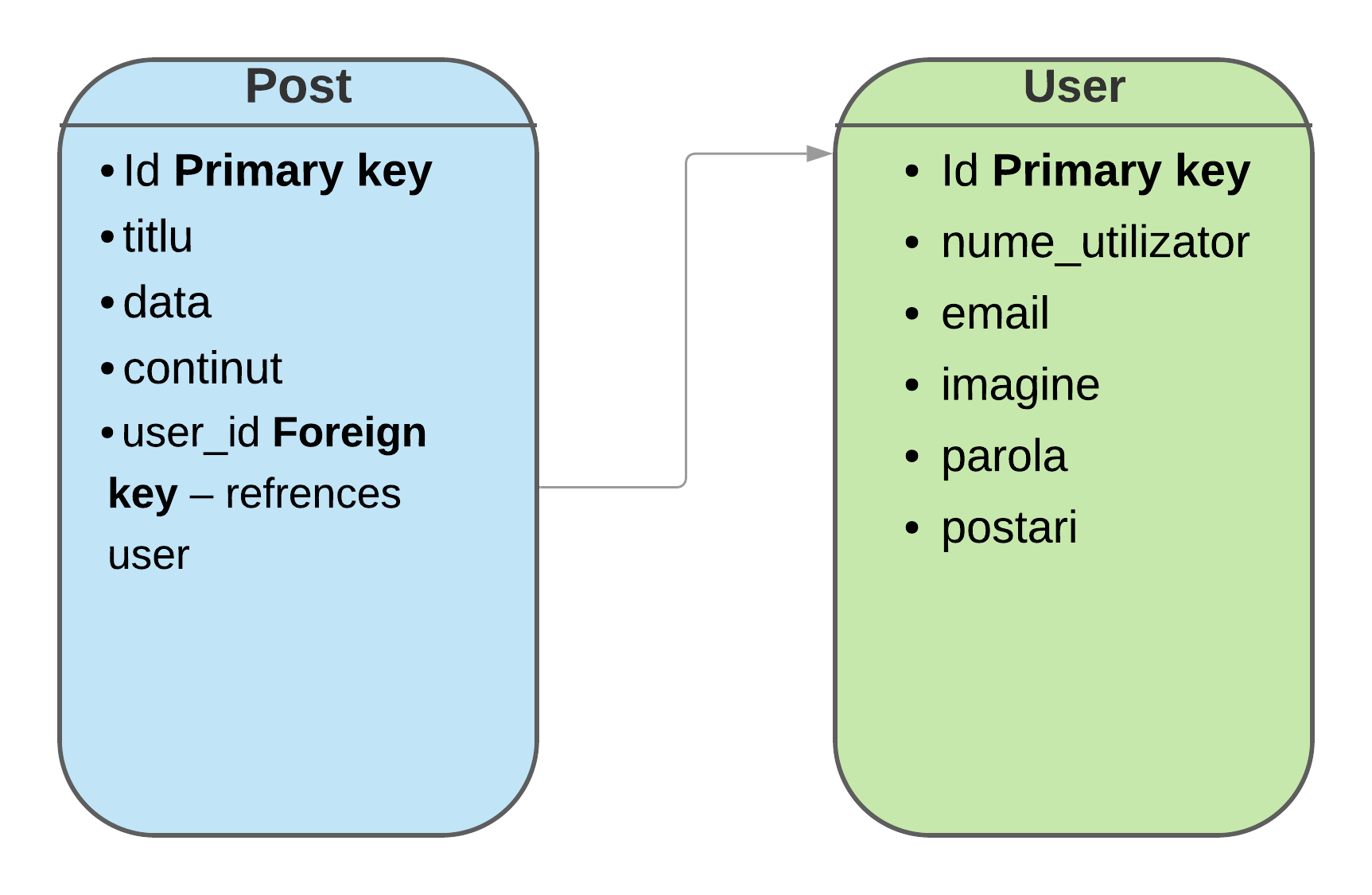
Figure 3.About

**Cap. II: Descrierea tehnica**

**2.1 Diagrama UML si schema de design:**

In diagrama de mai jos am ilustrat funcționalitatea blogului, din perspectiva utilizatorului și cu funcțiile care se apelează pentru fiecare acțiune:



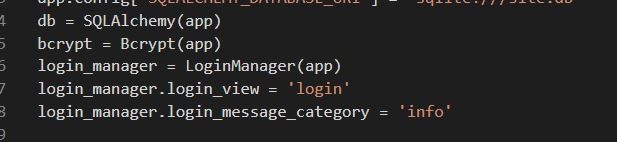
 Pentru schema de design, am creat tabelul bazei de date și am ilustrat cum postările și utilizatorul sunt în strânsa legatură, prin intermediul câmpului “user\_id”:

**2.2 Descriere tehnica:**

**2.2.1 Introducere:**

În crearea proiectului am folosit o multitudine de funcții pentru a implementa fiecare feature în parte, ajungând ca proiectul să conțînă șase fișiere python, urmând ca mai jos să explicăm ce se întâmplă exact in cadrul fiecărei funcții, dar și protocoalele și funcțiile folosite pentru testare. **2.2.2 Descrierea functiilor/fisierelor:**

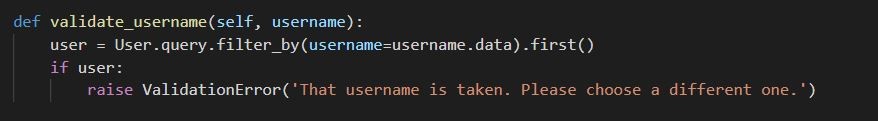
Primul fișier python numit “run.py” după cum o spune și numele are rolul de a porni aplicația.

 Cel de-al doilea, “\_init\_.py” se ocupă de partea de început ale aplicației, inițializare, crearea bazei de date și atribuirea “rolurilor” unor variabile, esențiale pentru această aplicație:

Astfel, prima variabilă, db, se ocupă cu reținerea informațiilor din baza de date, bcrypt este folosită pentru criptarea parolei introduse de utilizator, iar cele trei care au rămas se ocupă de procesul de logare al utilizatorului.

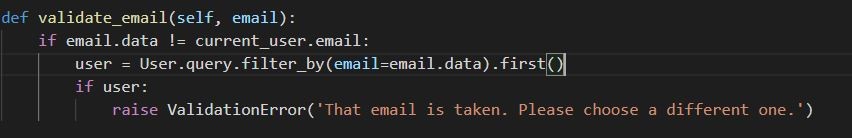
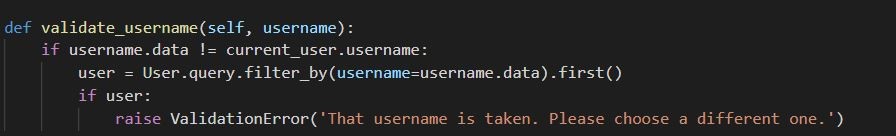
Fișierul python forms.py este alcătuit din mai multe clase ce reprezintă formularele de înregistrare, login etc. Astfel avem următoarele funcții și clase:

- “RegistrationForm”,este clasa în care se generează formularul de înregistrare pe aplicație. Înregistrarea constă în completarea a trei câmpuri: username, email și parolă. Aceasta conține două funcții distincte ce validează următoarele informații:

1. Dacă username-ul pe care dorim să-l folosim la înregistrare este deja utilizat sau nu, iar în cazul în care username-ul este deja folosit, pe ecran ne apare mesajul: “That username is taken. Please choose a different one.”

2. În mod similar, verificăm dacă email-ul pe care dorim să îl folosim la înregistrare este deja asociat altui cont:

- În clasa “UpdateAccountForm” găsim formularul de editare al informațiilor contului, pe scurt, “actualizarea” constă în completarea campurilor cu noul username și/sau email. Conține două funcții de validare a acestor informații, create in mod similar cu cele din “RegistrationForm”:



- “LoginForm” este clasa ce conține formularul de login al aplicației. Prin intermediul acesteia utilizatorul se conectează la propriul cont, completând câmpurile pentru email și parolă. Aceasta nu conține nicio funcție la care să putem face referire, cum am făcut anterior.

- Clasa “PostForm”, conține formularul pentru crearea unei noi postări. Utilizatorul poate posta pe “peretele” paginii sale un nou articol, prin accesarea zonei de postare nouă, unde va avea de completat câmpurile pentru titlul postării, respectiv pentru conținutul acesteia. Din nou nu avem funcții la care să facem referire.

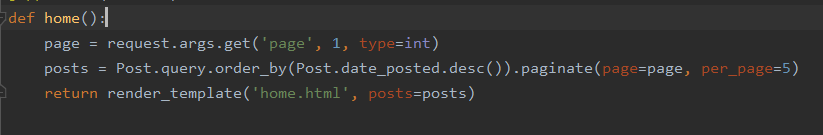
Toate formularele din fișierul forms.py, prezentate mai sus, conțin la final un buton de submit, dar fiind un lucru comun pentru toate, am ales să menționez despre asta la final pentru a nu oferi informații repetitive.

Fișierul Models.py conține două clase, ce construiesc “template-ul” pentru contul utilizatorului și pentru postare. Astfel avem:

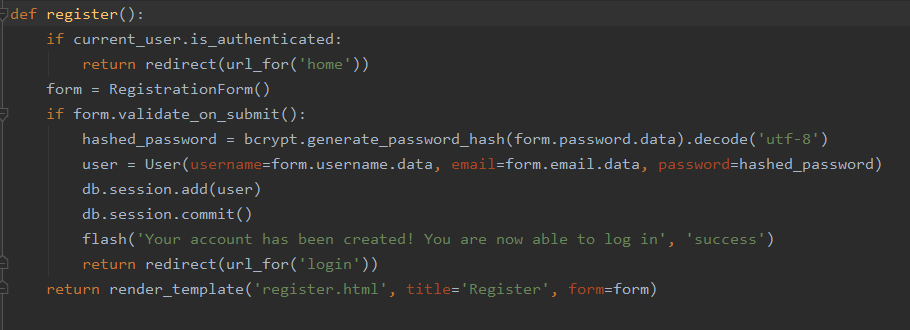
- “User” conține câmpurile de completare a datelor contului. Utilizatorul are și opțiunea de a avea o poză la profilul său, iar dacă acest câmp nu este completat, atunci poza va fi una ce a fost anterior selectată de creeatorii aplicației ca default.

- “Post” conține câmpurile de completare a informațiilor necesare creării unei postări.

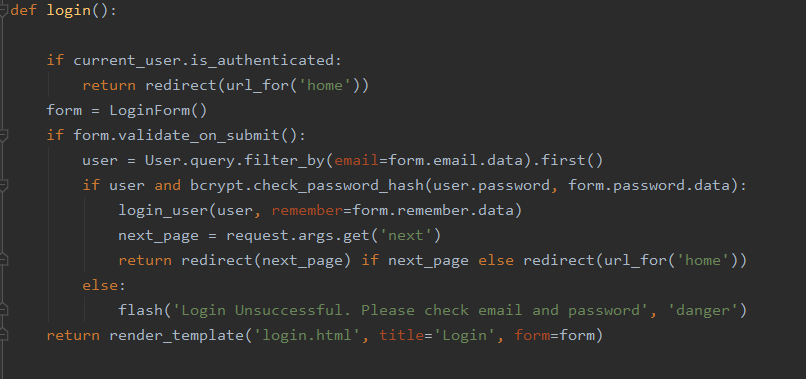
În fișierul “routes.py” găsim o multitudine de funcții ce asigura funcționarea corectă a aplicației:

****1. “home()” este functia ce returneaza template-ul paginii de home a aplicatiei. Cand utilizatorul doreste sa ajunga inapoi pe pagina de home, spatele programului se apeleaza aceasta functie ce il redirectioneaza pe pagina home.

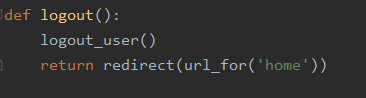
2.Funcția “about()” redirectioneaza utilizatorul pe pagina de about, funcționând în mod similar cu funcția “home()”.

3. “register()” este funcția ce se ocupă cu înregistrarea unui cont nou. Folosind câteva funcții deja implementate în librăria “Flask”, se realizează înregistrarea contului. Librăria “flask” are un rol important pentru crearea proiectului nostru, fiind importată în fiecare fișier python pe care îl avem. Aceasta este des întâlnită în construirea unei aplicații web, deoarece conține uneltele necesare pentru a putea face posibile logarea, înregistrarea și alte acțiunii specifice acestui tip de aplicație. De asemenea, în interiorul acestei funcții se realizează criptarea parolei, prin funcția “Bcrypt” din librăria menționată anterior. Bineînțeles, verificăm dacă informațiile introduse la înregistrare aparțin deja unui alt cont. Practic, atunci când se creează un cont nou, este apelată această funcție, returnându-se template-ul paginii de login dacă informațiile introduse sunt corecte, iar în caz contrar se va reveni la pagina de înregistrare.

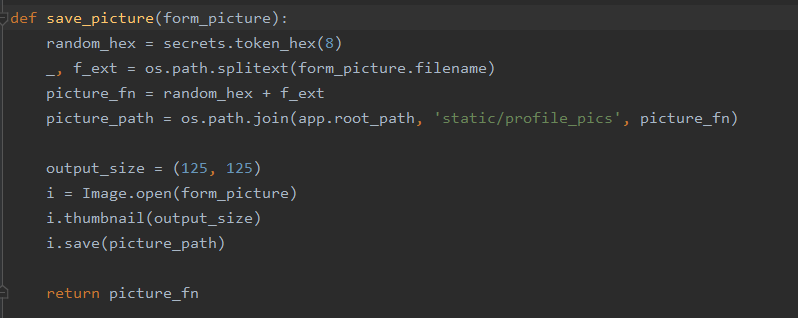
4. “login()” , după cum sugerează și numele este funcția ce se ocupă de login-ul utilizatorului în aplicație. Funcția se folosește de formularul de login din forms.py, iar când utilizatorul introduce parola aceasta este comparată cu cea criptată din baza de date, fiind decriptată pentru comparare cu ajutorul funcției ”generate\_password\_hash”. Dacă utilizatorul completează corect formularul, se afișează un mesaj de succes,iar utilizatorul este trimis la pagina de home, în acest moment fiind conectat.



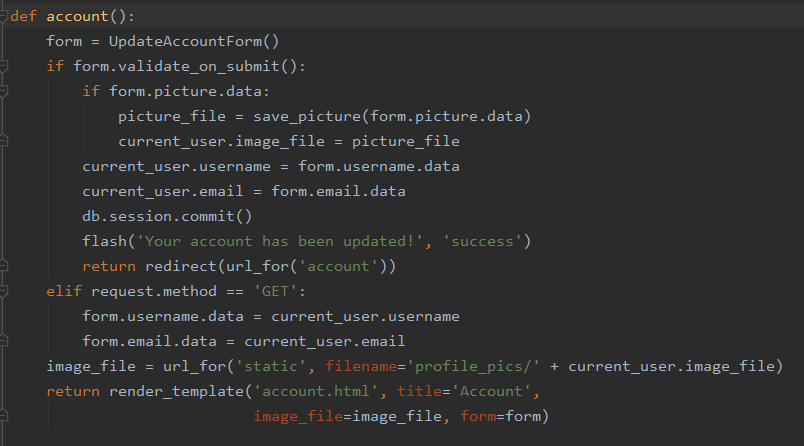
5. “logout()” este funcția ce realizează delogarea contului personal al utilizatorului. Folosește funcția “logout\_user()” din flask\_login pentru realizarea acestei acțiuni și ne redirectioneaza către pagina de home a aplicației, după deconectare.



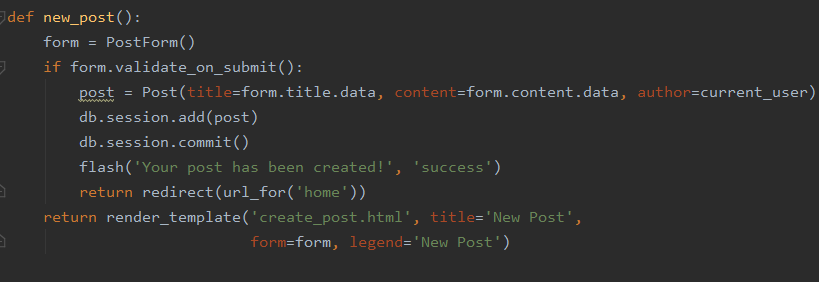
6. “save\_picture()” setează fotografia de profil a contului utilizatorului. Aceasta primește un nume randomizat și este salvata ca 125/125. În momentul în care utilizatorul dorește să își modifice poza de profil, această funcție este utilizată pentru a păstra poza dorită.



7. “account()” este funcția apelată atunci când modificăm informațiile contului. Aceasta are ca punct de start formularul de modificare a contului, explicat mai sus. După completare se va returna template-ul ce reprezintă pagina contului, dar cu informațiile actualizate.



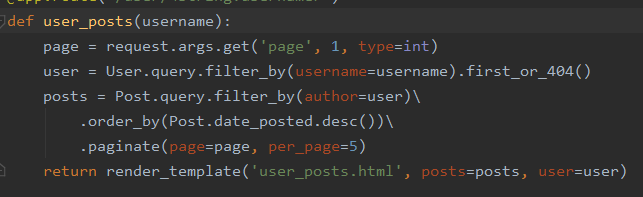
8. “new\_post()”, “post()” sunt funcțiile prin intermediul cărora utilizatorul creează o noua postare, respectiv este redirecționat la pagina post.html unde se pot face noi postări.



9. Funcțiile “update\_post()”, “delete\_post()” realizează modificări asupra unei postări/șterge o postare a utilizatorului, iar “db.session.commit(.delete)” actualizează baza de date cu noile informații.

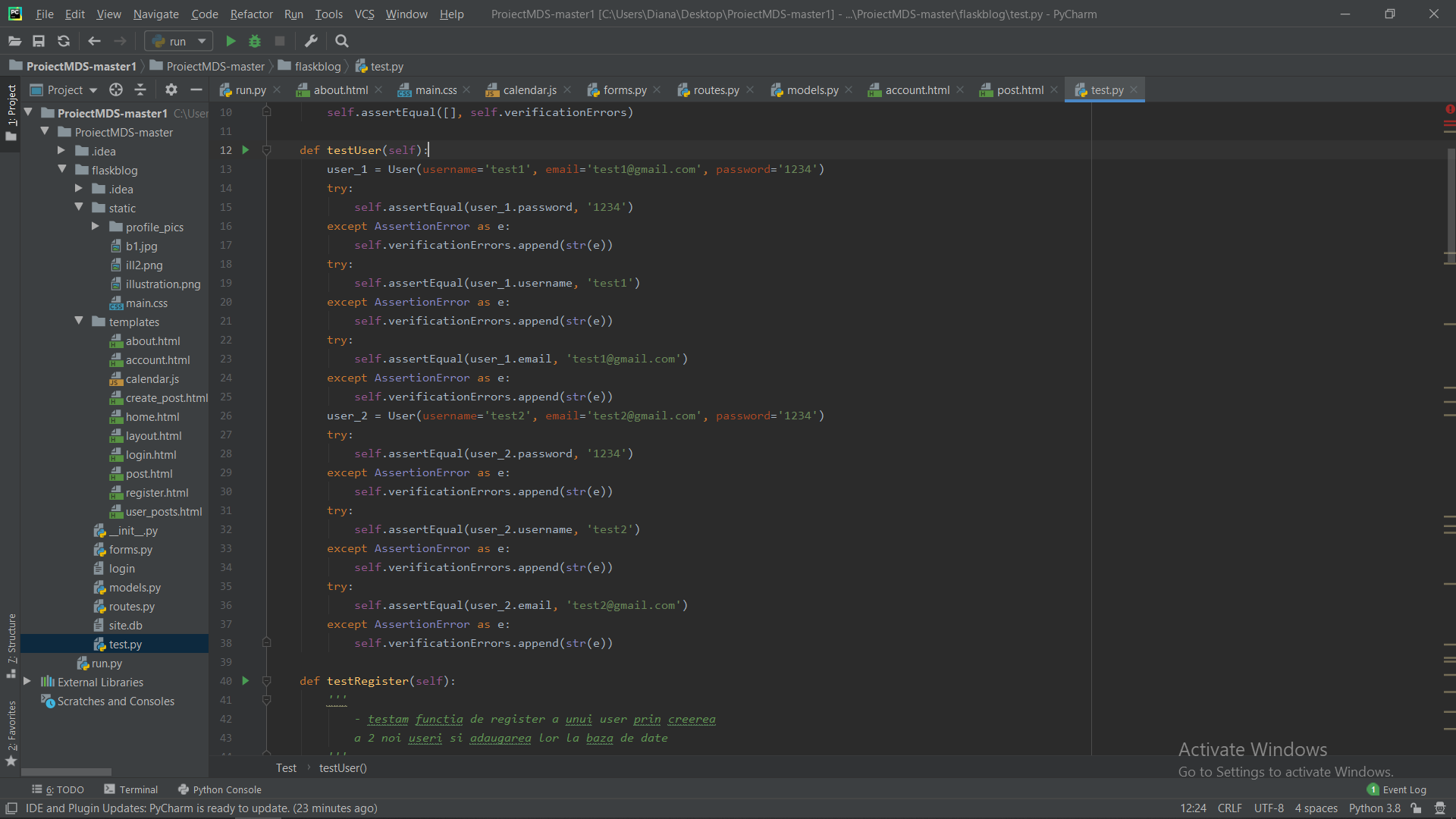


10. “user\_post” face posibilă vizualizarea postărilor unui utilizator, returnând pagina “user\_post.html” , unde vedem articolele scrise de utilizator în ordine cronologica.

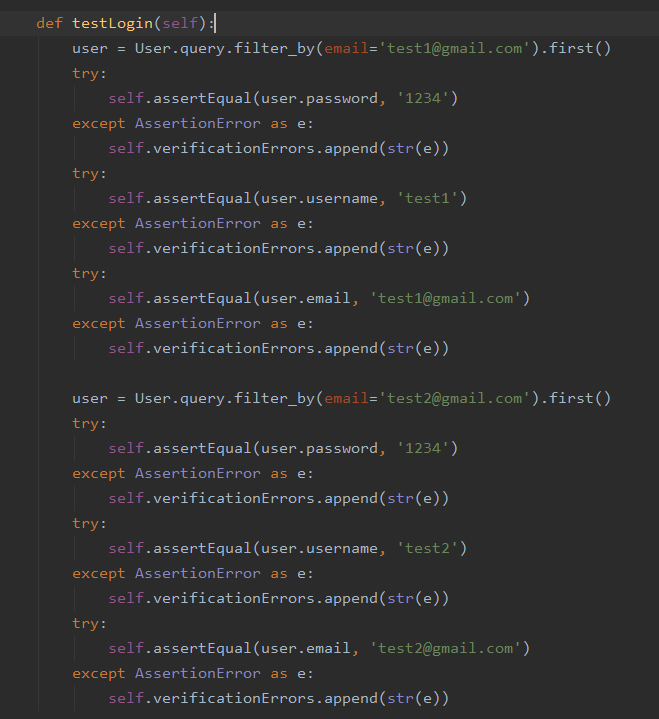


**2.2.3 Testare:**

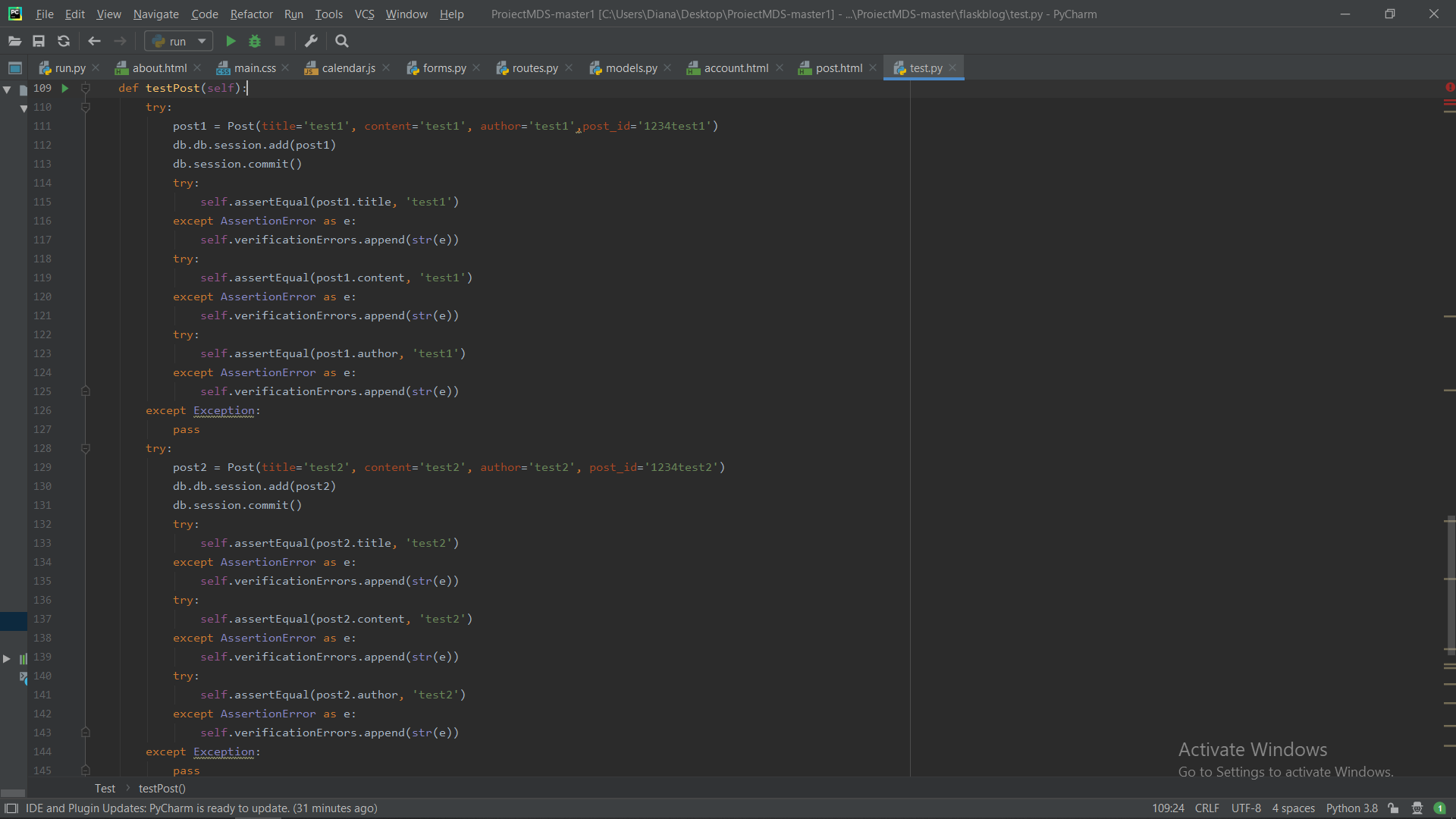
În principal, în cadrul proiectului am creat câteva funcții care testează actualizarea bazei de date, dar și a unor clase, după cum urmează:

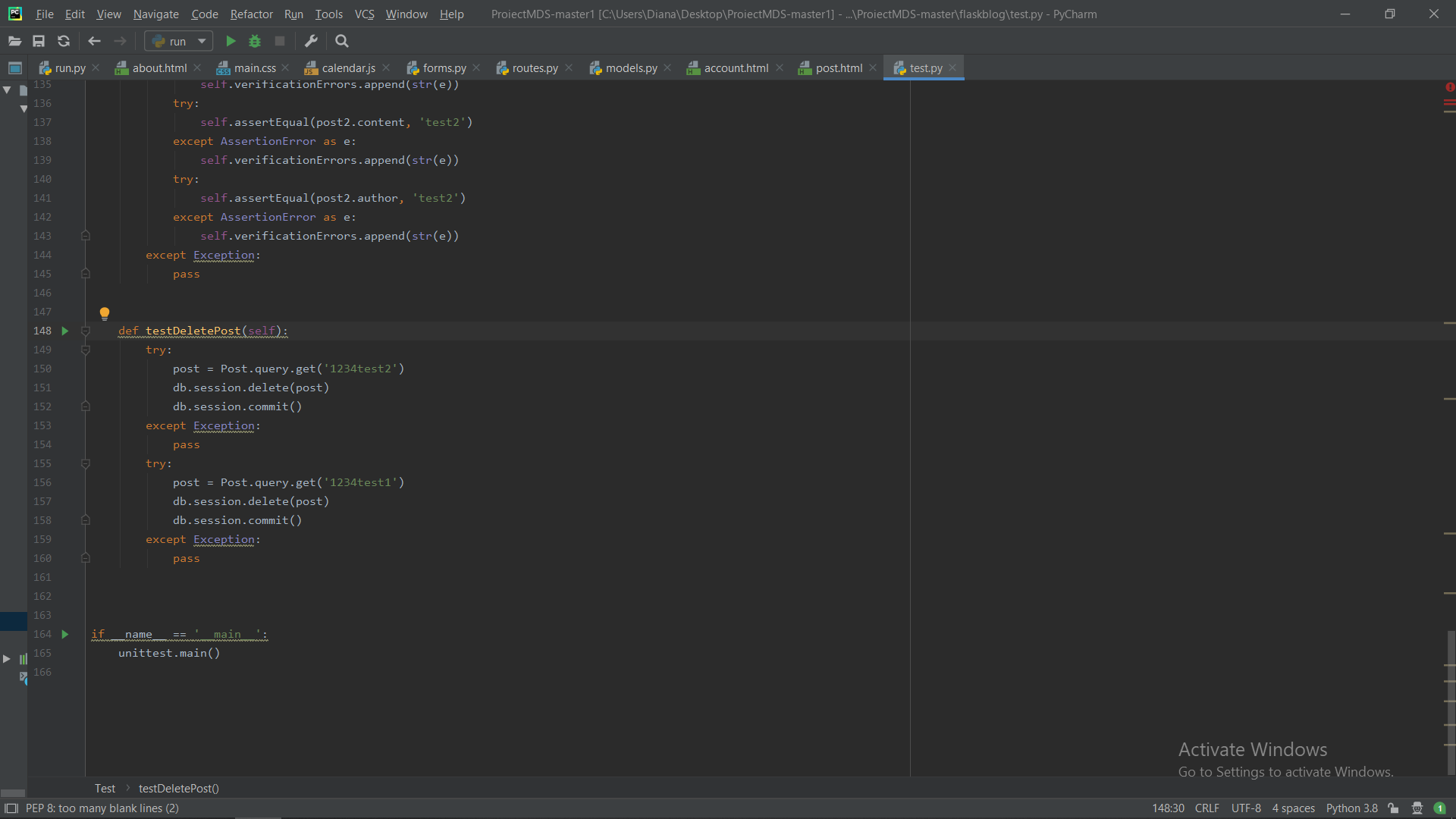
1. “testUser()” testăm clasa “user”, verificând dacă obiectele clasei se creează, prin adăugarea a doi utilizatori noi.
2. “testRegister()” testează funcția de register, verificând dacă noii utilizatori sunt adăugați în baza de date, prin crearea a 2 useri noi.



1. “testLogin()” este testată funcția de login a unui user, folosind utilizatorii adăugați anterior, la testarea înregistrării.

4. “testPost()” testează dacă se poate face o nouă postare, prin crearea altor două postări și adăugarea lor în baza de date.



1.  “testDeletePost()” se ocupă cu testarea funcției de ștergere a unei postări, prin indentificarea postărilor ce au fost create anterior când se testa adăugarea unui articol și ștergerea lor din baza de date.