UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

MANAGEMENTUL CAZĂRILOR ÎN CĂMINE

propusă de

Panciu-Rusu Mihai

Sesiunea: iulie, 2019

Coordonator științific

Prof. Colab. Olariu Florin

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

FACULTATEA DE INFORMATICĂ

MANAGEMENTUL CAZĂRILOR ÎN CĂMINE

Panciu-Rusu Mihai

Sesiunea: iulie, 2019

Coordonator științific

Prof. Colab. Olariu Florin

Cuprins

Introducere

Motivație

Perioada de studenție este una minunată, poate chiar cea mai frumoasă perioadă a vieții noastre. Pentru unii dintre noi reprezintă etapa în care începem să devenim independenți și plecăm într-un oraș nou, cu oameni necunoscuți. De aceea este important locul unde stăm în perioada studenției, din acest motiv unii preferă să locuiască în chirie, dar cu toții știm că viața la cămin este unică.

Cu toate acestea, procesul de cazare într-unul din căminele disponibile este destul de incomod și anevoios, mai ales pentru un student care urmează să intre în primul an de facultate și pot spune asta din proprie experiență.

Spre exemplu, un lucru neplăcut este faptul că studentul trebuie să vină personal la secretariatul facultății doar pentru a se semna pe o foaie pentru a fi luat în considerare la repartizarea locurilor, mai ales în condițiile în care domiciliul studentului este la o distanță mare de facultate. Acest lucru reprezintă o risipă destul de mare de timp și bani.

Mai mult decât atât, durata de repartizare a locurilor este destul de mare, deoarece repartizarea se face manual de o singură persoană. Această sarcină poate dura zile și totodată se pot strecura ușor erori umane care pot complica și mai mult procesul de cazare.

Un alt aspect dificil de realizat este o distribuție clară cu studenții care au solicitat cazări în vederea alocării locurilor de cazare în mod optim din punct de vedere al cererii și ofertei.

Și nu în ultimul rând, este nevoie de o evidență ușor de accesat în care se poate vedea situația celor cazați și căminul în care au fost repartizați.

Datorită tuturor acestor motive, am ales să realizez o aplicație web care să faciliteze întregul procesul de cazare în cămine și de evidență a celor cazați, cât și a locurilor disponibile.

Obiective generale

Primul obiectiv este să ne asigurăm că toți utilizatorii acestei aplicații sunt studenți ai facultății noastre (în cazul nostru Facultatea de Informatică). Pentru acest lucru administratorul trebuie să adauge individual studentul sau să încarce documente CSV cu situația studenților care să conțină de asemenea și alte informații suplimentare legate de aceștia, cum ar fi rezultatele academice, adresa de email, anul de studiu etc. Această acțiune poate fi realizată doar de un administrator de cazări, iar în urma importării datelor, aplicația va crea în mod automat conturile de studenți. Aceste conturi se trimit la fiecare student însoțite de o parolă generată aleator, cu posibilitatea de a o schimba după preferință.

După ce studentul a fost înregistrat cu succes de către administrator, acesta poate intra în aplicație pentru a solicita un loc de cazare. El este nevoit să completeze un formular online unde este prezentă și lista de preferințe aleasă de el. Studentul are facilitatea de a-și face un clasament al căminelor unde dorește să fie cazat.

Totodată administratorul poate posta diverse articole sau anunțuri legate de procesul de cazare, eventuale informații suplimentare sau noutăți de interes public. Într-un articol poate adaugă imagini și text formatat. La vizualizarea lor poate avea acces oricine, chiar dacă nu este student al facultății iar cititorii pot vedea când a fost postat articolul și de către cine.

Administratorul poate vizualiza toate cererile create însoțite de punctajele și preferințele studenților. Acesta are libertatea de a filtra datele sau de a căuta rapid pe cineva. De asemenea, este prezentă și o statistică în timp real a cererilor pe fiecare an pentru băieți și fete, unde este precizat procentajul de cereri raportat la numărul de studenți pe an în funcție de băieți sau fete. Această statistică oferă o perspectivă de ansamblu asupra cererilor și unde este nevoie de suplimentarea numărului de locuri.

În momentul în care sunt stabilite locurile disponibile pentru cazare, administratorul accesează aplicația și introduce toate căminele în care s-au oferit aceste locuri. De asemenea, dânsul are posibilitatea de a adăuga pentru fiecare categorie în parte (băieți/fete) numărul de locuri din fiecare cămin. În urma acestei acțiuni, aplicația calculează în mod automat un număr de locuri de rezervă pentru situațiile neprevăzute. În mod implicit se vor salva 10% din locurile

totale, cu posibilitatea de a modifica acest procentaj în mod manual sau de a scrie efectiv câte locuri se doresc a fi salvate. La final se afișează locurile rămase și se salvează.

După ce s-au introdus locurile, administratorul trebuie să le distribuie pe ani în funcție de băieți și fete. Aplicația vine în ajutorul lui și îi oferă o variantă de distribuire bazată pe o statistică. Locurile pot fi editate ușor reducând astfel nivelul de muncă deoarece se oferă deja o variantă de distribuire care poate fi ajustată.

În perioada în care se dorește publicarea locurilor primite de către studenți în cămine, administratorul poate genera un tur de cazări în care trebuie să specifice ce cereri vor fi luate în considerare pentru acel tur și va face acest lucru prin setarea intervalului calendaristic în care s-au efectuat cereri. După ce s-a realizat această operațiune, aplicația va repartiza locurile disponibile în funcție de rezultatele academice obținute de studenți și de preferințele acestora. Aplicația reușește să ofere aceste date în circa 3 secunde, în loc de zile, cât ar fi durat dacă ar fi fost făcute manual.

În momentul acesta administratorul poate revizui această listă și în final poate publica lista repartizărilor pentru toți, împreună cu alte informații suplimentare, cum ar fi termenul limită pentru confirmarea locurilor și când va apărea următorul tur.

Odată ce lista a fost publicată, studenții pot verifica plini de emoții dacă au avut norocul să se regăsească pe acea listă. În caz de reușită, acesta își poate descărca dispoziția de cazare din aplicație și să o tipărească. Dispoziția de cazare va avea precompletate datele studentului cu locul primit și alte date administrative iar studentul va trebui doar să o semneze. Deținând această dispoziție de cazare, studentul va fi nevoit să se prezinte la administratorul de cazare pentru a-și confirma locul. Doar administratorul poate confirma locurile studenților, în felul acesta ne asigurăm că nu vor fi cazuri de locuri confirmate și nerevendicate.

Acum studentul este prezent pe lista celor confirmați și își poate finaliza cazarea la căminul la care a fost repartizat.

Pentru tururile viitoare se va proceda la fel, selectând intervalul calendaristic în care s-au făcut cereri și se vor realoca locurile neconfirmate din turul anterior, iar în cazul în care un student dorește să renunțe la loc, administratorul va putea cu ușurință să îl scoată din lista celor confirmați iar locul lui va fi realocat în următorul tur.

Alte facilități ale studenților ar fi prezența paginii cu informații utile din campus și cum pot fi contactate persoanele responsabile de cazări.

Scurtă descriere a soluției

Aplicația curentă este una accesibilă cu ajutorul internetului, deci soluția construită poate fi descrisă ca o aplicație de tip client-server cu următoarea componență:

- **Aplicația server:** este reprezentată sub forma unui REST API [1] care expune puncte de acces prin care se pot manipula date.
- **Aplicația client:** reprezintă o interfață grafică, ce utilizează serviciile expuse de server, apelurile la server se realizează prin intermediul protocolului HTTP [2]care transmite date în format JSON.
- **Baza de date:** reprezintă o baza de date de tip relațională, prin intermediul căreia datele aplicației sunt stocate.

Abordare tehnică

Limbaje și medii de programare

Entity Framework Core este versiunea independentă de platforma a ORM[3]-ului Entity Framework. Acest ORM (Object Relational Mapping) este folosit în .NET Core pentru a lucra cu o bază de date folosind obiecte.

Swagger: Reprezintă un pachet NuGet în .NET Core care iți descrie structura API-ului. Oferă o prezentare plăcută a tuturor controalelor și metodelor fiind destul de util pentru a crea o documentație ușor de citit. Reprezentările pentru câteva controale ale aplicației se pot vedea în *Figura 1*.

[1] RESTful API - https://searchmicroservices.techtarget.com/definition/RESTful-API
[2] HTTP - https://ro.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[3] ORM - https://www.todaysoftmag.ro/article/73/analiza-mecanismului-object-relational-mapping-orm-cu-exemplificari-hibernate

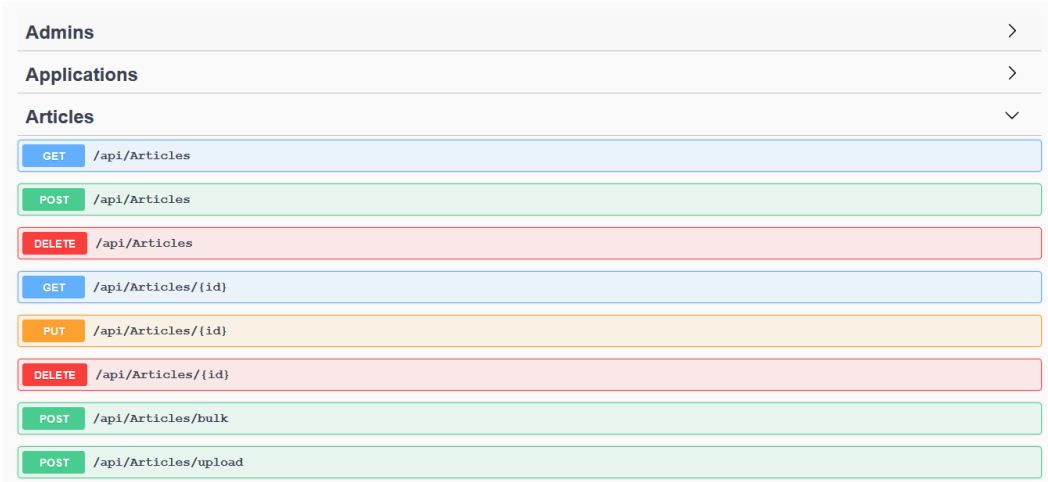

Figura 1: Controale afișate în Swagger

Vue: este un cadru de lucru structural în întregime creat cu ajutorul limbajului JavaScript ce poate fi folosit pentru a crea partea de client a unei aplicații. Acest framework a prins multă popularitate datorită simplității arhitecturale pe care o prezintă și a flexibilității de care dispune. Am ales Vue pentru partea de client deoarece este Single Page Application [4] și a modului de lucru prezentat în *Figura 2*. Prezintă și o librărie Vuex [5] care este un depozit centralizat unde componentele pot stoca informații care sunt accesibile oriunde, foarte util în cazul aplicațiilor mai complexe.

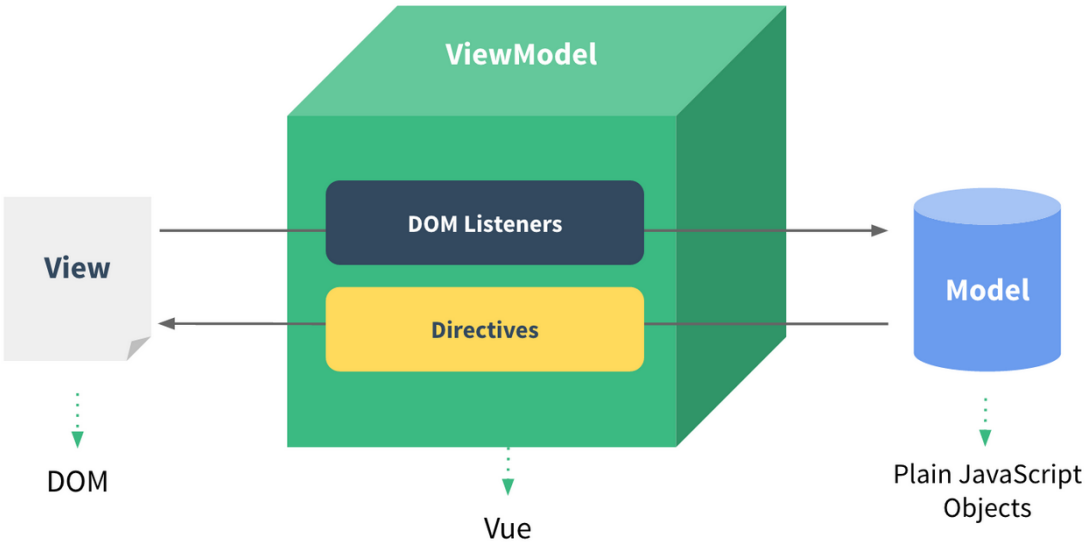

Figura 2: Perspectiva conceptelor din Vue [Sursa: https://012.vuejs.org/guide/]

[4] Single Page Application - https://www.quora.com/What-is-a-single-page-application-in-web-development
[5] Vuex - https://vuex.vuejs.org/

Vuetify: reprezintă o librărie grafică pentru Vue care oferă componente reutilizabile pentru interfață, astfel ajută aplicația să capete un aspect profesional.

De asemenea au fost folosite următoarele limbaje și cadre de programare: .NET Core, ASP.NET Core, C#, HTML, JavaScript, CSS. Despre toate acestea se pot afla mai multe detalii în cadrul *Anexei - Limbaje si cadre de programare.*

Instrumente software

Visual Studio este un IDE [6] creat de cei de la Microsoft cu ajutorul căruia se pot construi diverse tipuri de aplicații, cum ar fi desktop, servicii web sau pentru mobil. Am ales acest editor datorită avantajelor pe acesta le oferă, cum ar fi: sugestii de cod pe care acesta le oferă în timp real, accesibilitatea rapidă la funcții și ușurința creării unei arhitecturi optime pentru aplicație. Un alt avantaj din punctul meu de vedere foarte mare este prezența extensiei Resharper[7] care poate fi considerat colegul tău pentru programare, acesta oferind diverse sugestii despre cum să scrii un cod de calitate și ușor de menținut.

Visual Studio Code este un editor de cod care poate suporta mai multe limbaje de programare. L-am ales datorită simplității pe care o prezintă, fiind disponibil pentru Windows, Linux și MacOS. Acesta are un suport foarte bun pentru JavaScript și are consolă integrată făcând foarte eficientă testarea și depanarea de cod.

Postman reprezintă o unealtă software care poate fi utilizată pentru a simula cererile HTTP care ar fi în mod normal trimise de către un client către server. Utilitatea lui a fost întărită de nevoia testării serviciilor oferite de către server.

Microsoft SQL Server Management Studio este un program care ajută la gestionarea, vizualizarea și modificarea bazelor de date de tip MySQL. Am folosit acest instrument software pentru a modifica sau insera rapid date necesare testării aplicației sau pentru a viziona structura bazei de date. Oferă și un sistem prin care se pot salva versiunile anterioare ale bazei de date, cu posibilitatea de a reveni la o versiune anterioară dacă se dorește.

---

[6] IDE - https://ro.wikipedia.org/wiki/Mediu_de_dezvoltare
[7] Resharper - https://www.jetbrains.com/resharper/

1 Contribuții

Petrecând anii de studenție în cămin, am avut parte de multe experiențe de neuitat, însă perioada de cazări a reprezentat mereu un stres destul de mare pentru mine. În ciuda faptului că aveam o situație academică bună, nu am fost scutit de griji. Drumurile constante, cozile și eventualele dispute care sunt generate în perioada de cazări m-au făcut să mă întreb de ce nimeni nu face ceva în legătură cu acest proces anevoios. Acela a fost momentul în care m-am hotărât că eu pot să aduc o schimbare, să pun o cărămidă în procesul de construire a unui sistem fluid, eficient și plăcut prin care studenții pot să își găsească "casa lor studențească". Prin acest mod, studenților le sunt oferite mai multe amintiri plăcute din studenție, încurajând evitarea chiriilor scumpe și încurajându-i să-și trăiască cu adevărat viață de student alături de alți tineri care le vor deveni prieteni.

Am discutat mai întâi cu coordonatorul de licență despre această propunere care mi-a oferit aprobarea dânsului, iar după am decis să caut informații legate despre cum să realizez acest proces cât mai bine. Am început căutările pe la administratorii de cămine și în final am ajuns la domnul **Negrescu Radu, Administratorul șef**, care a fost de acord să mă îndrume și să îmi ofere toate informațiile care îmi sunt utile pentru a realiza aplicația.

Pentru realizarea obiectivului, am creat o aplicație de tip client-server pe care am dezvoltat-o pe parcursul mai multor luni, folosind variate unelte și limbaje de programare. Inițial, am creat codul ce a generat structura bazei de date, apoi am creat un set de date cu studenți fictivi care să respecte cât mai bine o situația reală din cadrul cazărilor. Am încercat să am nume românești de persoane, date cât mai reale iar numărul de studenți și alegerile să fie asemănătoare cu situațiile din anii anteriori. În cea mai mare parte a timpului am lucrat, în paralel, la construirea serverului și interfeței grafice.

Prin utilizarea unei multitudini de tehnologii moderne (spre exemplu Vue și .NET Core 2.1), respectarea bunelor practici atât în arhitectură cât și în implementarea aplicației și prin ușurința cu care aceasta poate fi extinsă, am reușit să creez o aplicație care să ajute studenții și chiar să îi provoace în a continua muncă începută de mine.

Pe tot parcursul dezvoltării aplicației am folosit cunoștințe dobândite de-a lungul anilor de facultate, în cadrul mai multor materii, și anume:

- **Introducere în .NET**: limbajul C# și principii generale

-**Topici Speciale .NET**: aprofundare C# și Entity Framework

- **Baze de date**: realizarea și managementul bazei de date

- **Ingineria programării:** realizarea diagramelor și optimizarea modului de lucru

- **Rețele de calculatoare**: protocolul HTTP

- **Tehnologii web**: realizarea unei aplicații web și principii REST

- **Programare orientată obiect**: principiile programării orientate obiect

2 Descrierea problemei

Atunci când un tânăr dorește să urmeze cursurile unei facultăți, de cele mai multe ori acea facultate sau cel puțin acea specializare de care este interesat nu se află în regiunea în care locuiește, de aceea el este nevoit să își schimbe domiciliul pentru a putea urma acele studii.

Fiind o problemă tot mai comună, oamenii s-au gândit să creeze căminele studențești, unde pot locui un număr mare de studenți, cu cheltuieli reduse, care asigură această nevoie pe parcursul anilor de studiu.

Însă cu trecerea timpului, procedura de ocupare a locurilor s-a schimbat datorită numărului tot mai mare de persoane care doresc să urmeze învățământul superior și a criteriilor tot mai diverse. Iar din această cauză a devenit destul de dificil pentru o singură persoană să se ocupe de acest proces.

Tehnologia a apărut pentru a ne simplifica viața. De aceea consider că și în acest caz ea ne poate fi de un mare folos, ușurând astfel munca celor responsabili de ocuparea locurilor cât și salvarea timpului și resurselor persoanelor care solicită acest serviciu.

3 Abordări anterioare

În acest moment nu există o aplicație care să fluidizeze și să gestioneze cazările în cadrul universității. Momentan tot acest proces se face pe hârtie fiind nevoie de drumuri repetate și de interacțiune cu alte persoane care sunt limitate de un program de lucru. Singura interacțiune indirectă din punct de vedere fizic este cea telefonică care de asemenea depinde de disponibilitatea persoanei de contact.

Inițial am dorit să fac acest sistem pentru întreaga universitate dar după câteva cercetări am ajuns la concluzia că este prea mult pentru o singură persoană și m-am rezumat la facultatea noastră.

Am decis să construiesc un mod nou prin care studenții să fie cazați cu scopul de a reduce cantitatea de timp irosit atât pentru administrație cât și pentru student, să reduc numărul de erori care pot surveni în acest proces și de asemenea să ofer o perspectivă în timp real asupra locurilor de cazare.

4 Descrierea soluției

4.1 Principalele funcționalități

Prima pagină care ia contact cu utilizatorul este alcătuită din cele mai recente anunțuri și articole postate de administrație. Acestea sunt niște previzualizări ale anunțurilor complete, ce conțin titlul, poza principală și primele fraze ale articolului. În partea de sus avem o bară de navigare la care au acces și utilizatorii străini, aceasta având următoarele rute:

- Articole: unde sunt previzualizările celor mai recente articole, aceasta fiind și pagina de întâmpinare a aplicației.
- Repartizări: unde sunt afișate rezultatele tururilor de cazare.
- Utile: aici sunt postate diverse documente sau regulamente care sunt utile în procesul de cazare.
- Contact: pagina ce conține informațiile de contact pentru cei care doresc să afle informații suplimentare sau vor să adreseze întrebări administrației.

Tot în bara de navigare avem și butonul de autentificare prin care utilizatorii pot să își acceseze conturile pentru a folosi aplicația. În *Figura 3* se poate viziona aspectul paginii de start cât și al barei de navigație.

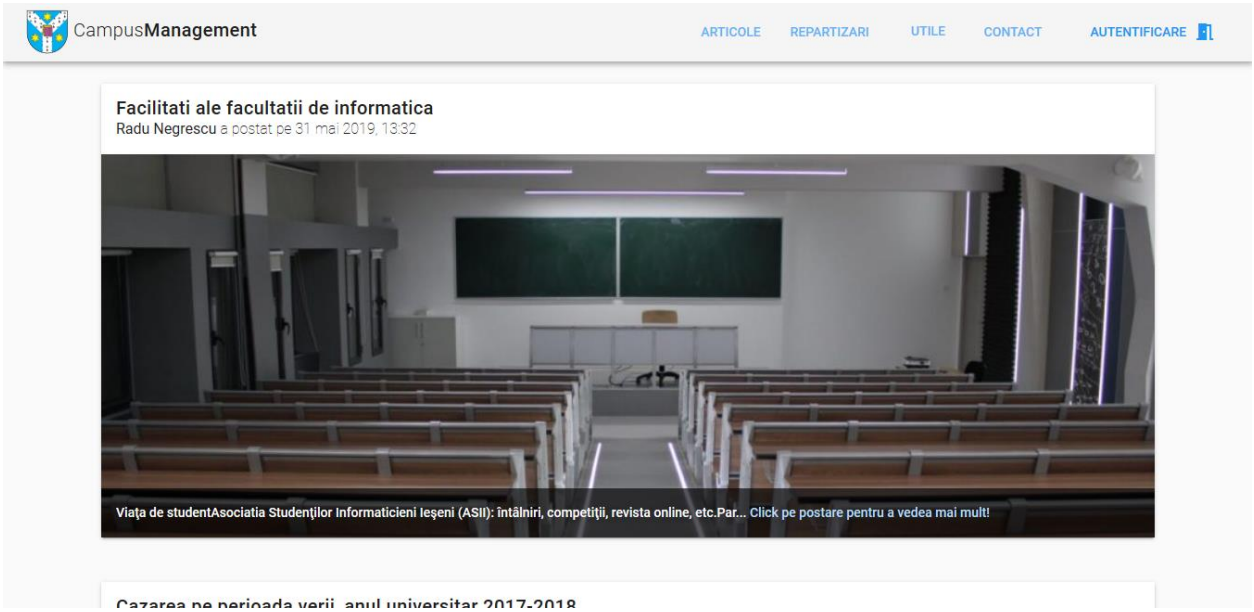

Figura 3: Pagina de start a aplicației

La apăsarea butonului de autentificare o fereastră va apărea în centrul ecranului și va solicita datele utilizatorului, în cazul nostru, email-ul de la facultate și parola. Acest formular are validări în timp real, în timp ce sunt completate datele, astfel utilizatorul poate vedea dacă respectă toate standardele de autentificare (email valid, parola din minim 6 caractere) *Figura 4*. Iar *Figura 5* arată că datele introduse sunt corecte sintactic.

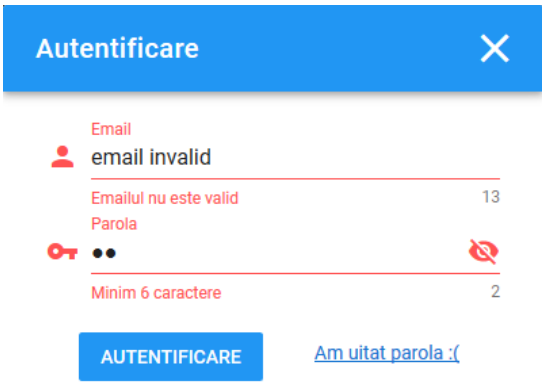

Figura 4: Date incorecte de autentificare

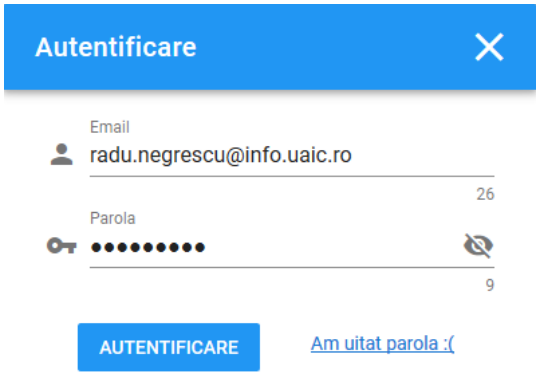

Figura 5: Date corecte de autentificare

După ce autentificarea a avut loc cu succes, în partea stângă va apărea un meniu unde sunt disponibile toate acțiunile pe care le poate efectua utilizatorul (student/admin).

Am să încep prima dată cu studentul pentru că în cazul lui lucrurile sunt destul de simple. El va avea în meniu butonul “Cerere cazare” unde va accesa un formular, iar în urma completării sale, studentul este înscris în primul tur de cazări. El are posibilitatea de a alege căminele pe care le preferă după cum se poate vedea în *Figura 6.*

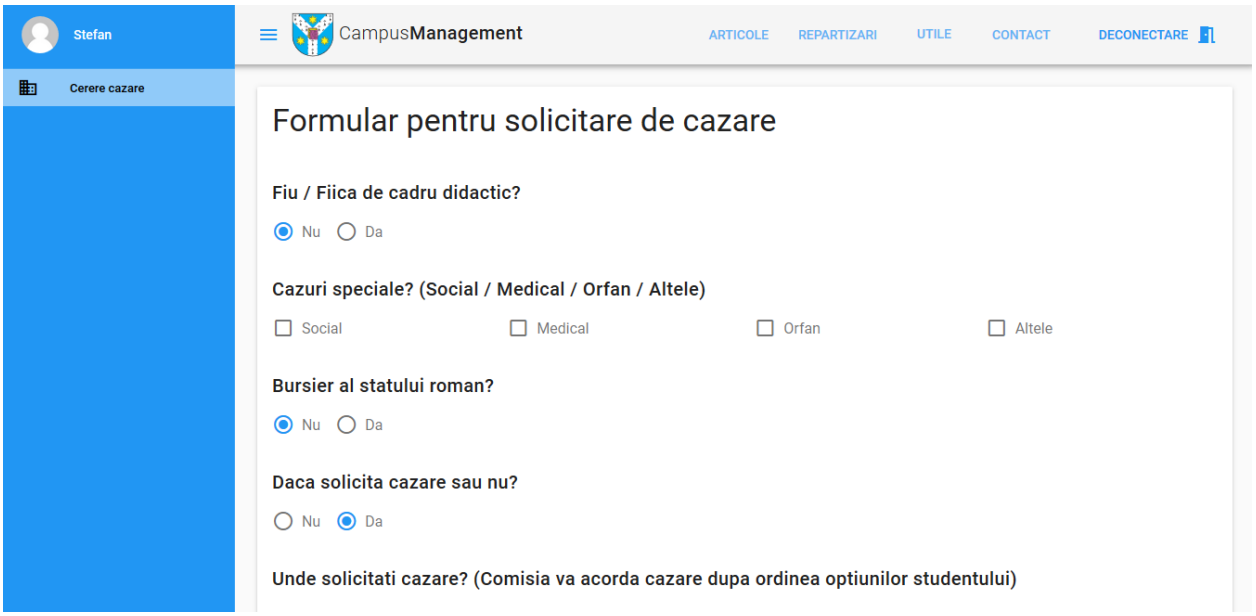

Figura 6: Formular pentru solicitare de cazare

Preferințele pot fi foarte ușor alese prin bifarea în ordine a priorităților după cum se poate vedea în *Figura 7.*

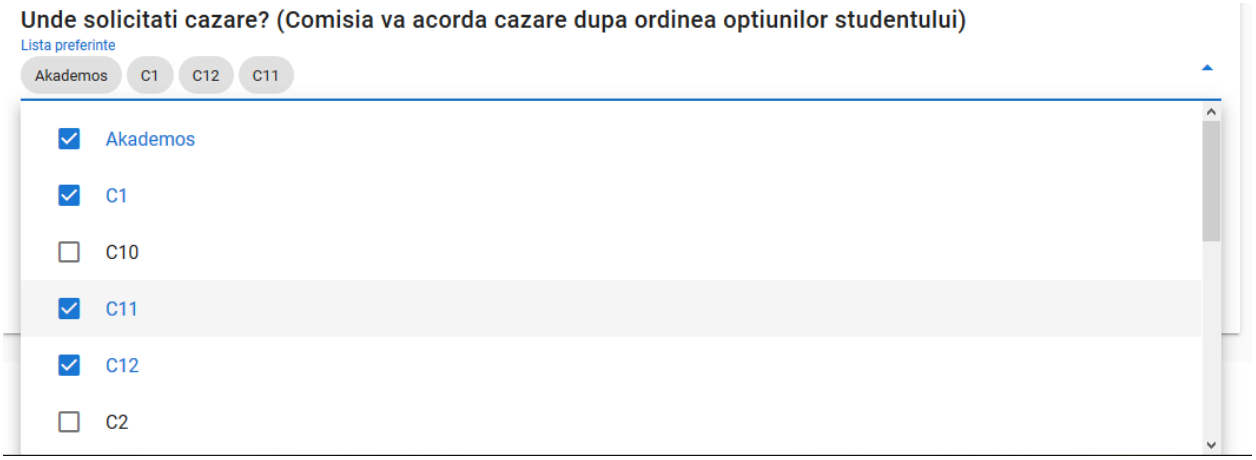

Figura 7: Alegerea preferințelor

După ce studentul a trimis cererea nu mai are nimic de făcut decât să aștepte informații de la administratorii de cazări. Dacă studentul reușește sa prindă loc într-un cămin, acesta va avea disponibilă în contul său o dispoziție de cazare precompletată cu toate datele sale, iar el va fi nevoit sa printeze documentul și să se prezinte la administratorul de cazări pentru a-și ocupa locul.

În cazul administratorului lucrurile stau puțin diferit. În meniu sunt prezente mai multe funcții pe care am să le prezint în continuare. Prima este pagina prin care poate adăuga articole. Interfața acestei pagini cât și toate componentele meniului de administrator se pot vedea în *Figura 8.*

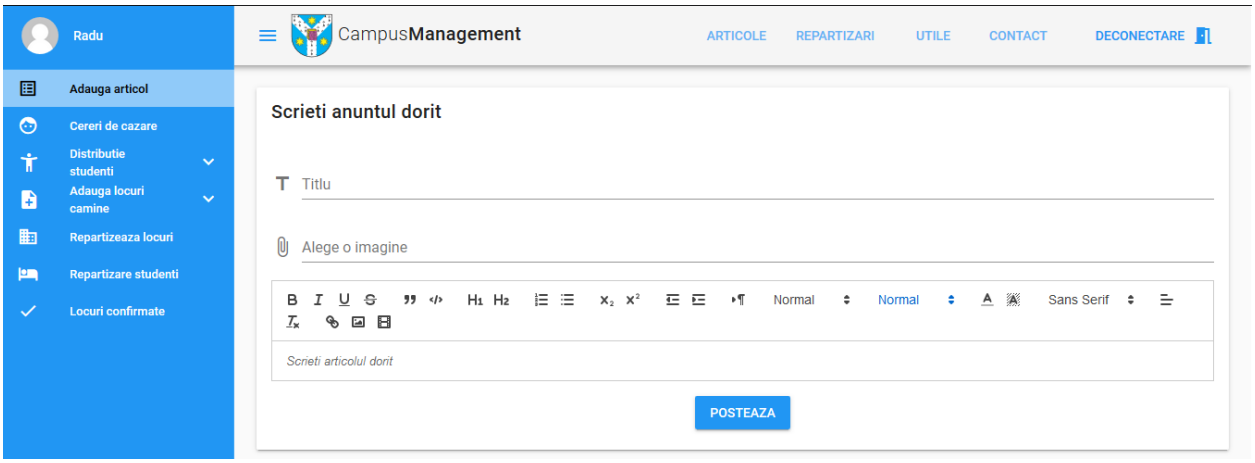

Figura 8: Adăugarea unui articol și componența meniului de administrator

Următoarea pagină conține toate cererile făcute de studenți. Aici se pot vizualiza punctajele lor, anul din care fac parte și ordinea preferințelor *Figura 9.* Studenții pot fi filtrați în funcție de anul de studiu și de sex sau pot fi căutați în mod explicit după nume.

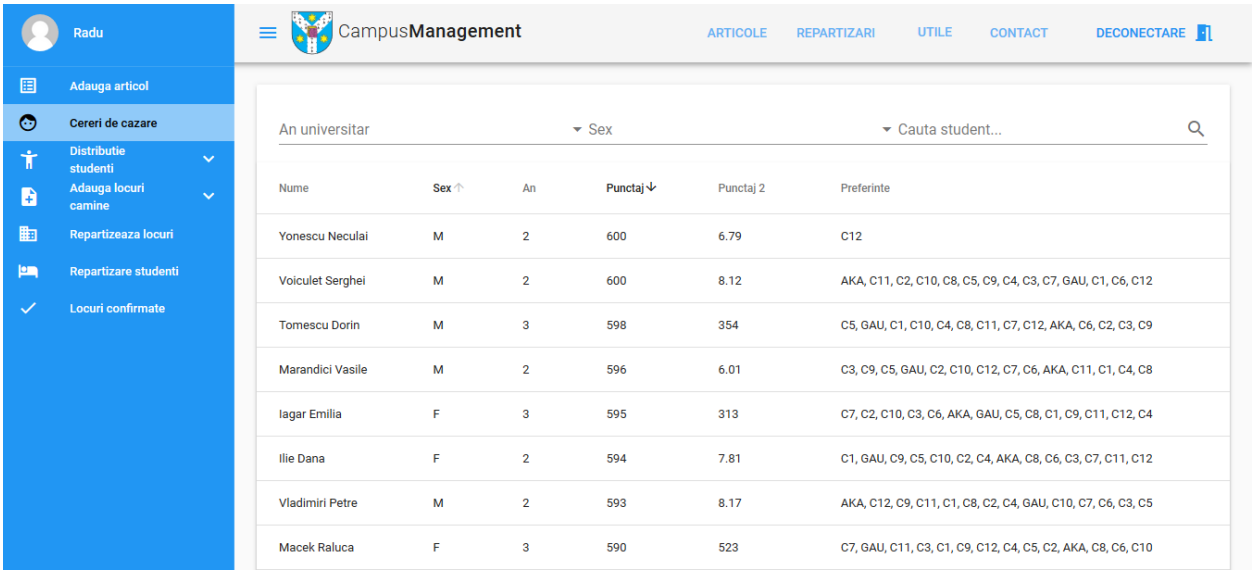

Figura 9: Pagina cu toate cererile de cazare

Pagina ce urmează conține distribuția studenților în funcție de numărul lor pe ani și sex dar și al numărului de cereri pe care l-au solicitat. Astfel aplicația calculează ce procentaj din locuri să fie alocat unui anumit grup de studenți. În *Figura 10* este prezentată pagina de distribuție pentru băieți.

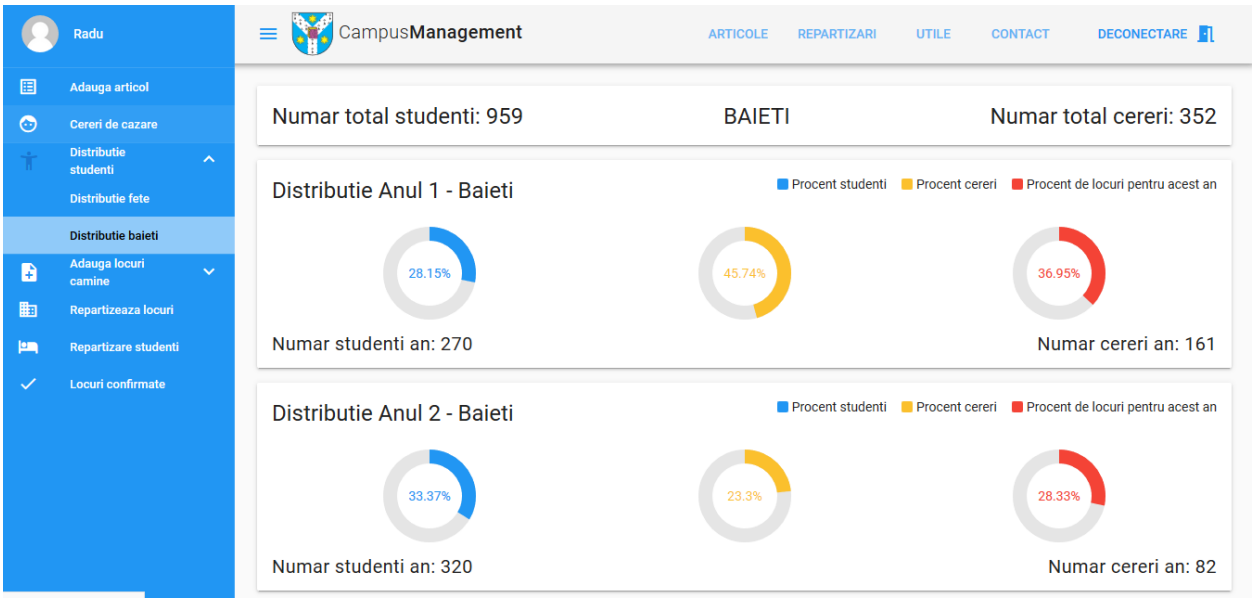

Figura 10: Distribuție băieți

Următoarea parte aplicației este reprezentată de adăugarea căminelor disponibile și a numărului de locuri. Este prezent un meniu unde bifează căminele, iar apoi va introduce pe rând locurile pe care acesta le-a primit. Aplicația va calcula, în mod automat, 10% din locuri pentru rezervă, iar

administratorul are posibilitatea de a regla acest procentaj sau de a edita manual după cum se poate vedea în *Figura 11*.

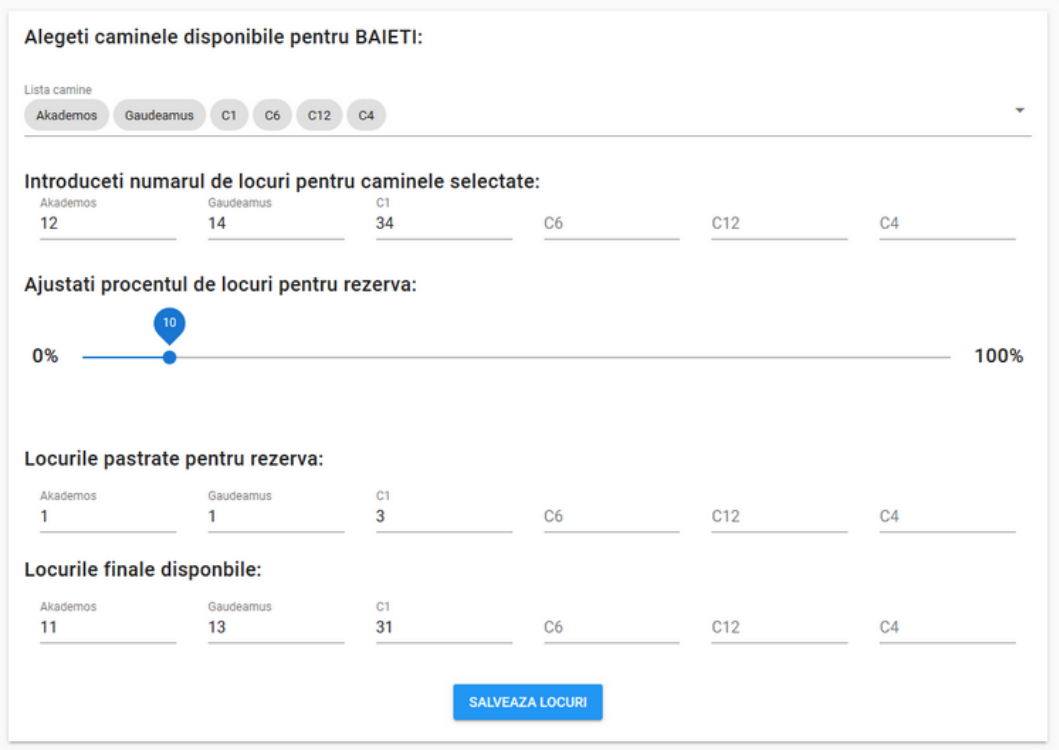

Figura 11: Formular pentru adăugarea de locuri în cămine

După ce au fost adăugate locurile, aplicația generează pe baza datelor prezente o repartizare a locurilor pentru fiecare an în parte, iar acest lucru se poate vizualiza și edita pe pagina ce urmează. Aici putem vedea numărul total de locuri pentru băieți sau fete din fiecare cămin și câte dintre ele au fost alocate pentru fiecare an. Un exemplu este prezent în *Figura 12*.

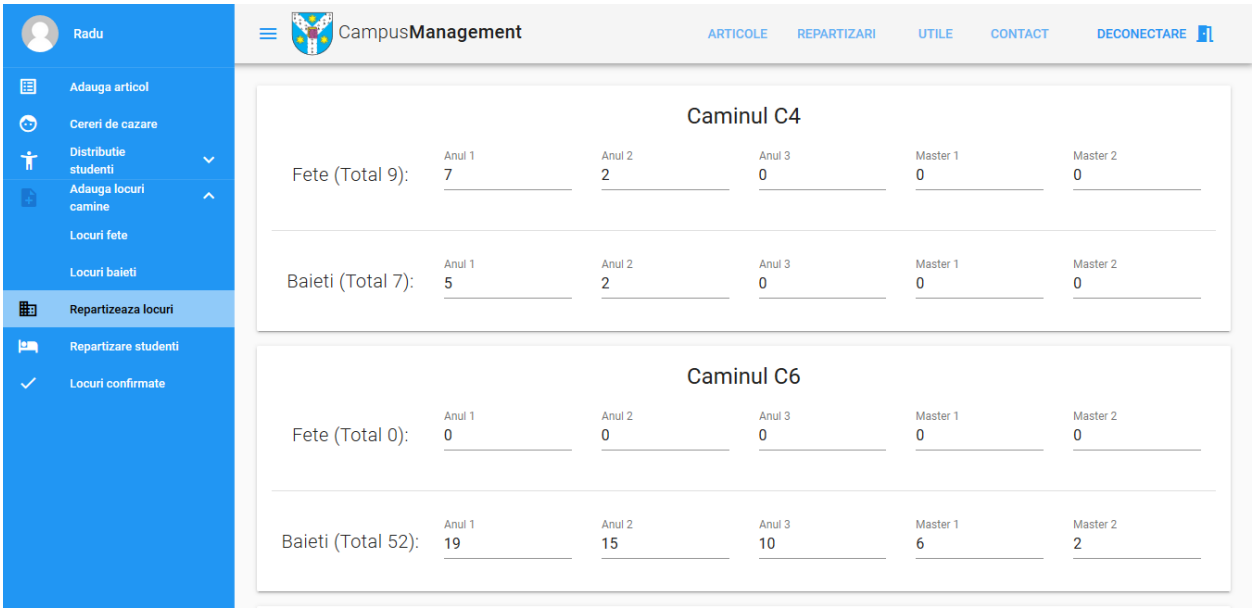

Figura 12: Repartizare locuri pe ani

Următoarea pagină conține rezultatele unui tur de cazare. Pentru a se putea genera această listă se va efectua click pe butonul "Generează Tur" unde administratorul va seta intervalul calendaristic în care s-au efectuat cereri, iar pe baza lui se va genera lista. După acestea, avem butonul "Postează Locurile" care se ocupă de publicarea rezultatelor pe pagina "Repartizări" și încă două butoane care descarcă rezultatele afișate în format Excel sau Pdf (*Figura 13*).

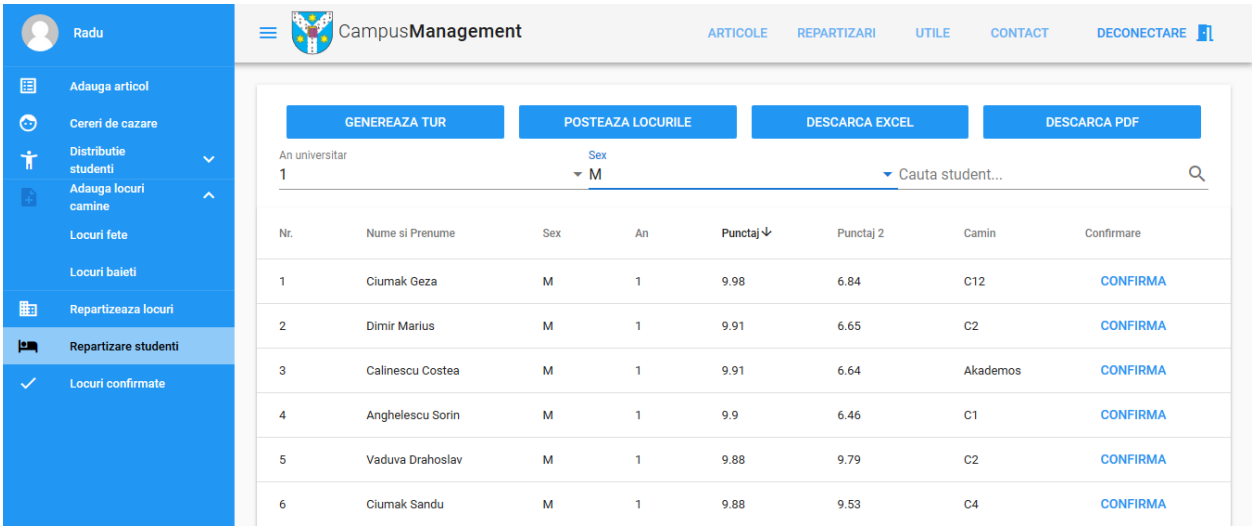

Figura 13: Repartizarea studenților

În momentul în care un student vine pentru a-și confirma locul, administratorul trebuie să apese pe butonul de confirmare din dreptul studentului, iar apoi să valideze confirmarea, cum este prezentat în *Figura 14*.

Figura 14: Confirmarea unui loc

După ce s-a finalizat acțiunea de confirmare, studentul este transferat pe lista celor confirmați. Aici sunt prezenți toți studenții care au acceptat locurile de cazare. Administratorul are libertatea de a decaza orice student în caz de nevoie, iar acel loc va fi realocat în viitoarele tururi de cazări. O imagine mai clară asupra acestei pagini se poate vedea in *Figura 15*.

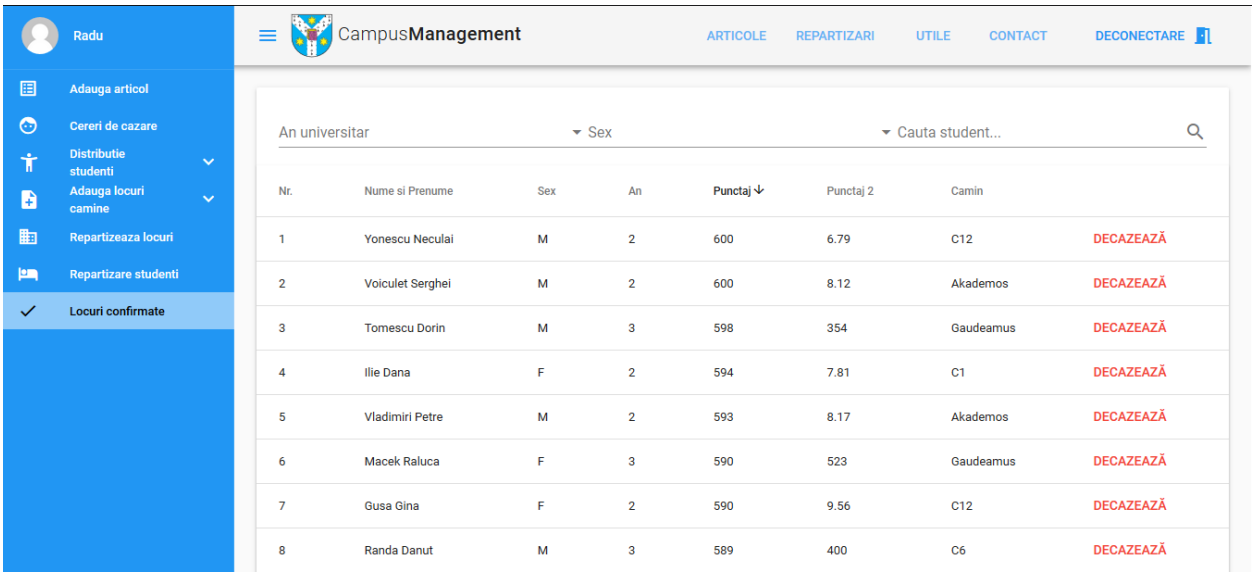

Figura 15: Lista locurilor confirmate

Acestea ar fi principalele funcționalități ale aplicației, fiind acoperite în mare parte nevoile existente în procesul de cazări. Sunt prezente și alte funcții însă am să las cititorul să le descopere.

4.2 Detalii de implementare

Pentru partea de server, aplicația a fost construită pe baza Arhitecturii Onion[8]. Această arhitectură împarte aplicația în mai multe nivele după cum se poate vedea în *Figura 16.*

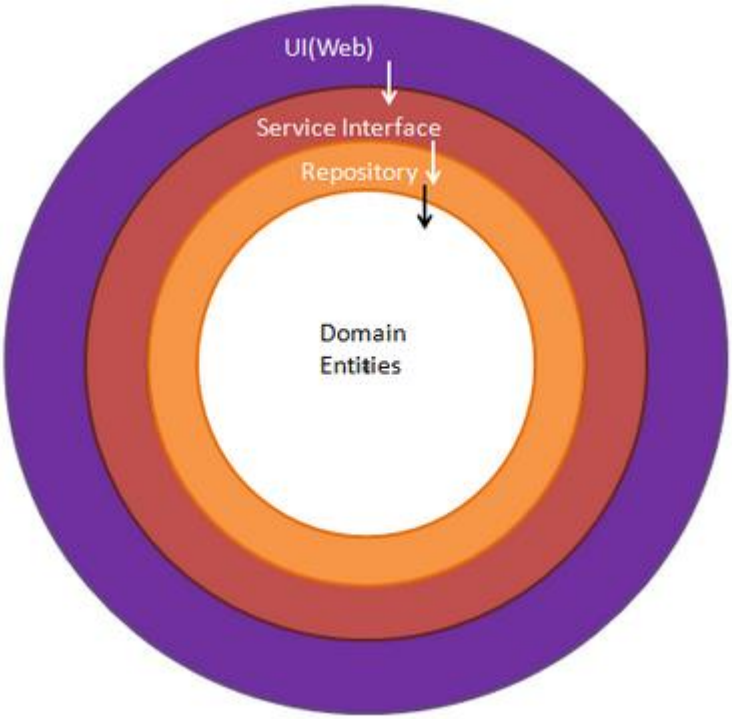

Figura 16: Nivelele modelului arhitectural Onion

[Sursa: https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/]

În primul nivel, adică cel din centru, sunt entitățile folosite în aplicație iar în cazul aplicației prezentate reprezintă și modelele care sunt salvate în baza de date, cum ar fi student, admin, articol etc.

[8] Athitectura Onion - https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/

Al doilea nivel este Repository[9] în care sunt implementate operațiile CRUD[10] asupra entităților și alte operații specifice pentru fiecare repository. Următorul nivel este cel de persistență în care sunt prezente legăturile cu baza de date, migrările pentru a putea gestiona ușor modificările aduse în timpul dezvoltării aplicației, iar ultimul nivel este reprezentat de punctele de acces, adică rutele unde un client face cereri pentru diverse informații.

Pentru a reduce nivelul de cod duplicat s-au folosit Generice[11]. A fost creat un repository generic care poate fi inițializat cu orice entitate iar acel repository se comportă ca și cum ar fi scris special pentru ea. Totodată există încă două servicii care implemtează acest concept: DetailsService care are doar operații destinate afișării datelor și CreateService care are ca scop modificarea datelor. Aceste două servicii necesită câte un model pentru a putea fi folosite. În cazul serviciului DetailsService avem nevoie de un Data Transfer Object, special creat pentru entitatea din baza de date, contractul interfeței se poate vedea în *Figura 17.* Aceste modele au rolul de a opri transmiterea de informații inutile cum ar fi transmiterea informațiilor despre un utilizator în care sunt prezente și date sensibile precum parola, CNP etc. Se folosesc obiecte special create pentru afișare, unde entitățile sunt alterate astfel încât în final modelul de afișare să conțină doar datele care se doresc a fi trimise. În acest mod oprim scurgerea de informații la care un utilizator nu ar trebui să aibă acces.

```
public interface IDetailsService<TEntityDetails> where TEntityDetails : class
{
    Task<TEntityDetails> GetAsync(Guid id, params string[] includes);
    Task<IEnumerable<TEntityDetails>> GetAllAsync(params string[] includes);
}
```

Figura 17: Interfață DetailsService

Iar pentru CreateService este nevoie de un model care să conțină informații care pot fi luate doar de la utilizator, informații pe care aplicația nu le poate genera singură. Interfața acestui serviciu se poate vedea în *Figura 18.*

[9] Repository - https://medium.com/falafel-software/implement-step-by-step-generic-repository-pattern-in-c-3422b6da43fd
[10] CRUD - https://en.wikipedia.org/wiki/Create,_read,_update_and_delete
[11] Generice - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/

```
public interface ICreateService<in TCreateEntity> where TCreateEntity : class
{
    Task<Guid> AddAsync(TCreateEntity entity);
    Task<IEnumerable<Guid>> AddAsync(IEnumerable<TCreateEntity> entities);

    Task<Guid> UpdateAsync(Guid id, TCreateEntity entity, params string[] includes);

    Task DeleteAsync(Guid id);
    Task DeleteAsync(IEnumerable<Guid> ids);
}
```

Figura 18: Interfață CreateService

Orice serviciu specific trebuie să moștenească DetailsService și CreateService. În felul acesta ne asigurăm că toate serviciile au implementate operațiile de bază și mai mult decât atât evităm rescrierea acestor operații pentru fiecare serviciu în parte, un lucru foarte util în cazul în care numărul lor este ridicat. Avem posibilitatea să adăugăm și alte operații specifice în caz de nevoie, în felul acesta blocăm accesul la modificare și creștem posibilitatea de a extinde.

De asemenea s-a folosit Design Pattern-ul Composite [12] care mai este și un înlocuitor al moștenirii. Avantajul acestui pattern vine ca soluție pentru imposibilitatea de moștenire multiplă în limbajul C#.

În *Figura 19* se poate vedea cum arată un serviciu pentru tururile de cazare ce folosește genericele și Design Pattern-ul Composite. Aici sunt prezentate și cele două tipuri de modele necesare pentru un serviciu. StageDetailsModel pentru DetailsService și StageCreateModel pentru CreateService.

[12] Composite - https://code-maze.com/composite/

```
public class StageService : IStageService
{
    private readonly IGenericRepository _genericRepository;
    private readonly IMapper _mapper;

    private readonly DetailsService<Domain.Entities.Stage, StageDetailsModel> _detailsService;
    private readonly CreateService<Domain.Entities.Stage, StageCreateModel> _createService;


    public StageService(IGenericRepository genericRepository, IMapper mapper)
    {
        _genericRepository = genericRepository;
        _mapper = mapper;

        _detailsService = new DetailsService<Domain.Entities.Stage, StageDetailsModel>
            (genericRepository, mapper);
        _createService = new CreateService<Domain.Entities.Stage, StageCreateModel>
            (genericRepository, mapper);
    }
```

Figura 19: Serviciul pentru tururile de cazare

Pentru a putea oferi implementări pentru interfețele ce sunt folosite la nivelurile superioare ale aplicației s-a folosit Dependency Injection[13], care este o metodă de a transmite implementări concrete pentru interfețe. Utilitatea acestei abordări constă în faptul că separă abstractizarea unui serviciu de implementarea sa. Mai exact se oferă doar un contract cu ceea ce se poate realiza cu ajutorul acelui serviciu fără a oferi vreo implementare iar atunci când aplicația este pornită, acel contract va primi o implementare.

Aplicația este construită pentru a rula în mod asincron. Pentru fiecare cerere primită se creează un nou fir de execuție care are ca scop îndeplinirea ei. În felul acesta toate apelurile la server sunt neblocante și permit astfel servirea mai multor utilizatori în același timp.

La nivelul de securitate s-a folosit autentificarea și autorizarea pe bază de tokeni. Aplicația fiind un REST API, aceasta nu are sesiuni pentru a putea ține conectat un utilizator, asta înseamnă că fiecare cerere este văzută ca și cum ar fi prima. Totuși pentru a putea ține evidența și de a controla accesul la resurse s-a decis folosirea acestei abordări unde utilizatorul oferă informațiile sale de autentificare iar serverul generează un JWT[14] care îi oferă accesul temporar la anumite servicii. Tokenul conține id-ul solicitantului pentru a putea determina cine face cererile, rolul sau rolurile pe care acesta le are în funcție de tipul de cont, în cazul nostru administrator sau student. În cazul în care utilizatorul nu mai accesează aplicația o perioadă de timp, el va fi

[13] Dependency Injection - https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.2
[14] JWT - https://medium.com/@mmoshikoo/jwt-authentication-using-c-54e0c71f21b0

deconectat automat pe baza unei valori din token care reprezintă valabilitatea lui. Pentru a fi sigură metoda prezentată, s-a folosit criptarea HS256[15] asupra tokenului cu o cheie de dimensiuni mari.

Utilizatorul va avea în antetul cererii tokenul iar acesta este verificat înainte de a efectua orice operație de pe server folosind un Middleware[16]. Acest middleware interceptează cererile înainte să ajungă la controller și extrage tokenul din cerere pentru a verifica datele pe care le conține și valabilitatea lui. Daca tokenul este valid atunci i se va onora cererea, în caz contrar returnează codul HTTP 401 care simbolizează accesul oprit. *Figura 20* ilustrează cum este securizată o cale de acces unde doar administratorii pot apela serviciul. În cazul acesta ștergerea unui articol postat.

```
[Authorize(Roles = "Admin")]
[HttpDelete("{id}")]
public async Task<IActionResult> Delete(Guid id)
{
    await _articleService.DeleteAsync(id);
    return NoContent();
}
```

Figura 20: Middleware-ul de autorizare

4.3 Diagrame

4.3.1 Diagrame pentru cazuri de utilizare

Fiecare utilizator trebuie să fie înregistrat în aplicație înainte de a putea face orice fel de acțiune, cu excepția câtorva zone unde accesul este permis tuturor. După cum s-a spus anterior, în cadrul aplicației există două tipuri de utilizatori: student și administrator. Ambii pot folosii mai multe tipuri de acțiuni astfel interogând sau modificând baza de date.

[15] HS256 - https://stackoverflow.com/questions/39239051/rs256-vs-hs256-whats-the-difference
[16] Midleware - https://en.wikipedia.org/wiki/Middleware

Toate tipurile de utilizatori pot vizualiza diverse tipuri de liste sau obiecte și pot efectua schimbări asupra elementelor acestora. Toate acțiunile pe care le pot săvârși aceștia sunt explicate cu ajutorul unor diagrame în figurile de mai jos, mai exact cele din intervalul *Figura 21 – Figura 24.*

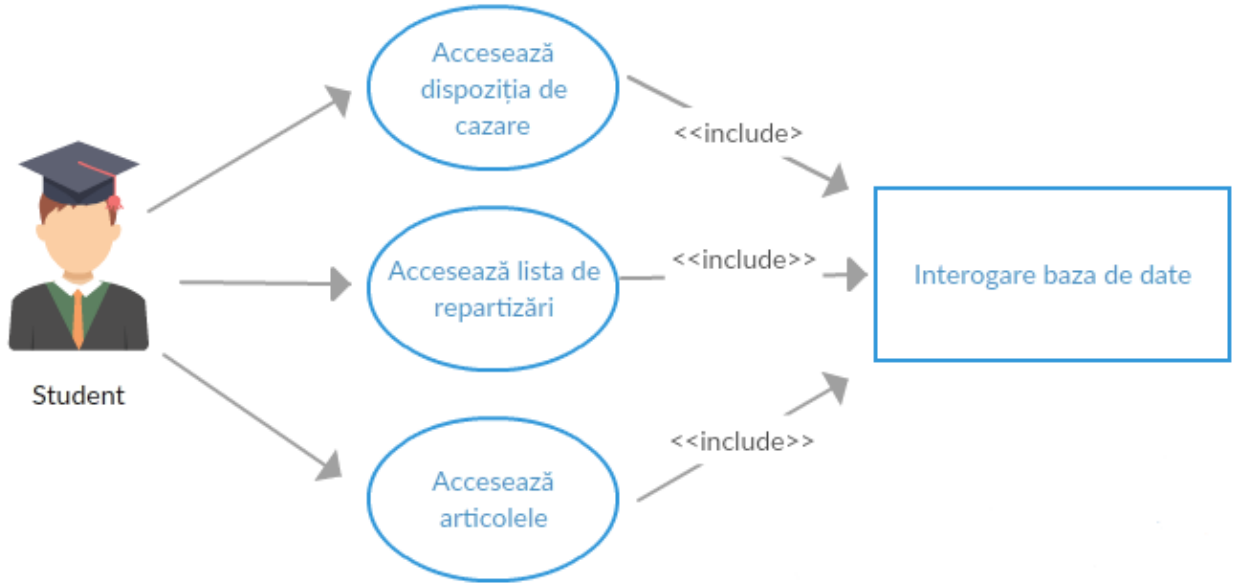

Figura 21: Acțiunile care generează interogarea bazei de date pentru studenți

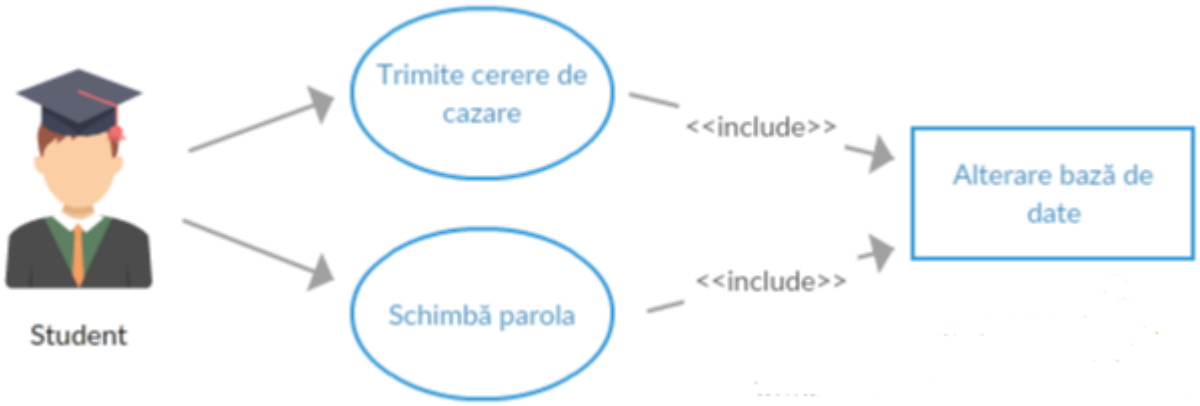

Figura 22: Acțiunile care generează alterarea bazei de date pentru studenți

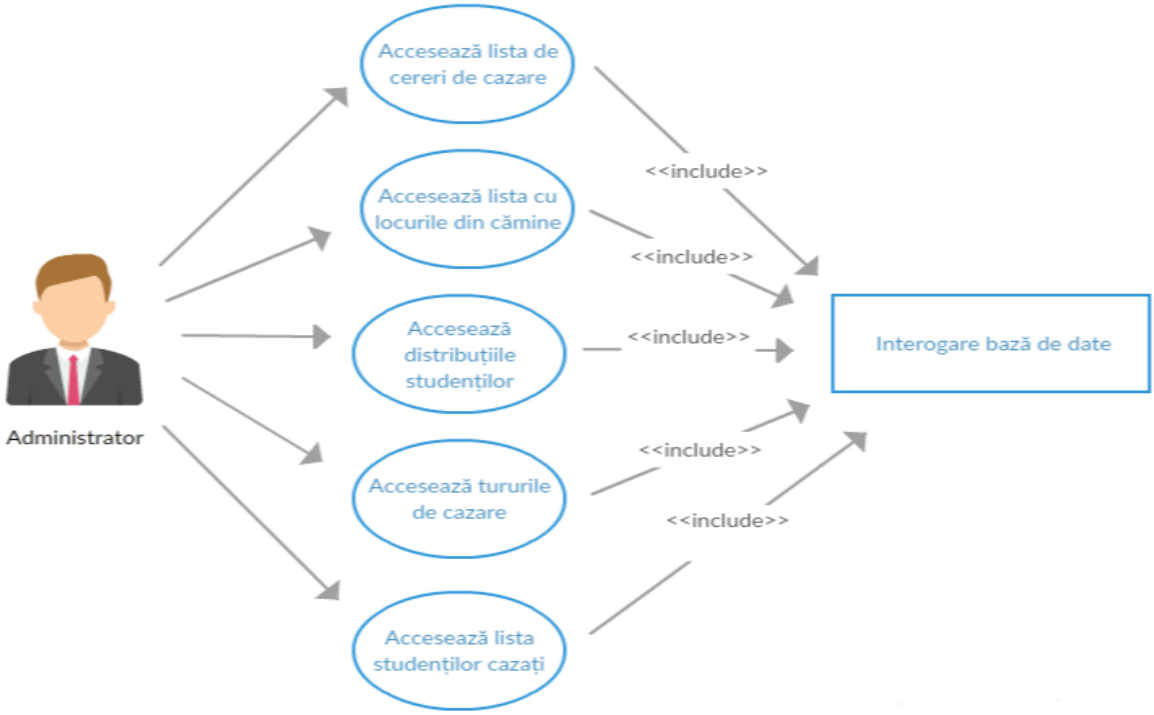

Figura 23: Acțiunile care generează interogarea bazei de date pentru administratori

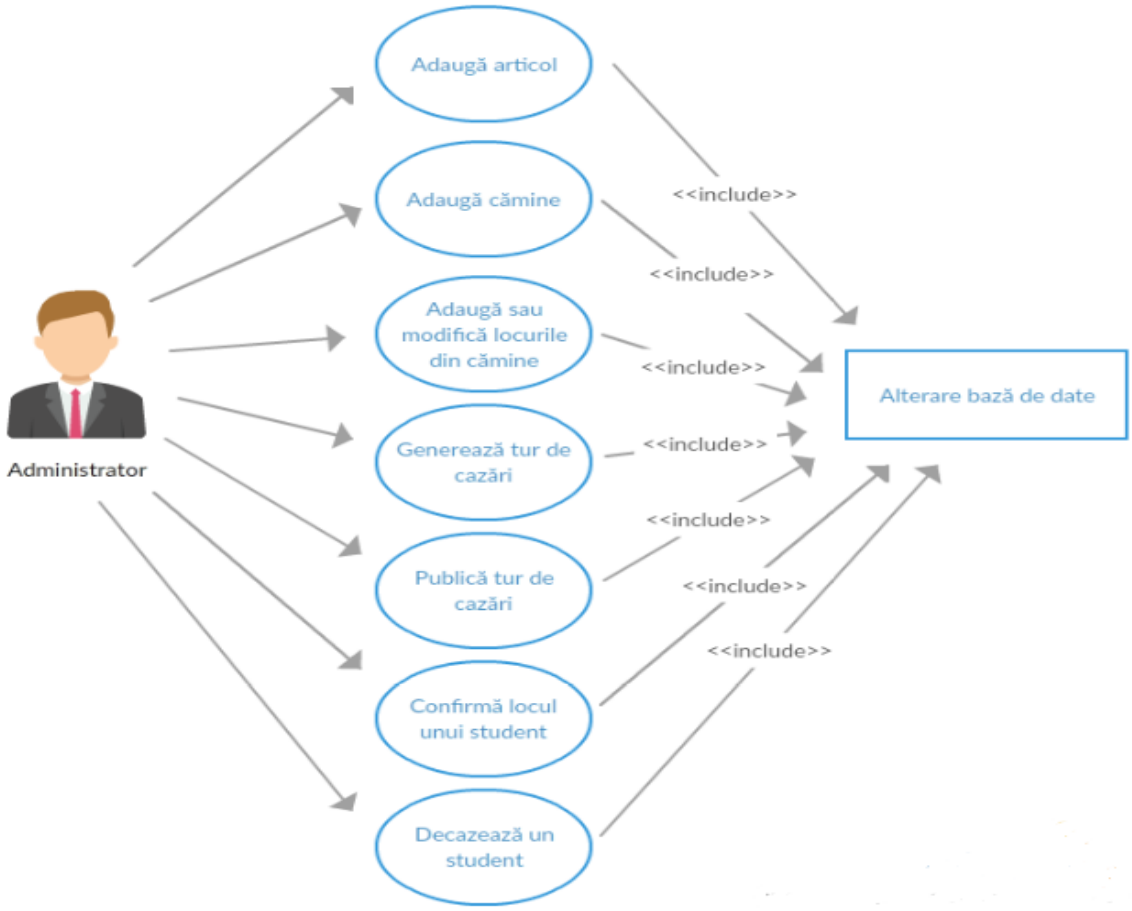

Figura 24: Acțiunile care generează alterarea bazei de date pentru administratori

4.3.2 Diagrame arhitecturale

Mai jos este diagrama care prezintă cum este construit un serviciu specific pe baza genericelor. *Figura 25.*

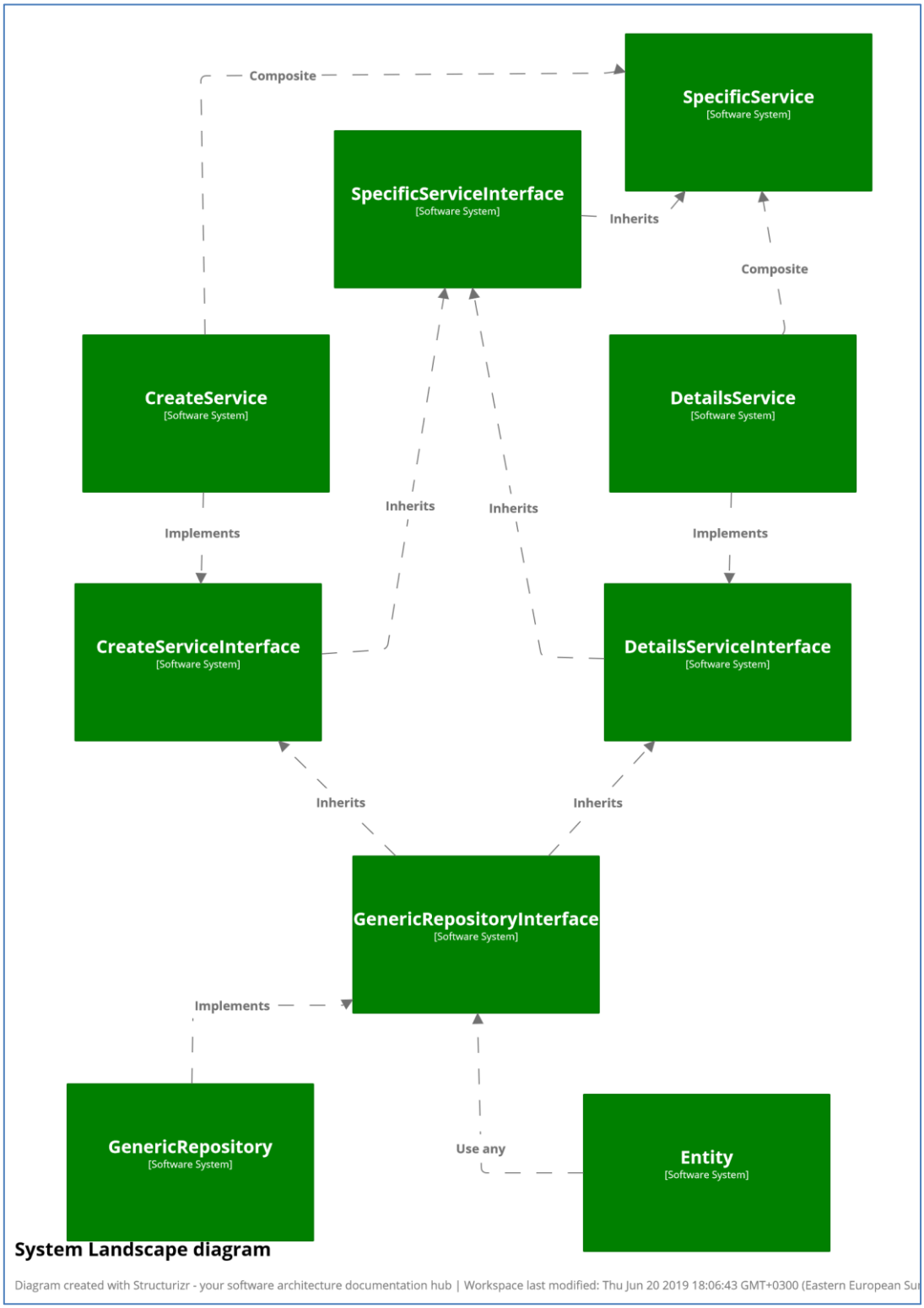

Figura 25: Diagrama unui serviciu specific pe baza genericelor

În *Figura 26* avem diagrama bazei de date împreună cu toate relațiile dintre tabele.

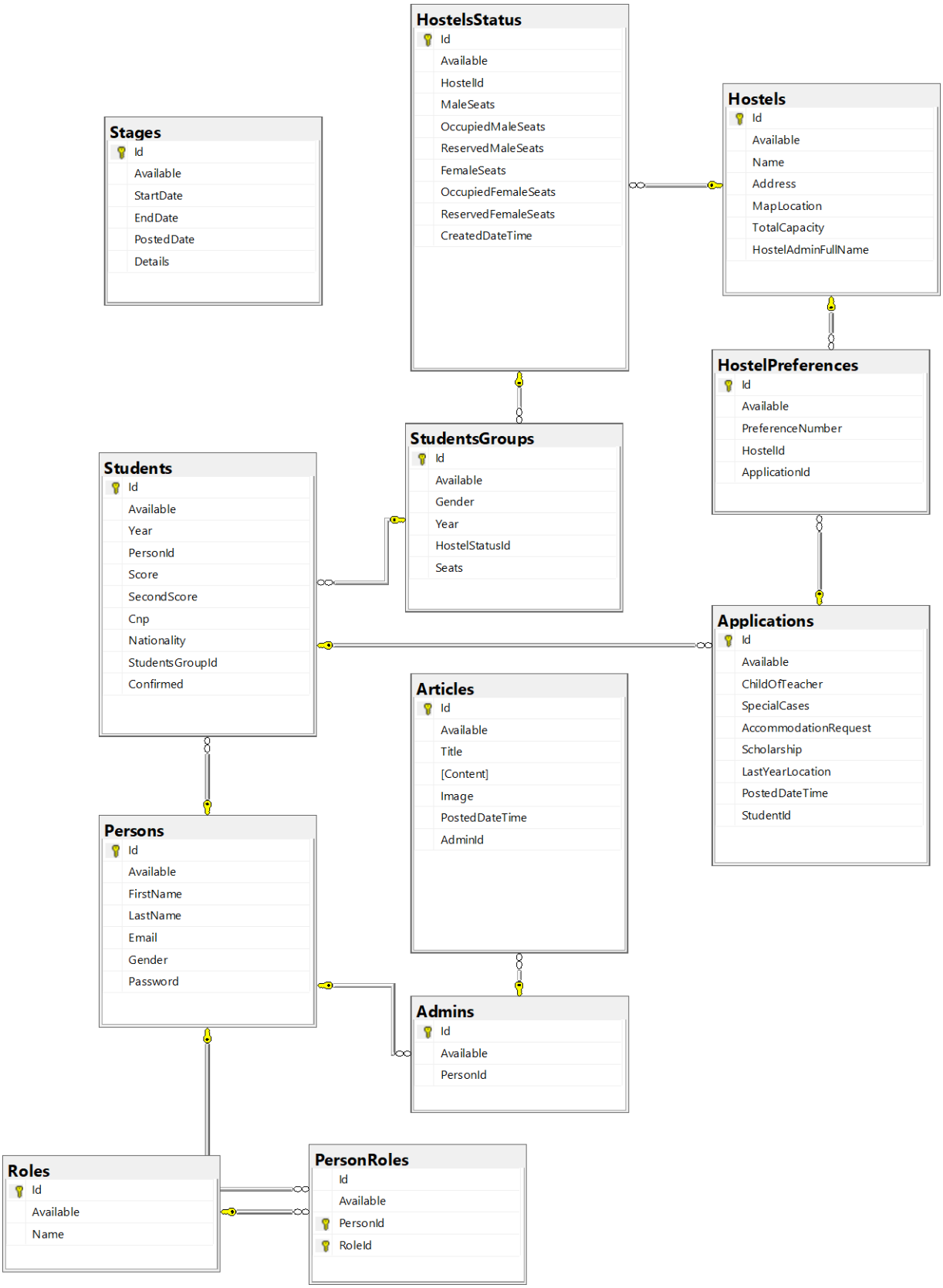

Figura 26: Diagrama bazei de date

Entități

Persons – este tabela care definește orice utilizator care intră în contact cu aplicația și aparține Facultății de Informatică din Iași. Fiecare utilizator are un email, o parola, un rol sau mai multe și alte informații adiționale. Acele roluri sunt pentru verificarea tipului de utilizator înainte de fiecare operație.

Roles – este tabela unde sunt înregistrate toate tipurile de roluri din aplicație, în cazul nostru student și administrator.

PersonRoles – este tabela care face legătura dintre persoane și rolurile pe care le dețin.

Students – este tabela care definește studenții care sunt înregistrați în aplicație. Aceștia au informații ca sexul, cnp, punctaje, an de studiu, naționalitate și desigur dacă este cazat sau nu.

HostelPreferences – este tabela care reține preferințele studenților legate de ordinea în care vor să primească loc în cămine.

Admins – este tabela care definește administratorii de cazări. Aceștia nu dețin informații suplimentare față de Person.

Articles – este tabela care definește articolele. Această tabelă conține detalii ca titlul articolului, calea unei imagini, data postării, autorul ei și conținutul.

Applications – este tabela care definește cererile de cazare. Aici sunt stocate toate cererile studenților cu informații precum: ordinea preferințelor, cazuri speciale, locație anterioară.

Hostels – este tabela care definește căminele disponibile. Conține informații generale ca numele căminului, locația, numele administratorului, numărul maxim de locuri și coordonatele pe hartă.

HostelsStatus – este tabela asociată unui cămin care conține date despre starea acestuia. Informații precum numărul de locuri pentru fiecare an și sex în parte, câte locuri sunt ocupate, câte mai sunt disponibile și data de când este starea.

Stages – este tabela care definește tururile de cazare. Aici sunt stocate date ca perioada calendaristică pentru care sunt luate în considerare cererile și eventuale informații adiționale ca durata lui.

Legături între entități

Roles – PersonRoles – Person: este o relație de mai multe la mai multe unde o persoană poate avea mai multe roluri iar un rol poate să aparțină mai multor persoane.

Persons – Students: este o relație de tip unul la unul. Un student conține toate informațiile unei persoane, pentru că din punct de vedere conceptual el este o persoană.

Admins – Persons: este o relație de unul la unul la fel ca la student și are toate informațiile pe care o persoană le poate oferi.

Students – Applications: este o relație de unul la mai multe. Un student poate avea mai multe cereri de cazare în perioade de timp diferite însă o cerere poate aparține doar unui singur student.

Applications – HostelPreferences: este o relație de unul la mai mulți. O cerere de cazare are o listă de preferințe, deci mai multe preferințe dar o preferință poate aparține doar unei cereri de cazare.

HostelPreferences – Hostels: este o relație de unul la unul unde o preferință pentru cămin are legatură cu căminul în sine.

Hostels – HostelsStatus: este o relație de unul la unul. Un cămin poate avea doar o stare la un moment de timp și o stare poate aparține doar unui cămin.

HostelsStatus – StudentsGroups: este o relație de unul la mai mulți. O stare de cămin poate avea mai multe grupuri de studenți (un grup înseamnă combinația: an universitar, sex și căminul în care se află) dar un grup de studenți poate fi doar într-un cămin.

Admins – Articles: este o relație de unul la mai mulți. Un administrator poate avea mai multe articole postate însă un articol poate avea doar un administrator ca și autor.

4.3.3 Diagrama structurii soluției

Soluția este structurată pe mai multe nivele pentru a asigura un mod optim de a face gestiunea și mentenanța codului sursă, fapt vizibil în *Figura 27* și în diagrama din *Figura 28.*

Fiecare nivel este reprezentat de un proiect care, la rândul lui este împărțit în mai multe dosare și subdosare pentru a permite localizarea rapidă și intuitivă a fișierelor.

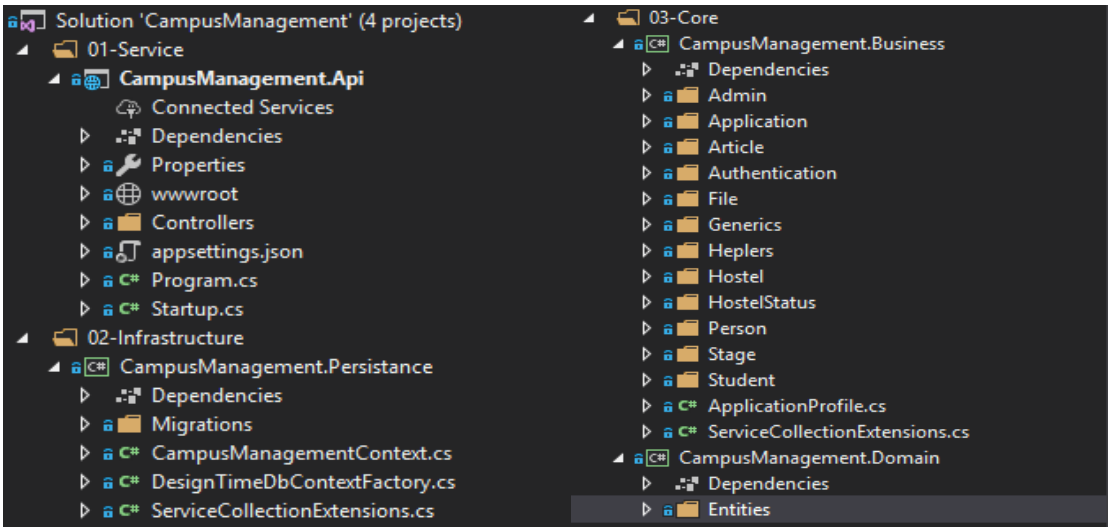

Figura 27: Structura soluției din Visual Studio

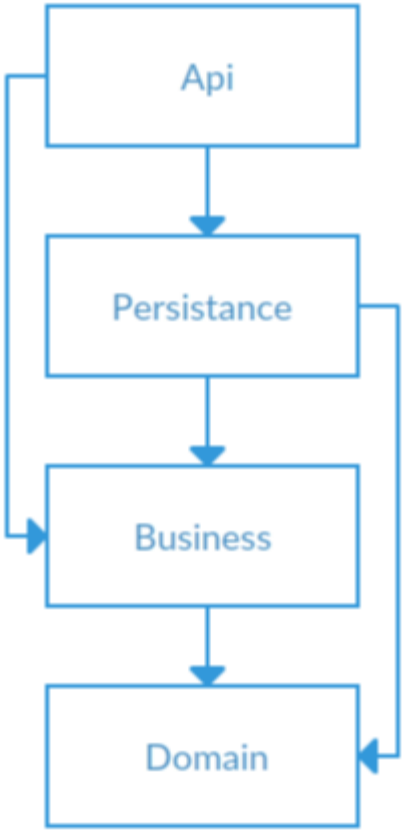

Figura 28: Structura soluției (proiecte și dependințe)

Api – acesta este nivelul destinat API-ului REST. Conține mai multe Controllere, adică clase care expun servicii către client și care pot fi accesate cu protocolul HTTP. Aici ajung toate injectările de servicii de la nivelele inferioare și tot aici este prezent și middleware-ul pentru autentificare care se ocupă cu validarea cererilor. Tokenul din cereri este verificat la acest nivel.

Persistance - scopul acestui nivel este de a lega logica aplicației cu baza de date, adică aici se face conversia din obiecte în tabele pentru baza de date. Tot aici este prezent sistemul de migrări care este ca un istoric pentru toate modificările ce au fost aduse în timpul dezvoltării aplicației, iar aceste modificări se efectuează și pe baza de date pentru a păstra consistența aplicației. În felul acesta utilizatorul nu mai are două responsabilități, una pentru cod si cealaltă pentru baza de date, ci doar aceea pentru cod. Entity Framework Core este folosit la acest nivel.

Business – la acest nivel este prezentă toată logica aplicației care răspunde la cererile clientului. Aici sunt prezente mai multe servicii împărțite pe concepte (student, articol, tur, etc.) care la rândul lor au în componență modelele necesare și validările.

Domain – este nivelul responsabil cu entitățile din aplicație care apar și în baza de date sub formă de tabele. Aici sunt toate informațiile necesare pentru client și legăturile dintre entități. Există o entitate simplă pe baza căreia toate celelalte o moștenesc. În felul acesta avem o legătură între toate entitățile pentru a putea fi folosite genericele.

Concluziile lucrării

Prezenta lucrare de licență demonstrează faptul că prin folosirea unei suite bine aleasă de tehnologii și instrumente software se poate realiza o aplicație web, simplă din punct de vedere al mentenanței și extinderii dar cu o utilitate foarte mare în automatizarea anumitor procese în beneficiul societății.

Ținând cont de faptul că tehnologia se află într-o evoluție continuă, cel mai benefic pentru noi devine faptul că o putem utiliza din ce în ce mai des în viața noastră pentru a ne simplifica și ușura totodată viața.

Așadar, aplicația **"Managementul cazărilor în cămine"** vine în sprijinul atât al viitorilor studenți care vor dori să locuiască în căminele de la facultatea noastră, cât și pentru administratorii care se ocupă de acest proces complex și costisitor din punct de vedere al timpului. Aceasta reușește să ajute toți utilizatorii să economisească timp prețios, resurse materiale și cel mai important favorizează amintirile plăcute legate de acest proces. Mai mult decât atât, încurajează studenții sa aleagă căminul ca domiciliu pe perioada studenției, iar în felul acesta, ei au șansa de a descoperi persoane extraordinare și experiențe de neuitat.

Versiunea curentă a aplicației poate fi îmbunătățită în mai multe direcții:

- să se adauge tratarea cazurilor speciale, cum ar fi studenții cu cetățenie moldovenească din anul 1 care trebuie să primească locuri în mod obligatoriu;
- se pot adăuga alerte prin SMS pentru studenții care au reușit să prindă un loc;
- aplicația să fie folosită pe întreaga universitate, să se poată transfera locuri rapid de la o facultate la alta în caz că acestea rămân neocupate, astfel se poate ține o evidență mult mai clară asupra situațiilor de cazare.

Bibliografie

1. Principiile REST - https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-2.2&tabs=visual-studio
2. HTTP - https://www.c-sharpcorner.com/UploadFile/dacca2/http-request-methods-get-post-put-and-delete/
3. Arhitectura Onion - https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/
4. Genericele - https://stackoverflow.com/questions/43843929/use-of-generic-types-in-dotnetcore
5. ASP.NET Core - https://www.codemag.com/Article/1807041/What%E2%80%99s-New-in-ASP.NET-Core-2.1
6. Entity Framework Core - https://www.pluralsight.com/courses/entity-framework-core-2-getting-started
7. FluentValidation - https://fluentvalidation.net/
8. Stucurarea in Repository - https://code-maze.com/net-core-web-development-part4/
9. Vue - https://vuejs.org/
10. Vuetify - https://vuetifyjs.com/
11. Quill Editor - https://quilljs.com/
12. JWT in .NET Core - https://garywoodfine.com/asp-net-core-2-2-jwt-authentication-tutorial/
13. Vuex - https://www.youtube.com/watch?v=BGAu__J4xoc&list=PL4cUxeGkcC9i371QO_Rtkl26MwtiJ30P2
14. Axios - https://github.com/axios/axios
15. Json to Excel - https://docs.sheetjs.com/
16. jsPDF - https://parall.ax/products/jspdf