|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE ȘTIIȚE  DEPARTAMENTUL DE INFORMATICĂ |  |

LUCRARE DE DIPLOMĂ

Stoentel Alexandru-Eduard

COORDONATOR ȘTIINȚIFIC

Asist. Dr. Tudorache Cristina

Iulie 2024

CRAIOVA

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE ȘTIIȚE  DEPARTAMENTUL DE INFORMATICĂ |  |

Platformă de blogging realizată cu tehnologii web şi baze de date

Stoentel Alexandru-Eduard

COORDONATOR ȘTIINȚIFIC

Asist. Dr. Tudorache Cristina

Iulie 2024

CRAIOVA

*„Nu există vânt favorabil decât pentru cei*

*care știu în ce direcție se îndreaptă”*

**Seneca**

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul Stoentel Alexandru-Eduard, student la specializarea Informatică din cadrul Facultății de Științe a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul [Platformă de blogging realizată cu tehnologii web şi baze de date](https://cis01.central.ucv.ro/evstud/diplome/detaliu_tema.php?id=57205),
* coordonată de Asist. Dr. Tudorache Cristina,
* prezentată în sesiunea IULIE 2024.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
* redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
* prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfârşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
* precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, etc.,
* precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Științe  Departamentul de Informatică | Aprobat la data de  …………………  Şef de departament,  Lect. Dr. Stoian Gabriel |

**LUCRARE DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului/-ei: | Stoentel Alexandru-Eduard |
| Enunțul temei: | Platformă de blogging realizată cu tehnologii web şi baze de date |
| Datele de pornire: | Pentru dezvoltarea proiectului de licentă am avut ca punct de plecare:   * modele și informații din diversele aplicații web de blogging. * Secolul în care majoritatea populației caută informații și doreste să fie la curent cu știrile de actualitate |
| Conținutul proiectului: | Structura proiectului va cuprinde următoarele secțiuni: introducere, tehnologii și framework-uri folosite, componente software utilizate, specificații și reprezentarea aplicației, dezvoltarea aplicației, implementarea aplicației și concluzii. |
| Material grafic obligatoriu: | Diagrame, scheme, capturi. |
| Consultații: | Periodice |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Asist. Dr. Tudorache Cristina |
| Data eliberării temei: | 1.12.2023 |
| Termenul estimat de predare a proiectului: | 20.06.2024 |
| Data predării proiectului de către student și semnătura acestuia: |  |

***Cuvinte cheie***: blog, știri, informație, Blazor, .NET, SQL Server.

**Recunoștiință**

În primul rând, doresc să îmi exprim recunoștința și mulțumirile întregului corp profesoral al acestei facultăți. De-a lungul acestor patru ani, prin implicarea și devotamentul lor, prin temele provocatoare și examenele semestriale, m-au ajutat să devin persoana harnică pe care am aspirat întotdeauna să o fiu.

De asemenea, doresc să mulțumesc coordonatorului meu de lucrare, doamnei Tudorache Cristina, căruia îi sunt profund recunoscător pentru sprijinul acordat. A fost mereu alături de mine, oferindu-mi sfaturi, idei și perspective noi. Domnul profesor a oferit întotdeauna feedback pozitiv și a fost întotdeauna disponibil pentru a răspunde întrebărilor mele. Vă sunt recunoscător și vă mulțumesc pentru tot sprijinul acordat!

Dar și profesorilor care mi-au fost mereu alături și nu au ezitat să mă susțină și să mă învețe tot ce știau ei și au avut o contribuire semnificativă în formarea mea profesională și personală. Fiecare dintre acești profesori mi-a oferit nu doar cunoștințe teoretice, ci și perspective practice valoroase, care m-au ajutat să îmi dezvolt abilitățile și să mă pregătesc pentru viitoarea mea carieră. Fără implicarea și dedicarea lor, nu aș fi reușit să ajung la acest nivel de pregătire. Le sunt profund recunoscător pentru tot ceea ce au făcut pentru mine.

Nu în ultimul rând, doresc să mulțumesc părinților și colegilor mei. Părinții mei m-au sprijinit necondiționat în orice situație, încurajându-mă mereu să fiu o persoană dedicată și devotată. Alături de colegi, am trăit cea mai bună experiență profesională și personală în acești trei ani.

Sprijinul academic pe care l-am primit din partea profesorului meu coordonator a fost esențial. El a fost un mentor de la care am învățat multe lucruri valoroase, inclusiv ce înseamnă motivația și dorința de succes.

**Cuprins**

[1. INTRODUCERE 11](#_Toc1)

[1.1 Scop principal 11](#_Toc2)

[1.2 Motivație 11](#_Toc3)

[2. TEHNOLOGII ȘI FRAMEWORK-URI FOLOSITE 11](#_Toc4)

[2.1 C# 11](#_Toc5)

[2.2 ASP.NET Web API 11](#_Toc6)

[2.3 Entity Framework 11](#_Toc7)

[2.4 Identity 11](#_Toc8)

[2.5 LINQ 11](#_Toc9)

[2.5 HyperText Markup Language (HTML) 11](#_Toc10)

[2.6 Cascading Style Sheets (CSS) 11](#_Toc11)

[2.7 Blazor 12](#_Toc12)

[2.8 Blazorise 12](#_Toc13)

[2.9 Bootstrap 12](#_Toc14)

[2.10 Refit API 12](#_Toc15)

[2.11 Structured query language (SQL) 12](#_Toc16)

[2.12 NuGet Package Console 12](#_Toc17)

[3. ELEMENTE SOFTWARE FOLOSITE 12](#_Toc18)

[3.1 Visual Studio 12](#_Toc19)

[3.2 SQL Server Management Studio 20 12](#_Toc20)

[3.3 Postman 12](#_Toc21)

[4. SPECIFICAȚII ȘI REPREZENTAREA APLICAȚIEI 12](#_Toc22)

[4.1 Specificații funcționale 12](#_Toc23)

[4.2 Specificații tehnice 13](#_Toc24)

[4.3 Diagramele cazurilor de utilizare 13](#_Toc25)

[4.4 Organizarea bazei de date 13](#_Toc26)

[5. DEZVOLTAREA APLICAȚIEI 13](#_Toc27)

[5.1 Popularea bazei de date 13](#_Toc28)

[5.2 Modele de baze de date Entity Framework 17](#_Toc29)

[5.3 Sistemul de autentificarea 23](#_Toc30)

[5.4 Sistemul de blogging 26](#_Toc31)

[5.5 Sistemul de comentarii 30](#_Toc32)

[6. DESFĂȘURAREA APLICAȚIEI 31](#_Toc33)

[7. CONCLUZIE 32](#_Toc34)

[8. BIBLIOGRAFIE 33](#_Toc35)

[9. REFERINȚE 34](#_Toc36)

# 1. INTRODUCERE

## Scop principal

## Motivație

# 2. TEHNOLOGII ȘI FRAMEWORK-URI FOLOSITE

## 2.1 C#

## 2.2 ASP.NET Web API

## 2.3 Entity Framework

## 2.4 Identity

## 2.5 LINQ

## 2.5 HyperText Markup Language (HTML)

## 2.6 Cascading Style Sheets (CSS)

## 2.7 Blazor

## 2.8 Blazorise

## 2.9 Bootstrap

## 2.10 Refit API

## 2.11 Structured query language (SQL)

## 2.12 NuGet Package Console

# 3. ELEMENTE SOFTWARE FOLOSITE

## 3.1 Visual Studio

## 3.2 SQL Server Management Studio 20

## 3.3 Postman

# 4. SPECIFICAȚII ȘI REPREZENTAREA APLICAȚIEI

## 4.1 Specificații funcționale

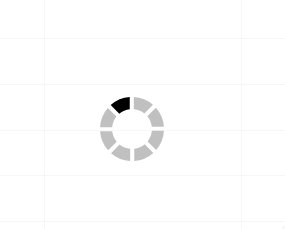
Platforma mea de blogging sau *Your blog*, sub numele în care se regăsește în proiect, este o aplicație web ce permite utilizatorilor să creeze, să publice și să vizualizeze postări ce pot conține o descriere, un titlu și o imagine. Acestea sunt grupate pe categorii, cum ar fi: fotbal, IT, music, etc. Utilizatorii sunt de 2 tipuri: useri normali și administratori. Administratorii au, în plus de utilizatorii normali, posibilitatea de a șterge blogurile și comentariile celorlați și posibilitatea de a crea categorii de bloguri noi.

În aplicație există un singur administrator. Ceilalți useri pot fi creați prin sistemul de înregistrare al unui cont nou.

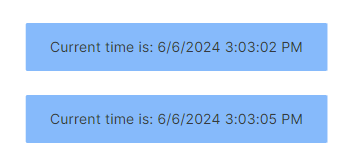
Detaliile despre utilizatori sunt: nume de utilizator (username), email, poză de profil, rol și parolă. Toate informațiile private sunt codate în baza de date.

Un user poate să adauge și comentarii la blogurile celorlalți, poate să vizualizeze profilul celorlalți, să își vizualizeze propriul profil dar și să îl editeze. Dacă utilizatorul nu își alege o imagine de profil, platfoma o să îi atribuie una predefinită.

Toate comunicările dintre interfață și API sunt așteptate, timp în care utilizatorului nu îi este permis să facă vreo modificare pe pagină, prin apariția unui *ecran de încarcare.*



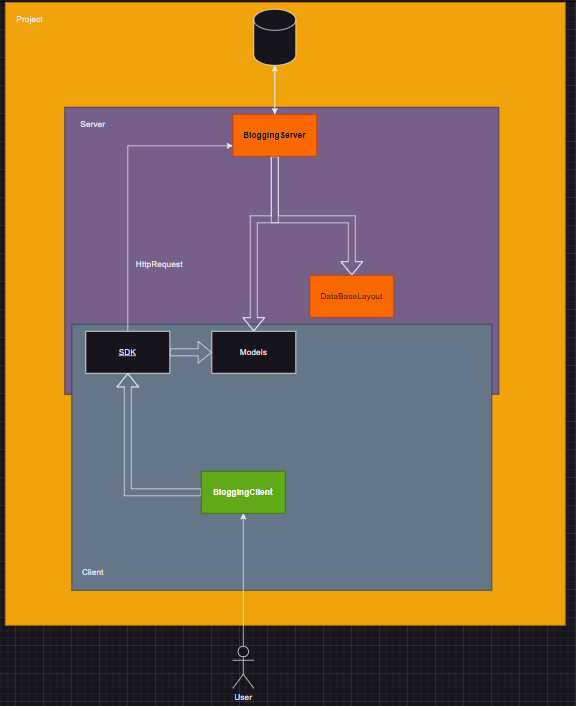
Utilizatorii sunt informați despre eventualele erori sau informări despre cererile lor către API printr-un *toast*, care apare în jocul paginii:



## 4.2 Specificații tehnice

Aplicația este împărțită în două proiecte: BloggingServer și BloggingClient.

BloggingServer reprezintă API-ul, proiectul prin care interfața comunică cu baza de date. Comunicarea se face prin Refit. Această soluție conține mai multe proiecte formând o structură care facilitează extinderea funcționalităților.

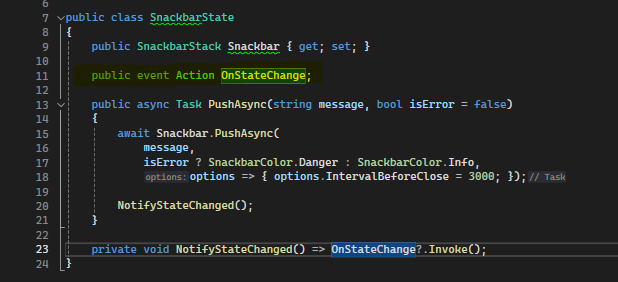


Proiectul *SDK* conține toate endpoint-urile disponibile în API (BloggingServer). Toate modelele de cerere sau de răspuns sunt puse în proiectul de *Models*.

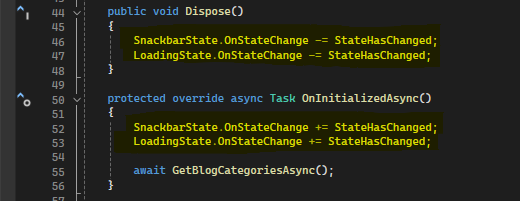
*BloggingClient* injectează un *ApiClient* de fiecare dată când are nevoie să facă un request către Server:



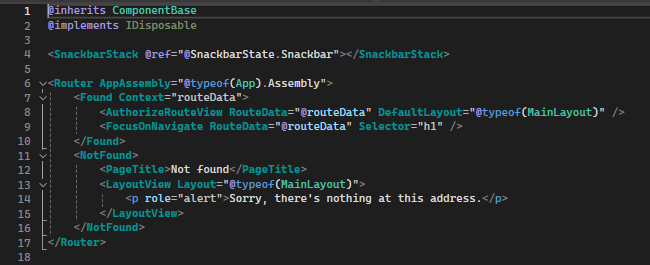
Pentru a pune în așteptare pagina, până se execută cererea către API, dar și pentru a notifica rezultatul în urma finalizării request-ului, se folosesc *state*-uri. Clase injectate *singleton*, care folosesc un *Action* pentru a notifica toate componentele care sunt abonate la acestea:



În momentul în care are loc o acțiune, componentele folosesc metoda *StateHasChanged* pentru a notifica contextul despre schimbari. Folosim interfața *IDisposable,* care apelează ca un destructor metoda *Dispose*, pentru a dezabona componentele de la state-uri și pentru a evite *memory leaks*.



Proiectul începe de componenta \_Host.cshtml care folosește ca și layout: \_Layout.cshtml. Aici se injectează toate referințele de la toate librăriile și se randează un body. Body care reprezentat de un routing pus la dispoziție de către Blazor, prin care trec toate paginile:



## 4.3 Diagramele cazurilor de utilizare

## 4.4 Organizarea bazei de date

# 5. DEZVOLTAREA APLICAȚIEI

## 5.1 Popularea bazei de date

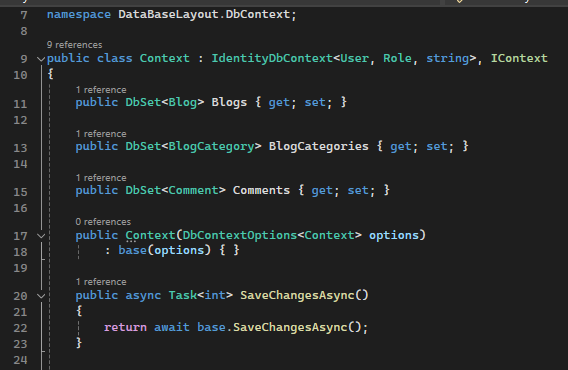
Popularea bazei de date este esențială în configurarea aplicație. În cazul de față folosim SQL Server împreună cu Entity Framework și Identity pentru a gestiona autentificarea și operațiile pe date.

Enity Framework (EF) este un ORM (Object-Relational Mapper) dezvoltat de Microsoft care permite dezvoltatorilor să lucreze cu o bază de date folosind obiecte .NET. Una dintre cele trei abordări principale ale EF este Code First, care permite dezvoltatorilor să definească modelul de date folosind clase C# obișnuite (POCO - Plain Old CLR Objects) și apoi să genereze schema bazei de date pe baza acestor clase.

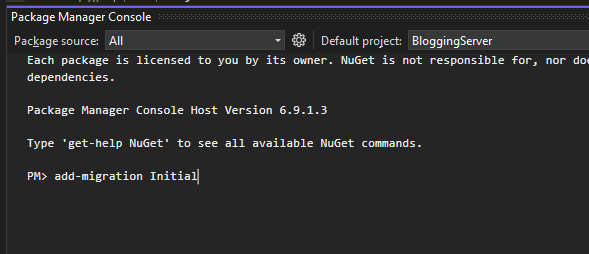
O entitate în Entity Framework (EF) reprezintă o clasă C# care este mapată la o tabelă din baza de date. Fiecare instanță a acestei clase corespunde unei rând din tabelă.

Code First permite dezvoltatorilor să creeze modelul de date prin scrierea de cod C#. Aceasta înseamnă că nu este necesar să existe o bază de date preexistentă, deoarece EF poate crea baza de date și tabelele pe baza modelului definit în cod. Acest lucru oferă flexibilitate și control complet asupra designului modelului de date.

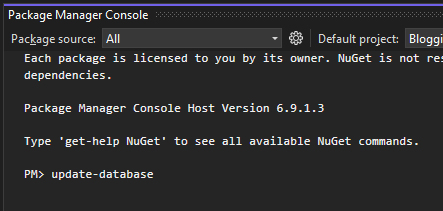
Folosind *Entity Framework*, avem definit contextul de date, prin clasa *Context*, care extinde IdentityDbContext pentru gestiona utilizatorii, folosind *ASP.NET Core Identity*. Contextul de date definește seturile de entități care vor fi folosite în baza de date:



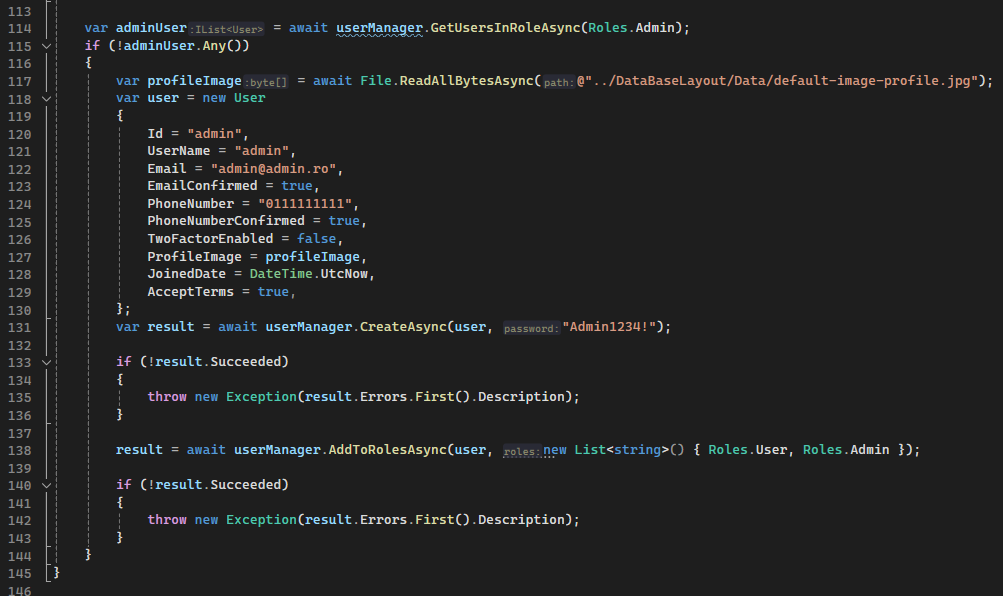
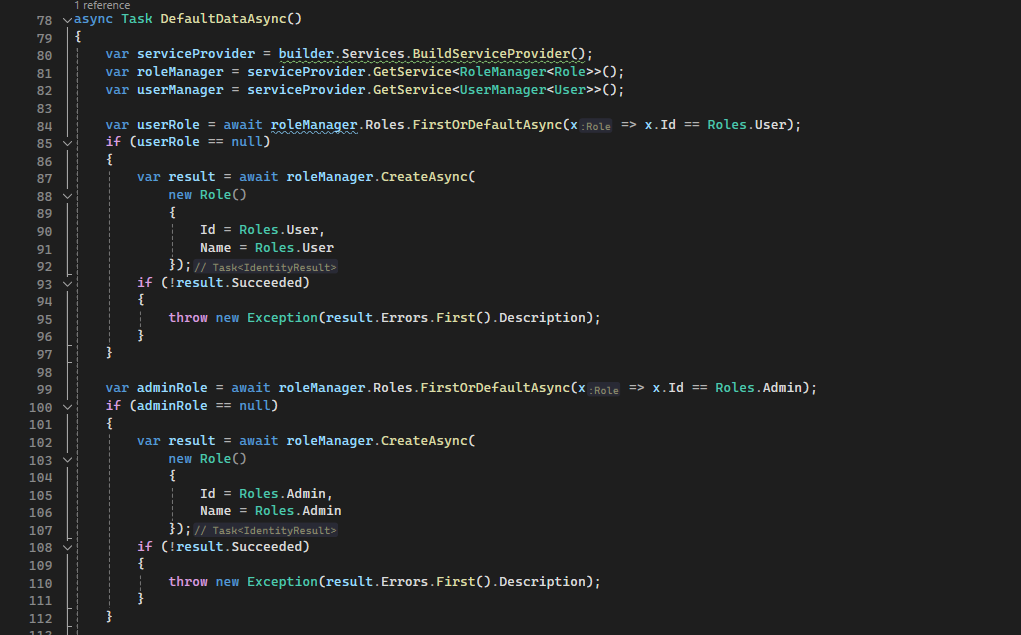
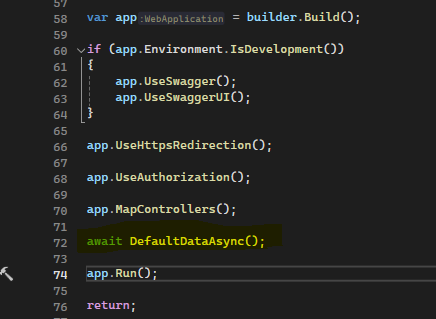
Pentru a genera baza de date, mai întâi trebui generate migrațiile. Migrațiile în Entity Framework (EF) sunt un mecanism prin care modificările aduse modelului de date în cod sunt reflectate în schema bazei de date. Acestea permit gestionarea și aplicarea modificărilor astfel încât baza de date este sincronizată cu modelul de date definit în cod. Pentru acest lucru se vor executa în Package Manager Console următoarele comenzi:



Iar pentru a rula aceste migrații se va executa:



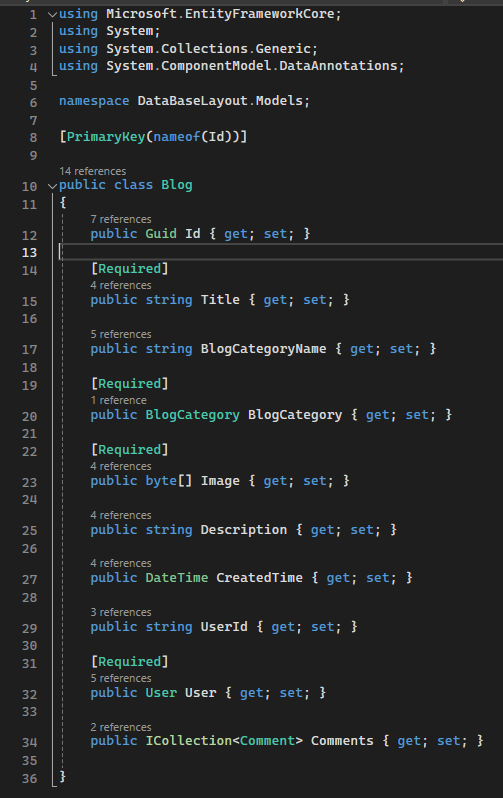
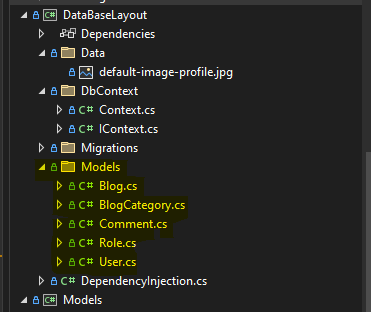
Toate entitățile din aplicație sunt populate din partea de Client a proiectului, de către toți utilizatorii care și-au creat un cont. În aplicație există 2 roluri: *User* și *Admin*. Acestea sunt adăugate automat de fiecare dată când server-ul rulează. Cu rolul *Admin* există un singur cont, care, de asemenea, este generat la fiecare runtime:



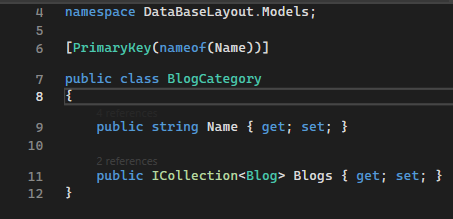
Se adaugă rolurile: User și Admin, în bază, dacă nu există, la fel și user-ul cu rolul *Admin*.

## 5.2 Modele de baze de date Entity Framework

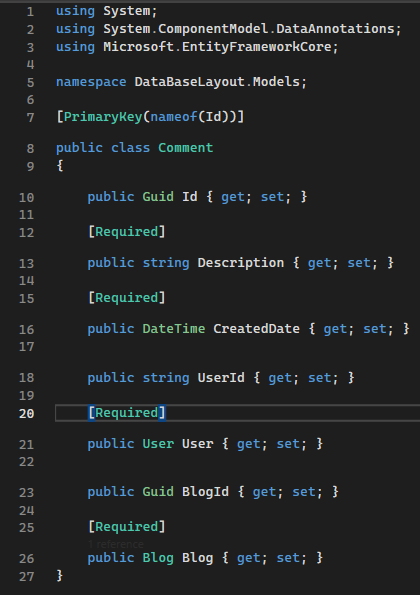
Modelele de date în Entity Framework (EF) sunt reprezentări ale entităților din aplicație, utilizate pentru a interacționa cu baza de date. Aceste modele sunt adesea clase C# care definesc structura și relațiile datelor pe care le gestionăm. Prin intermediul acestor modele, EF poate traduce operațiunile efectuate asupra obiectelor în comenzi SQL care, mai apoi sunt trimise către baza de date și interpretate. Aplicația noastră dispune de cinci astfel de modele:



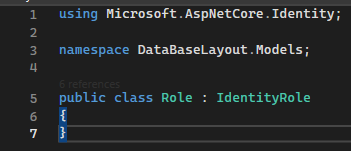
Un blog conține id-ul, titlul, o categorie, o imagine, o descriere, data creării, utilizatorul care a inițiat crearea blogului dar și mai multe comentarii.



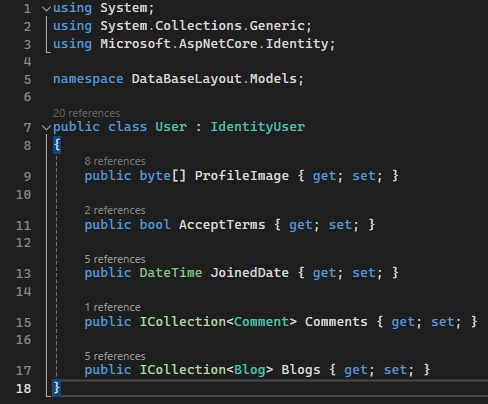
Categoria unui blog conține doar numele și blogurile care aparțin acelei categorii. Acestea sunt unice prin nume.



Un comentariu conține un Id, o descriere, data creării, utilizatorul care a creat comentariul și blog-ul la care a fost atribuit.

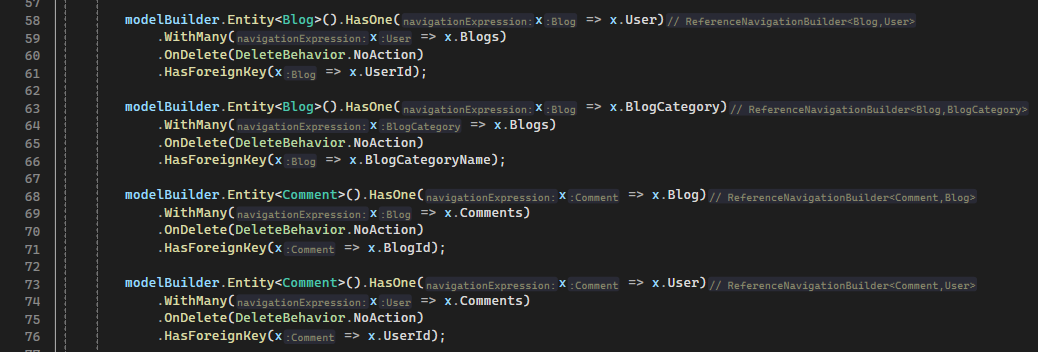


*Role* este un *IdentityRole*, ce conține numele rolului. Acesta are referințe de many-to-many cu *User*, făcute de către Identity. Această referință este suprascrisă în *Context*, pentru a păstra convențiile de nume din proiect:



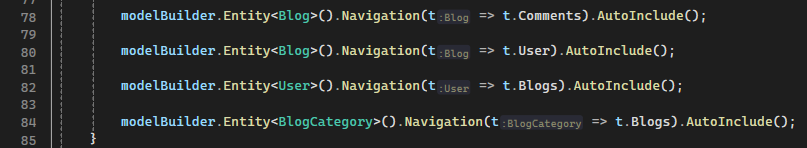
*User* definește utilizatorul și conține în plus, pe lângă proprietățile standard oferite de *IdentityUser* (username, email, etc.), imaginea de profil, dacă a acceptat termini și condițiile, data la care s-a înregistrat în platformă și relația de one-to-many dintre acesta și *Comments*, plus *Blogs*. Astfel, un utilizator poate scrie mai multe comentarii și bloguri, însă un comentariu, respective un blog poate fi scris de mai multi utilizatori.

În caz de ștergere sau actualizare, *Context*-ul este configurat să nu facă operația în cascadă pentru că toate entitățile noastre pot fi create și pot exista independent.



Prin aceste modele se evidențiază tipul pe care fiecare proprietate ar trebui să îl aibă, valorile pe care le pot avea, dar și relațiile dintre entități: one-to-one, one-to-many, many-to-many. Astfel se facilitează mult mai ușor accesarea acestora în cod.

Pentru ca aceste referințe să fie accesate se folosește noțiunea de *AutoInclude.* AutoInclude este o caracteristică introdusă în Entity Framework Core 6 care permite încorporarea automată a relațiilor la interogările LINQ fără a fi nevoie de includerea explicită a acestora în cod.



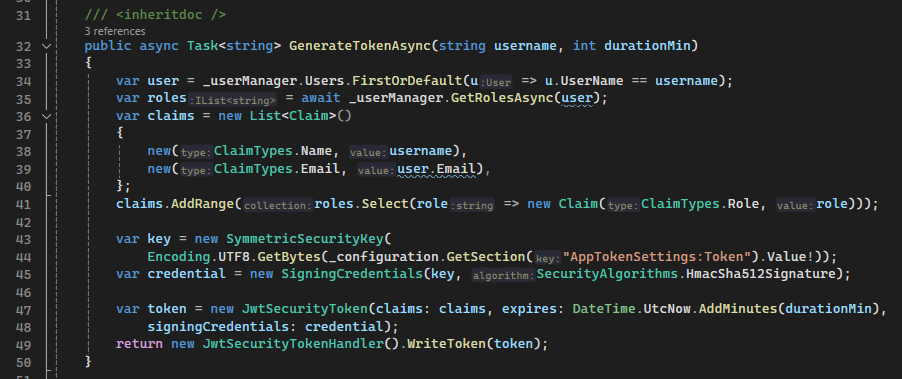
## 5.3 Sistemul de autentificarea

Sistemul de autentificare este partea esențială a aplicații deoarece verifică identitatea utilizatorilor și le acordă acces la diferite resurse ale aplicației în funcție de permisiunile lor. Acesta asigură securitatea și confidențialitatea datelor prin autentificarea utilizatorilor și gestionarea sesiunilor.

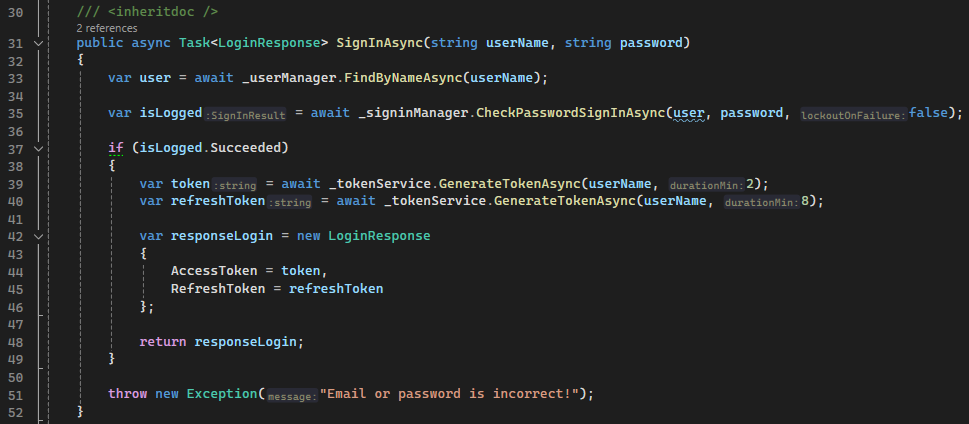
Atât autentificarea cât și autorizarea se realizează prin Bearer Token.

Autentificarea Bearer (numită și autentificare cu token) este o schemă de autentificare HTTP care implică jetoane de securitate numite Bearer Token. Tokenul este un șir criptic, generat de obicei de server ca răspuns la o solicitare de conectare. Clientul trebuie să trimită acest token în header-ul *Authorization* atunci când face cereri către API.

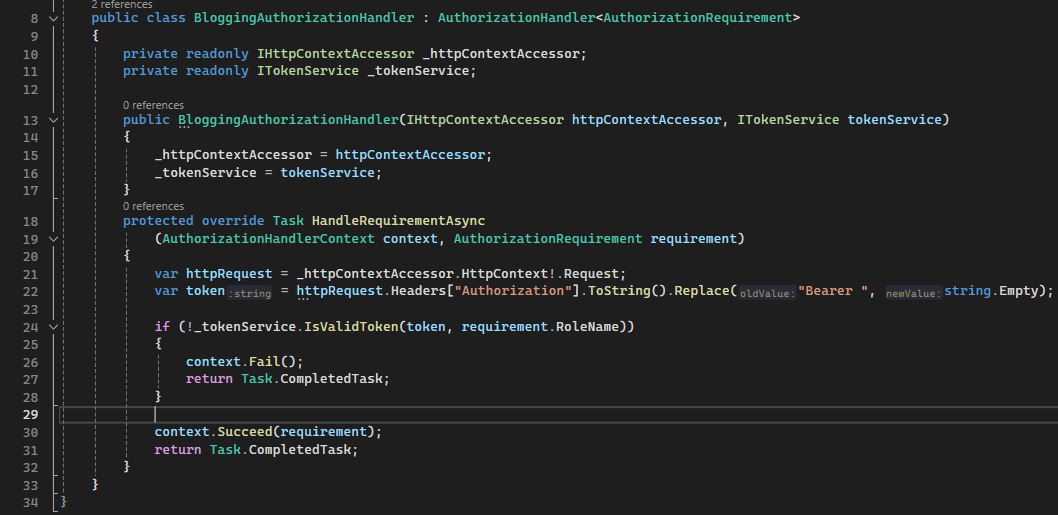
Pentru generarea acestui token, se folosește JwtSecurityToken:



Acesta este apelat de către controller, în momentul în care utilizatorul inițiază operația de *Login*:



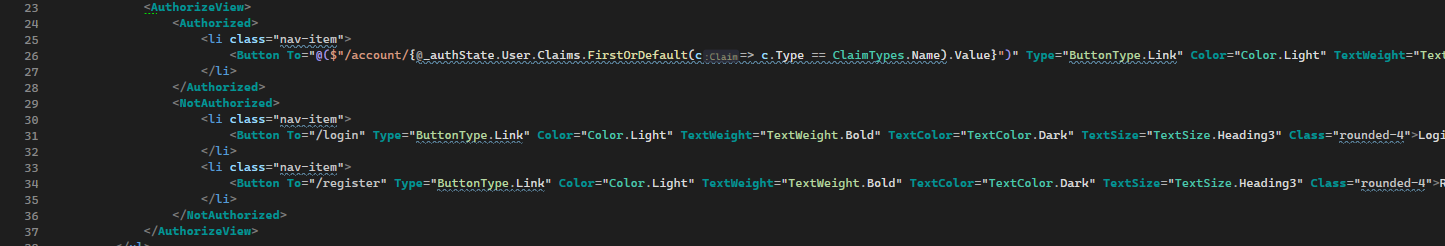
Toate rolurile utilizatorului se criptează ca și Claims în token. Toți utilizatorii care vor să facă diferite operații sau să acceseze resurse din platformă trebuie să ofere un token pentru a le verifica identitatea. În *backend* acest lucru se face printr-un atribut definit la fiecare request:



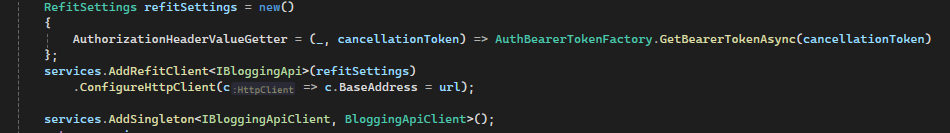
Acesta are o perioadă de expirare. Dacă este expirat accesul este restricționat. Utilizatorul își poate folosi un al doilea token, numit și *refreshToken* pentru a regenera un alt token, fără a fi nevoit să repete pașii de login. Dacă și acest refreshToken este expirat atunci sesiunea se încheie și utilizatorul este nevoi să se logheze din nou. Toate acestea se fac prin intermediul Refit-ului, care în spate pune la dispoziție un *HttpClient*, prin intermediul căruia *frontend*-ul apelează controller-ele din API.

În client-side, token-ul este salvat în *Local Storage*. Local Storage este o tehnologie de stocare web care permite aplicațiilor web să stocheze date local, direct în browser-ul utilizatorului. Aceasta este parte a specificației Web Storage și oferă o modalitate simplă și eficientă de a păstra datele pe partea clientului fără a fi nevoie de servere sau baze de date externe.

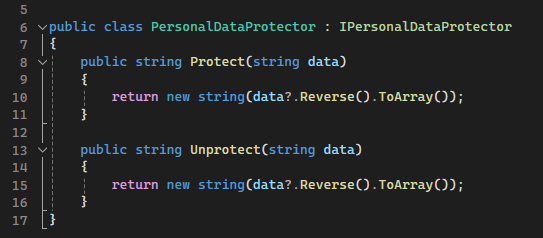
Tot acest flow este susținut de un *AuthenticationStateProvider*. Acesta este folosit ca un CascadeParameter, prin care notifică toate componentele că *state*-ul s-a schimbat. Blazor pune la dispoziție un tag care verifică dacă user-ul este autentificat și autorizat:



Refit-ul este configurat ca la începutul fiecărui request să acceseze acest storage și să atașeze în header-ul *Authorization* token-ul de acces:

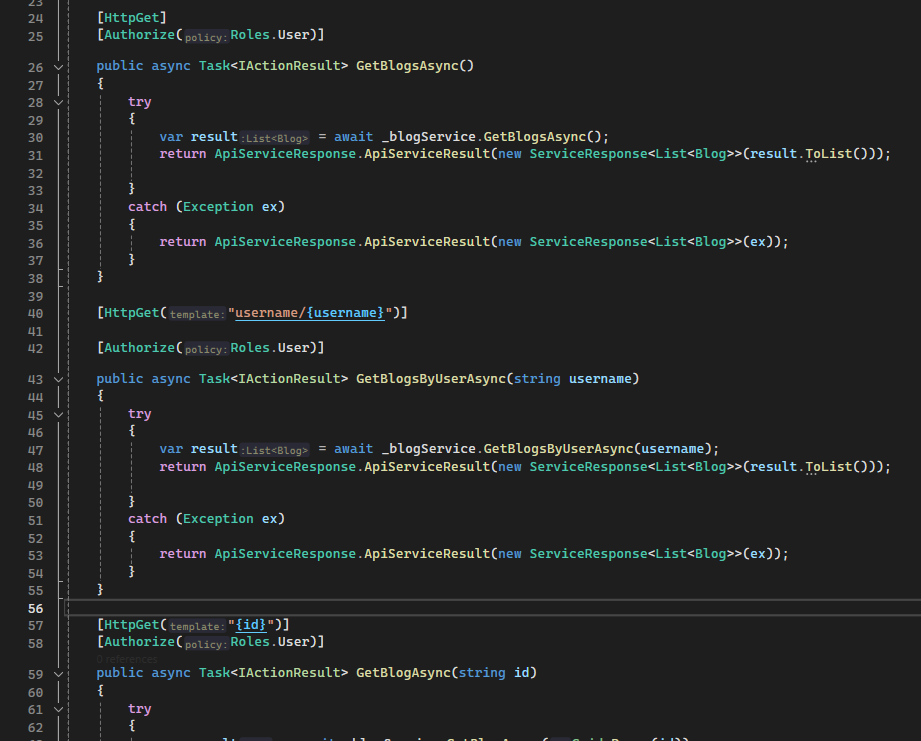


Utilizatorul poate face o cerere de înregistrare din interfață. Toate informațiile private sunt condificate folosind inversarea caracterelor, iar parola folosește un HASH prestabilit:



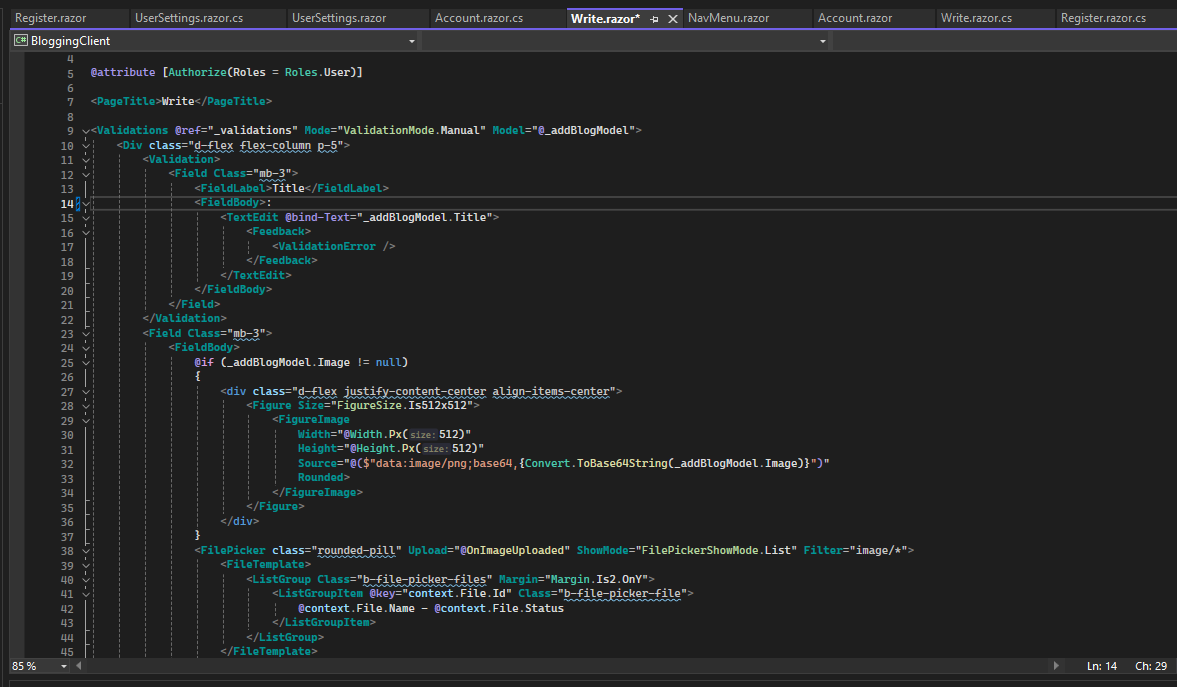
## 5.4 Sistemul de blogging

Sistemul de blogging oferă utilizatorilor să vizualizeze, dar și să creeze blog-uri noi. Pentru toate acestea, *user*-ul trebuie să fie autentificat dar și autorizat cu rolul de User sau Admin:

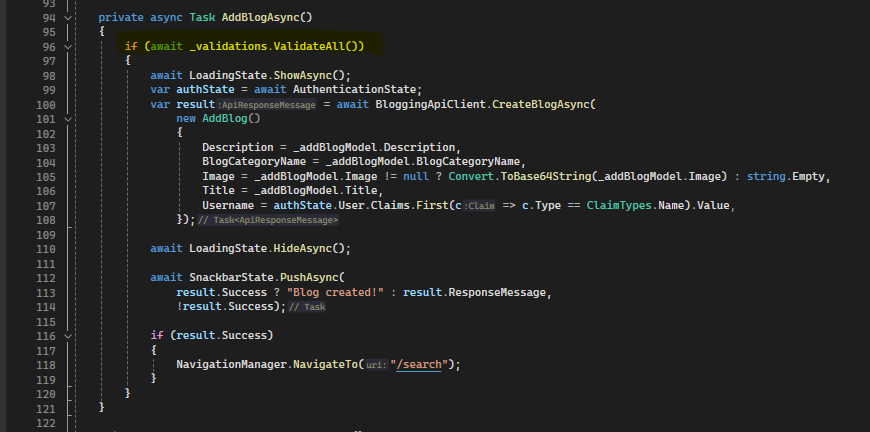


Admin-ul, în acest proces, are dreptul de a șterge și blog-urile celorlalți Useri.

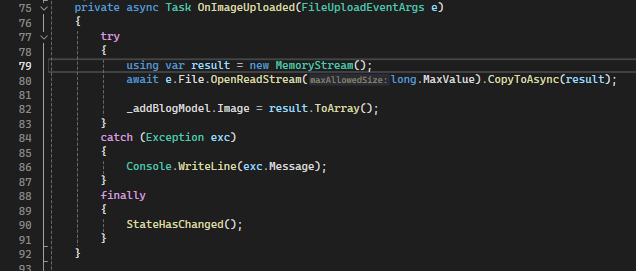
Pentru scrierea unui blog, utilizatorul trebuie să ofere un titlu, o imagine, o descriere și o categorie. Categoriile pot fi adăugate doar de către admini și sunt unice prin nume. Aceste intrări sunt create folosind componentele de care dispune librăria Blazorise:



Pentru validarea datelor se folosește componenta *Validation* care este trigger-uită de fiecare dată când se apasă butonul de Save:

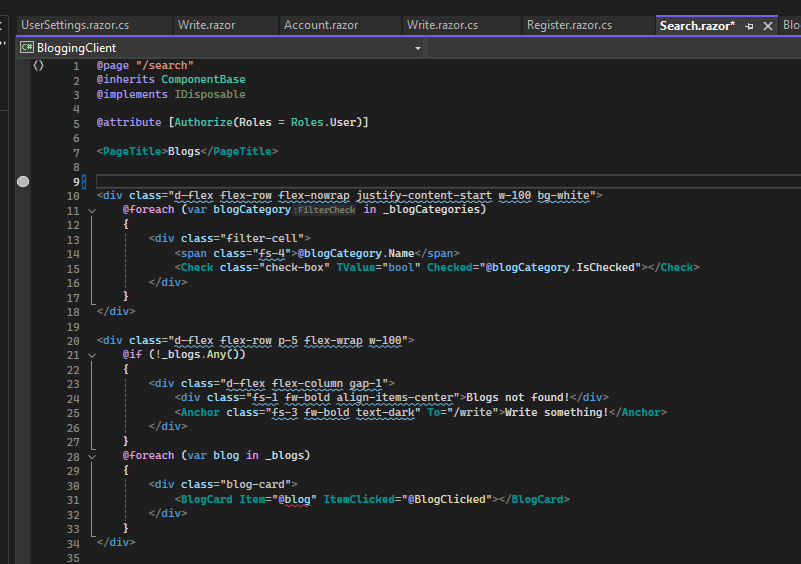


Imaginea se introduce de către utilizator printr-un *FilePicker* care convertește input-ul într-un *byte array*:

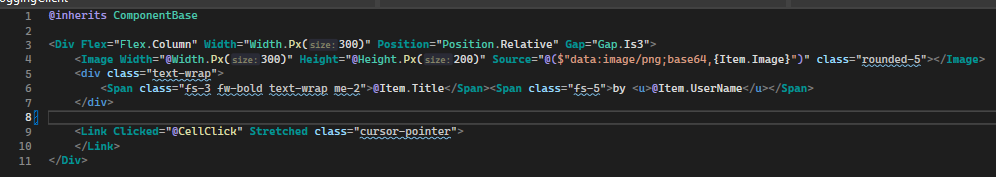


Metoda *StateHasChanged()* notifică toate componentele că s-au efectuat modificări pe contextul curent.

Toate blog-urile se găsesc sub pagina */search/{parameters}* care, parcurge toate blog-urile înregistrate în platformă, și le afișează într-un format prietenos pentru utilizator:

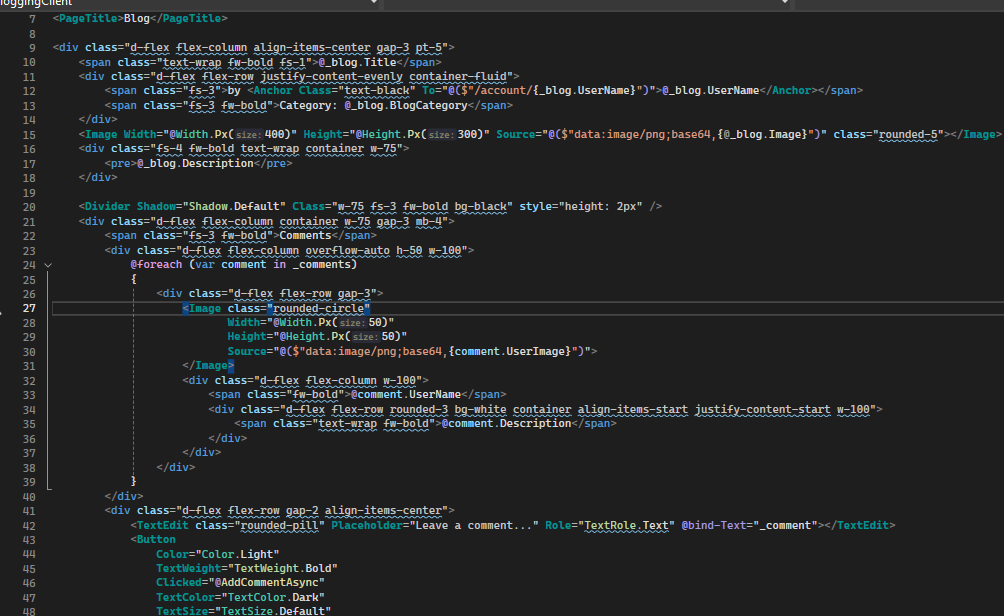


Pentru a pastra principiul Single Responsability, s-a creat o componentă separată pentru un card blog:



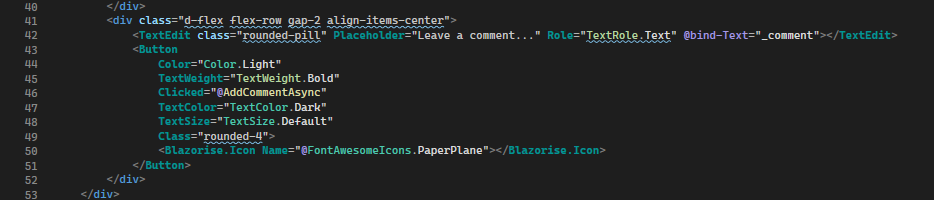
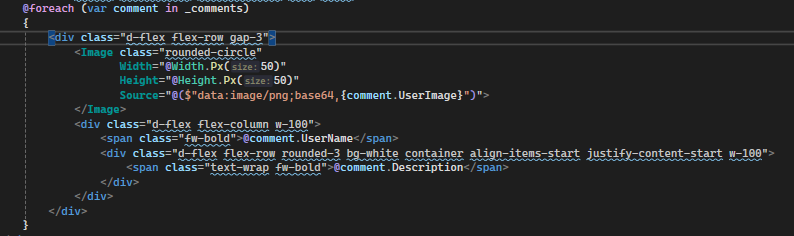
Această componentă conține un *Link* care este *Stretched*, ceea ce înseamnă că oriunde s-ar apăsa click pe acel card, se va face un fire event către acel *Link*. Acesta v-a naviga user-ul către pagina de detaliu al unui blog.

În pagina de detaliu al unui blog se regăsește în plus, descrierea și comentariile lăsate de către utilizatori:



## 5.5 Sistemul de comentarii

Sistemul de comentarii permite oricărui utilizator să își exprime idei despre o anumită postare. Oricine este înregistrat poate să lase sau să vadă un comentariu din secțiunea de detaliu al unui blog:



# 6. DESFĂȘURAREA APLICAȚIEI

# 7. CONCLUZIE

În concluzie, subliniez că realizarea acestui proiect a fost o adevărată provocare pentru mine. Pe parcursul întregului proces, am întâmpinat diverse situații care m-au pus la încercare. Cu toate acestea, prin îmbunătățiri constante, multe căutări și documentări, am reușit să duc la bun sfârșit acest proiect.

Pe parcursul anilor de facultate, prin studiu și efort constant, dar și prin realizarea acestui proiect, am reușit să-mi îndeplinesc un scop personal: acela de a acumula cât mai multă experiență profesională și personală.

**SFÂRȘIT.**

# 8. BIBLIOGRAFIE

Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Moderns JS Libraries, Eric Elliott

JavaScript: The Definitive Guide, David Flanagan

Eloquent JavaScript: A Modern Introduction to Programming, Marijn Haverbeke

JavaScript: The Good Parts, Douglas Crockford

[The Road to React with Firebase, Your journey to master advanced React for business web applications,](https://www.amazon.com/dp/1795010894?tag=uuid10-20" \t "_blank) [[Robin Wieruch](https://www.amazon.com/dp/1795010894?tag=uuid10-20" \t "_blank)](https://bookauthority.org/author/Robin-Wieruch)

# 9. REFERINȚE

*Autentificarea Bearer* . (2024). Preluat de pe https://swagger.io/docs/specification/authentication/bearer-authentication/.

*AutoInclude*. (fără an). Preluat de pe https://learn.microsoft.com/en-us/ef/core/querying/related-data/eager.

*CascadeParameter*. (2024). Preluat de pe https://learn.microsoft.com/en-us/aspnet/core/blazor/components/cascading-values-and-parameters?view=aspnetcore-8.0.

*Code First*. (fără an). Preluat de pe https://learn.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database.

*Enity Framework (EF)*. (2024). Preluat de pe https://learn.microsoft.com/en-us/ef/core/.

*entitate*. (2024). Preluat de pe https://learn.microsoft.com/en-us/previous-versions/office/developer/sharepoint-2010/ee536692(v=office.14).

*Local Storage* . (2024). Preluat de pe https://www.w3schools.com/html/html5\_webstorage.asp.

*Migrațiile*. (2024). Preluat de pe https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli.