

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**MEDDIS - APLICAȚIA DE REPARTIZARE A
MEDICILOR REZIDENTI LA SPITALE**

propusă de

Tiberiu-Gabriel Bădeliță

Sesiunea: Iunie/Iulie, 2023

Coordonator științific

Lect. Dr. Cristian Andrei Frăsinaru

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA

**MEDDIS - APLICATIA DE
REPARTIZARE A MEDICILOR
REZIDENTI LA SPITALE**

Tiberiu-Gabriel Bădeleță

Sesiunea: Iunie/Iulie, 2023

Coordonator științific

Lect. Dr. Cristian Andrei Frăsinaru

Avizat,

Îndrumător lucrare de licență,

Lect. Dr. Cristian Andrei Frăsinaru.

Data: Semnătura:

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Bădețiă Tiberiu-Gabriel** domiciliat în **România, jud. Suceava, localitatea Rădăuți, str. Mihai Viteazu, nr.26A, bl. 24**, născut la data de **04 octombrie 2001**, identificat prin CNP **5011004330523**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2023, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **MEDDIS - APlicația de Repartizare a Medicilor Rezidenți la Spitale** elaborată sub îndrumarea domnului **Lect. Dr. Cristian Andrei Frăsinaru**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consumând inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe propria răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data:

Semnătura:

Declarație de consumămant

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **MEDDIS - APLICAȚIA DE REPARTIZARE A MEDICILOR REZIDENTI LA SPITALE**, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Tiberiu-Gabriel Bădețiă**

Data:

Semnătura:

Cuprins

| | |
|--|-----------|
| Introducere | 2 |
| 1 Specificații funcționale | 4 |
| 1.1 Problema studiată | 4 |
| 1.2 Fluxuri de lucru | 4 |
| 2 Arhitectura sistemului | 6 |
| 2.1 Layered Architecture | 6 |
| 2.1.1 Domain Layer | 7 |
| 2.1.2 DataAccess Layer | 7 |
| 2.1.3 Application Layer | 8 |
| 2.1.4 API Layer | 8 |
| 2.2 Architectural Diagram | 9 |
| 3 Implementare și testare | 10 |
| 3.1 Tehnologii Folosite | 10 |
| 3.1.1 React.js | 10 |
| 3.1.2 .NET | 11 |
| 3.1.3 Microsoft SQL SERVER | 16 |
| 3.2 Algoritmi utilizati | 16 |
| 3.2.1 Algoritmul 1 | 16 |
| 3.2.2 Algoritmul 2 | 18 |
| 4 Manual de utilizare | 20 |
| 4.1 Pagina de logare | 20 |
| 4.2 Pagina de înregistrare | 21 |
| 4.3 Pagina principală | 22 |
| 4.4 Pagina de completare a preferințelor | 24 |
| 4.5 Pagina de vizualizare a repartizărilor | 25 |
| 4.6 Pagina de vizualizare a contului | 26 |
| Concluzii și direcții viitoare | 28 |
| Bibliografie | 29 |

Introducere

După terminarea programului de rezidențiat , fiecare medic rezident dorește să fie repartizat la spitalul cel mai plăcut preferințelor lui. De asemenea, fiecare spital dorește medici cât mai bine pregătiți pe fiecare specializare la care sunt locuri disponibile. Deși pare o provocare simplă la o primă vedere, repartizarea medicilor la spitale astfel încât ambele părți să fie mulțumite devine o provocare dificil de rezolvat. Pe măsură ce numărul de medici, respectivii spitale crește, totodată gama de preferințe crește, astfel ridicând și complexitatea procesului de asignare a unui medic la un spital. Drept urmare, lucrarea de licență pe care am propus-o vine în sprijinul rezolvării acestei probleme care există în lumea reală.

Lucrarea de licență propusă de mine este o aplicație web, o aplicație de repartizare a medicilor rezidenți la spitale, care simplifică și în același timp optimizează transmiterea de preferințe a medicilor, transmiterea nevoilor de medici pe anumite specializări ale spitalelor și ajută la distribuirea automată a medicilor rezidenți pe locurile disponibile speciaților lor la spitalele dorite. Totodată design-ul prietenos al aplicației facilitează procesul pentru care este creată aplicația. Îmbinarea domeniului IT cu domeniul medical este o abordare inovatoare ce se regăsește tot mai des în ziua de astăzi, ce vine cu beneficii pentru domeniul medical, sporind calitatea serviciilor medicale și calitatea îngrijirii pacienților.

În urma colaborării acestor două domenii au rezultat și aplicații ce au servicii de repartizare a medicilor rezidenți în spitale. Cele mai cunoscute aplicații similare cu aplicația propusă de mine sunt: Oriel, National Resident Matching Program (NRMP) și Canadian Resident Matching Service (CRMS). Aceste programe de repartizare folosesc algoritmi compexi care iau în considerare preferințele medicilor și posturile disponibile în spitale pentru a optimiza procesul de atribuire. Acestea urmăresc să asigure o repartizare corectă și eficientă a rezidenților către spitale pe baza unui sistem de clasare reciprocă.

În cadrul aplicației web propusă de mine ca și lucrare de licență, tehnologiile folosite sunt:

- client-side: React.js Library;
- server-side: .NET Framework;
- database: Microsoft SQL Server.

Rolul acestora în cadrul acestui proiect va fi detaliat în capitolele ce urmează. Structura tezei scrise conține 4 capitole principale urmate de concluzii și directii viitoare. Cele 4 capitole se referă la:

Capitolul 1. Specificații funcționale prezintă problema abordată și rezolvată în cadrul lucrării de licență și fluxuri de lucru.

Capitolul 2. Arhitectura sistemului descrie în detaliu arhitectura din spatele aplicației de repartizare.

Capitolul 3. Implementare și testare cuprinde atât tehnologiile folosite alături de exemple reprezentative de cod din cadrul acestui proiect, cât și partea algoritmică a acestuia.

Capitolul 4. Manual de utilizare prezintă modul cum se instalează aplicația și pașii de folosire a aplicației din cele 3 perspective: spital, doctor rezident și admin.

Capitolul 1

Specificații funcționale

1.1 Problema studiată

Problema ce a fost tratată în cadrul lucrării de licență se referă la repartizarea medicilor la spitale pe cât de bine posibil în funcție de niște costrângerile. Constraințele sunt constituite din:

- nota doctorului la examenul de rezidențiat;
- specializarea pe care o deține medicul;
- preferințele doctorului pentru spitale;
- numărul de locuri disponibile per specializare din fiecare spital;
- poziția spitalului (aspect important în cazul repartizării unor doctori căsătoriți);

Având în vedere aceste constrângerile, algoritmul de stable matching ("potrivire stabilă") utilizat realizează repartizările cât mai bine posibil. Scopul potrivirii stabile este de a asigura o asignare echitabilă și eficientă a medicilor în spitale, luând în considerare preferințele acestora și nevoile fiecărei unități sanitare. De asemenea, se poate lua în calcul și o abordare ce implică indiferență medicilor asupra unor preferințe pentru spitale.

1.2 Fluxuri de lucru

În cadrul problemei discutate se identifică trei tipuri de actori ce vor fi folosiți ca și tipuri de utilizatori în cadrul aplicației. Cele trei tipuri sunt :

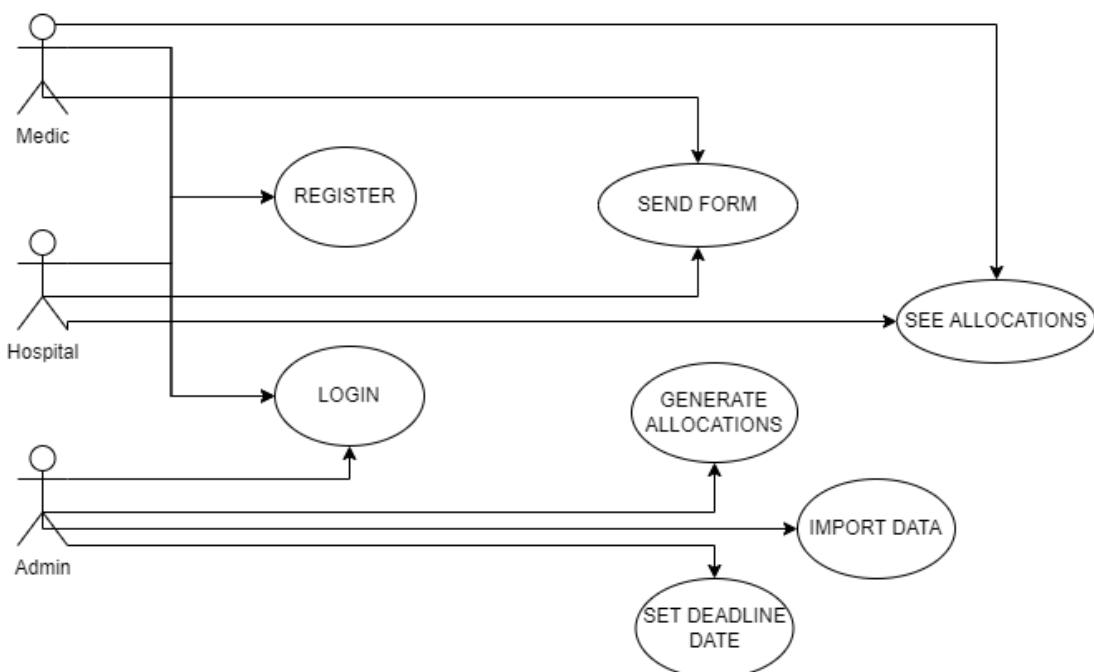
1. Medic – > completează și trimit un formular ce conține preferințele pentru spitale împreună cu nota la examenul de rezidențiat și cu specializarea sa către admi-

nistrator până la data limită; (formularul poate fi trimis de câte ori dorește medicul în cazul în care și-a modificat preferințele)

2. Spital – > completează și trimit un formular ce conține specializările și locurile libere per fiecare specializare până la data limită de completare;(formularul poate fi trimis de câte ori dorește spitalul în cazul în care se modifică locurile libere disponibile)

3. Administrator – > setează data limită pentru completarea/trimiterea formularelor de către spitale, respectiv medici și generează repartizarea medicilor la spitale, alegând unul din algoritmii puși la dispoziție; poate importa date pentru generarea repartizărilor (spitale cu specializări și locuri disponibile; doctori cu preferințe, specializare și nota la examen)

Înainte ca fiecare tip de utilizator să pornească fluxul său de lucru va trebui să se autentifice prin email și parolă sau să-și creeze un cont în cadrul aplicației pentru a se autentifica. În cadrul creării unui cont se va specifica rolul de medic/spital, numele medicului/spitalului, precum și datele necesare autentificării. Important de specificat este că utilizator de tip administrator nu își poate crea un cont, ci primește contul de admin de la developerul aplicației sau poate fi făcut admin de către un administrator deja existent.



Capitolul 2

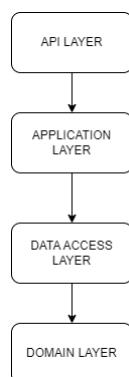
Arhitectura sistemului

În acest capitol se va vorbi în detaliu despre arhitectura și schema bazei de date din spatele acestei aplicații de repartizare. Arhitectura aplicației este de tip client-server, clientul este reprezentat de un Frontend React App, iar serverul de un .NET Backend App.

2.1 Layered Architecture

În cadrul acestei aplicații s-a folosit pe partea de backend o arhitectură structurată pe patru straturi. Acest tip de arhitectură este numită Layered Architecture și este menită pentru ca fiecare strat să aibă propriul lui set de responsabilități, iar dependențele dintre acestea să curgă în direcție descendentală, de la cel mai superior strat la cel mai inferior. Straturile superioare depind de straturile inferioare, iar straturile inferioare nu sunt conștiente de straturile superioare. Această separare a preocupărilor sporește modularitatea, menținabilitatea și testabilitatea aplicației.

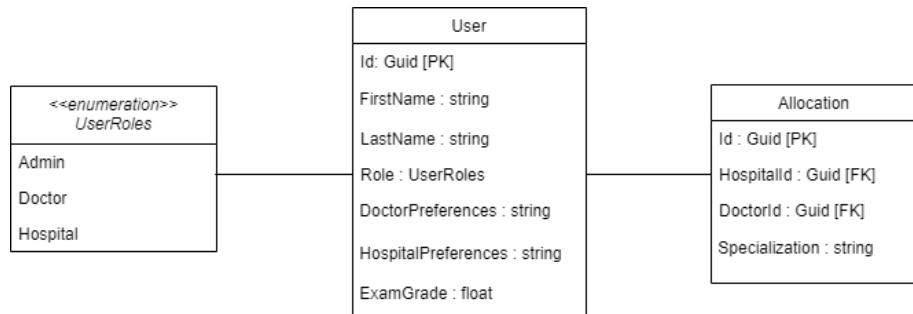
Cele patru straturi ce stau la baza acestei arhitecturi sunt: Domain, DataAccess, Application și API.



Layered Architecture Flow

2.1.1 Domain Layer

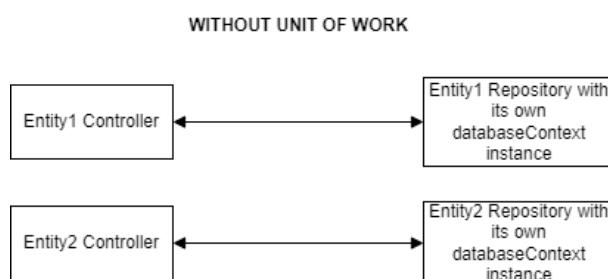
Acest strat este reprezentat de catre modelele ce sunt folosite în cadrul aplicației. Modelele au rolul de a oferi o modalitate de a abstractiza implementarea stocării datelor de bază. Totodată acestea mențin coerența datelor în aplicarea logicii aplicației și în menținerea integrității datelor. În cadrul proiectului propus de mine se regăsesc trei modele: User, UserRoles, Allocation.

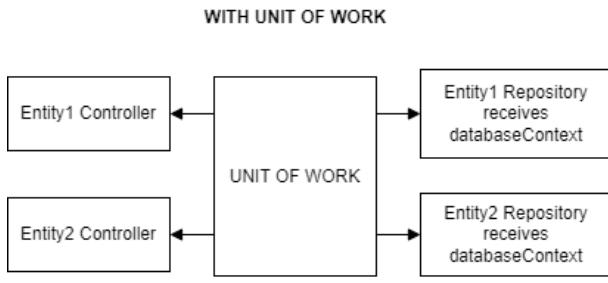


Class Diagram

2.1.2 DataAccess Layer

În cadrul acestui strat se manipulează și se accesează datele din cadrul bazei de date. De asemenea, din acest nivel se realizează conexiunea cu baza de date și se transmite modul în care baza de date va fi creată. Structura bazei de date sau modul în care baza de date va fi creată se transmite prin intermediul migrărilor. Aceste migrări sunt salvate în cadrul acestui strat și tot prin intermediul lor se pot aduce modificări asupra structurii bazei de date. Aici sunt prezente și repozișoarele/depozitele ce sunt destinate pentru a efectua operații asupra entităților pe care le gestionează. Repozișoarele la randul lor sunt gestionate de un UnitOfWork. UnitOfWork este un pattern ce este folosit cu Repository Pattern pentru a efectua mai multe operații în cadrul unei singure tranzacții. Pentru a se înțelege mai bine se poate vizualiza figura de mai jos.





2.1.3 Application Layer

Stratul application servește ca și un intermediar între stratul API și straturile Domain și DataAccess. La acest nivel se gestionează comenzi și interogările din cadrul aplicației, fiind responsabil de logica și funcționalitățile specifice aplicației. Este folosit pattern-ul arhitectural Command and Query Responsibility Segregation, ceea ce înseamnă că sunt segregate/separate comenzi(insert, delete, update) de queries(read). Fiecare comandă și fiecare interogare este pregătită cu un datele necesare pentru a fi gestionată de către un handler. Acest lucru ajută la folosirea pattern-ului Mediator implementat cu ajutorul pachetului MediatR(care va fi prezentat în capitolul următor). În același timp se regăsesc aici și interfețele ce sunt folosite în logica aplicației.

2.1.4 API Layer

Stratul API are rolul de a asigura integrarea aplicației cu alte sisteme, de a gestiona interacțiunea cu servicii externe prin intermediul application programming interface (API). În acest loc se află controller-ele, DTO-urile, profilele modelelor existente în Domain Layer și tot aici se definesc endpoint-urile, metodele HTTP, parametri, formatul datelor și alte detalii necesare pentru a accesa și utiliza funcționalitățile aplicației.

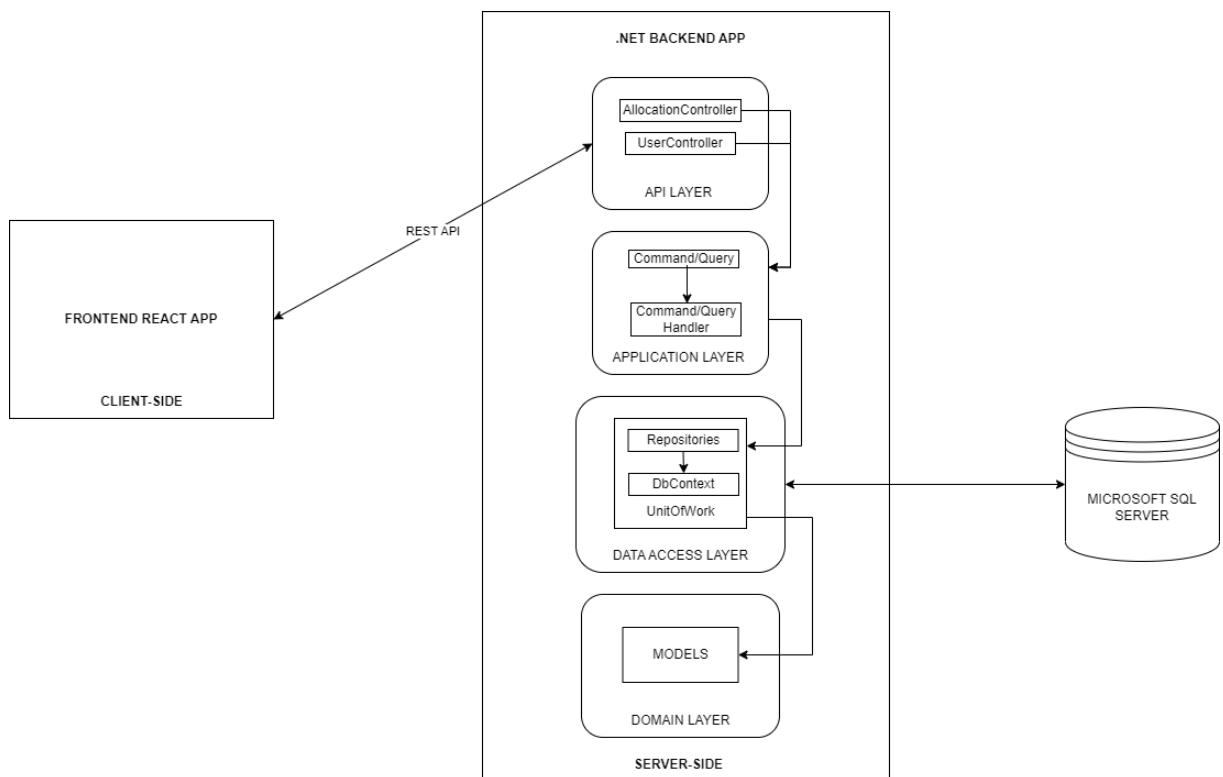
Controller-ele au rolul de a gestiona cererile din partea utilizatorului sau a serviciilor externe. Un controller este destinat să gestioneze cererile specifice unei entități.

DTO-urile sau data transfer object-urile sunt obiecte ce sunt folosite pentru transferul de date între straturile aplicației. Prin utilizarea DTO-urilor, se poate controla exact ce informații sunt transferate și cum sunt structurate.

Un profil este utilizat pentru a specifica mapările între obiectele sursă și obiectele destinație. Aceasta definește regulile și convențiile necesare pentru a realiza mapările într-un mod consistent și eficient.

De asemenea, atât administrarea securității aplicației, cât și gestionarea rolurilor, sunt furnizate tot în cadrul acestui strat.

2.2 Architectural Diagram



Capitolul 3

Implementare și testare

3.1 Tehnologii Folosite

3.1.1 React.js

React.js este o bibliotecă open-source front-end JavaScript pentru construirea de interfețe cu utilizatorul bazate pe componente.

Pe partea de frontend a aplicației de repartizare a fost folosită tehnologia React.js, proiectul de frontend fiind împărțit pe componente, fiecare componentă având un fișier CSS cu stilul ei, existând totodată și un stil general pentru toată partea de UI. De asemenea, este prezentă și o componentă de tip Higher-Order numită WithAuth, care are rolul de a gestiona autentificarea componentelor. În cazul în care utilizatorul este autentificat, se va randa componenta primită ca argument, alfel se va randa componenta Login.

```
1 function WithAuth(Component) {
2   return function AuthenticatedComponent(props) {
3     const isAuthenticated = Cookies.get('jwtToken') !== undefined;
4
5     if (isAuthenticated) {
6       return <Component {...props} />;
7     } else {
8       return <Navigate to="/login" />;
9     }
10   }
11 }
```

Listing 3.1: Higher-Order Component(HOC)

Comunicarea cu backend-ul se face prin request-uri HTTP la endpoint-urile definite în stratul API cu ajutorul bibliotecii Axios.

3.1.2 .NET

.NET este o platformă de dezvoltare software dezvoltată de Microsoft. Limbajul de programare utilizat este C#, acest limbaj fiind compilat într-un limbaj intermediu ce se numește Microsoft Intermediate Language (MSIL) care la rândul lui este interpretat de către un mediu de execuție numit Common Language Runtime (CLR).

În aplicația propusă de mine s-au utilizat două framework-uri din cadrul .NET: ASP.NET Core și .NETCore. ASP.NET Core este folosit pentru crearea API-ului, iar .NETCore este folosit în toate straturile, fiind un framework modular, care oferă posibilitatea de construire de aplicații consolă sau class library.

Pachete folosite

Cele mai importante pachete folosite în acest proiect sunt :

- AutoMapper
- MediatR
- AspNetCore.Authentication
- AspNetCore.Identity
- AspNetCore.EntityFrameworkCore
- Faker.NET

AutoMapper

AutoMapper este un pachet pentru proiectele .NET care facilitează maparea automată a datelor între obiecte de tipuri diferite. Acesta utilizează convenții personalizate pentru a determina cum se face maparea între proprietăți și cum se rezolvă diferențele dintre obiectele sursă și destinație. În cazul acesta AutoMapper face maparea între DTO-urile și modelele existente în aplicație. A fost folosit în cadrul stratului API-ului pentru crearea profilelor. Un profil arată în felul următor:

```
1 public class UserProfile : Profile  
2 {  
3     public UserProfile()
```

```

4     {
5         CreateMap<User, UserGetDto>();
6         CreateMap<RegisterModel, InsertUser>()
7             .ForMember(dest => dest.UserName, opt => opt.MapFrom(src => src
8                 .Email))
9             .ForMember(dest => dest.Email, opt => opt.MapFrom(src => src.
10                Email))
11             .ForMember(dest => dest.FirstName, opt => opt.MapFrom(src =>
12                src.FirstName))
13             .ForMember(dest => dest.LastName, opt => opt.MapFrom(src => src
14                 .LastName))
15             .ForMember(dest => dest.Role, opt => opt.MapFrom(src => src.
16                 Role));
17             .ForMember(dest => dest.Password, opt => opt.MapFrom(src => src
18                 .Password));
19     }
20 }
```

Listing 3.2: Profil pentru modelul User

MediatR

MediatR este un pachet pentru proiectele .NET cu ajutorul căruia se implementează design pattern-ul Mediator. Design pattern-ul Mediator definește un obiect care încapsulează modul în care un set de obiecte interacționează între ele. Prin intermediul MediatR, obiectele pot trimite mesaje către mediator și pot primi răspunsuri sau pot solicita execuția unor acțiuni către alte obiecte fără a cunoaște detaliile despre acele obiecte.

Obiectul Mediator este existent în fiecare controller din cadrul stratului API, iar în

stratul Application sunt prezente interfețele din pachetul MediatR: IRequest și IRequestHandler. Aceste două clase sunt folosite astfel:

- IRequest este o interfață care reprezintă o cerere (în cazul acesta o comandă sau un query) trimisă de către obiectul Mediator din stratul API; pentru fiecare cerere se va crea propria clasă care implementează IRequest.

```
1  public class InsertAllocation : IRequest<Allocation>
2  {
3      public Guid Id { get; set; }
4
5      public Guid? HospitalId { get; set; }
6
7      public Guid? DoctorId { get; set; }
8
9      public string? Specialization { get; set; }
10 }
```

Listing 3.3: Clasa ce implementeaza IRequest

- IRequestHandler este o interfață prin care se gestionează cererile; O clasă care implementează IRequestHandler primește o cerere specifică și este responsabilă de executarea logicii necesare pentru a răspunde la cerere.

```
1  public class InsertAllocationHandler : IRequestHandler<InsertAllocation,
2          Allocation>
3  {
4      private readonly IUnitOfWork _unitOfWork;
5
6      public InsertAllocationHandler(IUnitOfWork unitOfWork)
7      {
8          _unitOfWork = unitOfWork;
9      }
10
11     public async Task<Allocation> Handle(InsertAllocation request,
12         CancellationToken cancellationToken)
13     {
14         var allocation = new Allocation
15         {
16             Id = Guid.NewGuid(),
17             HospitalId = request.HospitalId,
18             DoctorId = request.DoctorId,
19             Specialization = request.Specialization,
```

```

18     } ;
19
20
21     await _unitOfWork.AllocationRepository.Create(allocation) ;
22     await _unitOfWork.Save() ;
23
24     return allocation;
25 }
26 }
```

Listing 3.4: Clasa ce implementeaza IRequestHandler

AspNetCore.Authentication

AspNetCore.Authentication este responsabil de gestionarea autentificării în aplicație. Acest pachet oferă funcționalități pentru autentificarea utilizatorilor folosind diverse scheme de autentificare, cum ar fi autentificare bazată pe token-uri (Token-based Authentication) sau autentificare cu un furnizor tert (Third-Party Authentication). Aceste proiect folosește schema de autentificare bazată pe token-uri. Se verifică dacă datele de conectare sunt valide, iar în cazul în care sunt valide se crează JWT Token-ul ce este trimis de către server la client. În momentul în care Token-ul ajunge în aplicația client, acesta va fi salvat într-un Cookie.

AspNetCore.Identity

AspNetCore.Identity este un pachet care se concentrează pe gestionarea identității utilizatorilor în aplicații ASP.NET Core. Acesta oferă funcționalități pentru gestionarea utilizatorilor, autentificarea și autorizarea acestora. Prin intermediul acestui pachet se creează și gestionează conturile de utilizator. De asemenea, gestionarea rolurilor(doctor, hospital, admin) și a permisiunii rolurilor din cadrul aplicației sunt posibile tot datorită acestui pachet.

Cele două obiecte UserManager și RoleManager din AspNetCore.Identity fac posibile cele menționate mai sus.

AspNetCore.EntityFrameworkCore

AspNetCore.EntityFrameworkCore este un pachet din framework-ul ASP.NET Core care oferă suport pentru integrarea cu Entity Framework Core. Entity Framework

Core este un ORM (Object-Relational Mapping) modern și ușor de utilizat, care permite interactionarea cu baze de date relationale folosind obiecte și cod C#.

AspNetCore.EntityFrameworkCore oferă funcționalități esențiale pentru utilizarea și configurarea Entity Framework Core în aplicațiile ASP.NET Core. Funcționalitățile folosite în aplicație sunt următoarele:

- Contextul bazei de date – > definirea și configurarea contextului bazei de date, care reprezintă conexiunea și schema bazei de date utilizate în aplicație. Aici se stabilesc, totodată, relațiile dintre entitățile ce sunt prezente în aplicație.

- Migrări – > sunt utilizate pentru a gestiona evoluția schemei bazei de date în timpul dezvoltării și implementării aplicației. Acestea pot fi folosite pentru a crea, modifica sau șterge tabele, relații sau alte obiecte din baza de date;

- Interacțiunea cu baza de date – > această funcționalitate se referă la metodele și clasele existente în AspNetCore.EntityFrameworkCore; Cu ajutorul acestora se pot efectua operațiuni de creare, citire, actualizare și ștergere (CRUD) în baza de date.

Faker.NET

Faker.NET este un pachet din cadrul .NET folosit pentru generarea de date test. Acesta include generarea de nume, adrese, numere de telefon, adrese de email, texte, date, culori și multe altele. Pachetul are suport pentru o gamă largă de limbi și culturi, permitând generarea de date specifice unei țări sau regiuni.

În cadrul acestei aplicații, Faker a fost folosit pentru crearea unui serviciu de populare a bazei de date cu doctori și spitale cu preferințe deja stabilite pentru a putea testa algoritmii folosiți pe seturi de date mai mari. Se creează o clasă Generator cu ajutorul căreia serviciul va popula baza de date. Serviciul funcționează astfel, se stabilește numărul de spitale și numărul de doctori ce urmează a fi creați. Se creează spitalele cu numele și email-ul generat de Faker, iar dintr-o listă predefinită de specializări se aleg în mod aleatoriu un număr aleatoriu de specializări. De asemenea, pentru fiecare specializare în parte se alege un număr de până la 5 locuri libere.

După terminarea generării spitalelor, se vor crea doctorii cu nume, prenume și email ales de Faker. Din cadrul spitalelor existente se alege în mod aleatoriu un număr aleatoriu de spitale pentru preferințe. Din cadrul specializărilor spitalelor selectate se va alege o specializare care va fi specializarea medicului.

3.1.3 Microsoft SQL SERVER

Microsoft SQL Server este un sistem de gestiune a bazelor de date relaționale dezvoltat de Microsoft. Este unul dintre cele mai populare și puternice produse de acest tip disponibile pe piață.

Am ales Microsoft SQL SERVER datorită integrării puternice cu ecosistemul .NET. Există suport nativ pentru lucrul cu baze de date SQL Server în framework-ul .NET, ceea ce face integrarea și interacțiunea cu baza de date foarte ușoară și eficientă. Oferă scalabilitate și performanță, fiind proiectat pentru gestionarea a unui număr mare de încărcări de date și pentru a oferi rezultate rapide în urma interogărilor și tranzacțiilor. De asemenea, pune un accent deosebit pe securitatea datelor, oferind caracteristici puternice pentru autentificare, autorizare și criptare a datelor.

Microsoft SQL Server este însotit de o suită de instrumente puternice pentru administrarea și gestionarea bazelor de date, precum SQL Server Management Studio (acesta fiind folosit în cadrul acestui proiect) și Azure Data Studio. Aceste instrumente facilitează dezvoltarea, testarea și administrarea bazelor de date SQL Server.

3.2 Algoritmi utilizați

Fiind o aplicație de repartizare a medicilor la spitale, pentru partea repartizării s-au folosit doi algoritmi de repartizare pentru două abordări diferite, prima abordare fiind una clasică în care fiecare medic are un top al preferințelor, iar cea de-a doua abordare este cea în care medicul privește unele preferințe cu indiferență.

3.2.1 Algoritmul 1

Pentru abordarea clasică, algoritmul folosit este o variantă a algoritmului Gale-Shapley care este folosit pentru aflarea soluțiilor problemelor de asignare și repartizare a resurselor într-un mod eficient și stabil.

În cazul problemei discutate în această lucrare de licență, algoritmul funcționează după pașii următori:

1. Se pregătesc structurile de date ce urmează a fi folosite și se împart userii în două liste separate : doctori și spitale.

```

1 var query = new GetAllUsers();
2 var users = await _mediator.Send(query);
3 var usersGetDto = _mapper.Map<IEnumerable<UserGetDto>>(users);
4
5 var doctors = usersGetDto.Where(u => u.Role == "Doctor").ToList();
6 var hospitals = usersGetDto.Where(u => u.Role == "Hospital").ToList();
7
8 var allocation = new Dictionary<string, string>();
9 var specializationSelected = new Dictionary<string, string>();
10
11 var doctorPreferences = new Dictionary<string, List<string>>();
12
13 var hospitalSpecializationPreferences = new Dictionary<string, List<string>>()
    >> ();
14
15 var hospitalSpecializationAvailablePlaces = new Dictionary<string,
    Dictionary<string, int>>();

```

2. Se sortează descrescător lista de doctori după nota la examenul de rezidențiat, astfel încât procesul de repartizare să fie mai ușor de făcut.

```
1 doctors = doctors.OrderByDescending(p => p.ExamGrade).ToList();
```

3. Se "decodează" preferințele doctorilor și a spitalelor , acestea fiind trimise și salvate sub următorul format:

Preferințe doctori: preferință 1, preferință 2, preferință 3, ... , preferință n;

Preferințe spitale: specializarea 1-numărul de locuri disponibile, specializarea 2-numărul de locuri disponibile, ... , specializarea n-numărul de locuri disponibile;

În același timp cu acest proces se populează structurile de date mentionate mai sus cu datele decodeate.

4. Cât timp numărul de asignări a doctorilor la spitale este mai mic decât numărul total de doctori se execută următoarele acțiuni:

- se parcurge lista de doctori;
- dacă doctorul curent nu a fost repartizat, se încearcă repartizarea doctorului curent și se preiau preferințele și specializarea acestuia;
 - se parcurge lista de preferințe (de spitale)
 - se încearcă repartizarea medicului curent la primul spital ce are locuri rămase la specializarea medicului;

- în cazul în care sunt locuri la spitalul curent la specializarea medicului , medicul este alocat spitalului;

- în cazul în care nu mai sunt locuri la spitalul curent se ia medicul care a fost repartizat la acest spital și la specializarea medicului curent și care are spitalul curent pe cel mai jos loc în topul preferințelor comparativ cu restul medicilor repartizați la acest spital; Se compară locul preferinței acestui medic cu locul preferinței medicului ce urmează a fi repartizat, dacă preferința medicului ce urmează a fi repartizat este mai sus în top, celălalt medic va fi eliberat de la spitalul la care a fost alocat, iar medicul curent va fi alocat spitalului, altfel se merge mai departe;

- algoritmul se termină când toți doctorii sunt repartizați la spitale;

3.2.2 Algoritmul 2

Pentru cea de-a doua abordare a problemei, algoritmul inițial a fost modificat pentru cazul în care doctorii au indiferență de la un anumit loc în topul preferințelor în jos. Am stabilit ca primele două locuri din top să fie fără indiferență, iar de la a treia poziție în top a spitalelor să nu mai conteze ordinea, doctorii să fie indiferenți asupra spitalului la care vor fi repartizați. Aceasta înseamnă că medicul rezident acordă o importanță mai mare primelor două spitale și își dorește să fie repartizat acolo în primul rând. Dacă aceste spitale nu sunt disponibile sau nu îi acceptă propunerea, medicul rezident va continua să facă propuneri către spitalele rămase în ordinea preferințelor, cu indiferență între ele.

În cazul acestei abordări, algoritmul are aceeași primi 3 pași ca și algoritmul precedent, următorii pași fiind:

4. Se inițializează cu 0 o listă proposals în care se va reține dacă medicul a fost alocat sau nu unui spital.

5. Cât timp numărul de asignări a doctorilor la spitale este mai mic decât numărul total de doctori se execută următoarele acțiuni:

- se parcurge lista de doctori;

- dacă doctorul curent nu a fost repartizat și în lista proposals are valoarea 0 , se încearcă repartizarea și se preiau preferințele și specializarea acestuia; dacă nu va fi repartizat la nici un spital, valoarea sa în lista proposals devine -1;

- dacă doctorul curent nu a fost repartizat și în lista proposals are valoarea -1, se alege un doctor cu o probabilitate de $1/n$ (unde n este numărul de doctori ce au

fost repartizați la o preferință asupră căruia au indiferență și care corespunde cu una din primele două preferințe a doctorului curent; doctorii au aceeași preferință ca și doctorul curent) din lista de doctori ce respectă condițiile menționate anterior între paranteze; Doctorul curent este repartizat pe locul doctorului ales, iar doctorul ales este "concediat" din cadrul spitalului la care a fost alocat, urmând a fi supus din nou procesului de repartizare;

- algoritmul se termină când toți doctorii sunt repartizați la spitale;

Concluzii asupra algoritmilor

În concluzie, atât algoritmul clasic de repartizare, cât și varianta cu indiferență menționată pot fi utilizate în funcție de specificul situației și preferințele implicate. Este important să se definească regulile și criteriile de asignare în mod clar și să se comunice acest lucru tuturor participantilor pentru a asigura un proces de repartizare echitabil și transparent.

Capitolul 4

Manual de utilizare

Aplicația de repartizare a medicilor rezidenți la spitale (numită MEDDIS) este o aplicație web cu un design simplu și placut ce poate fi utilizată de pe orice dispozitiv. În cadrul acesteia utilizatorii se pot conecta ca și administratori, spitale sau doctori. În acest capitol se va prezenta aplicația propusă de mine și pașii pentru o bună utilizare a acesteia.

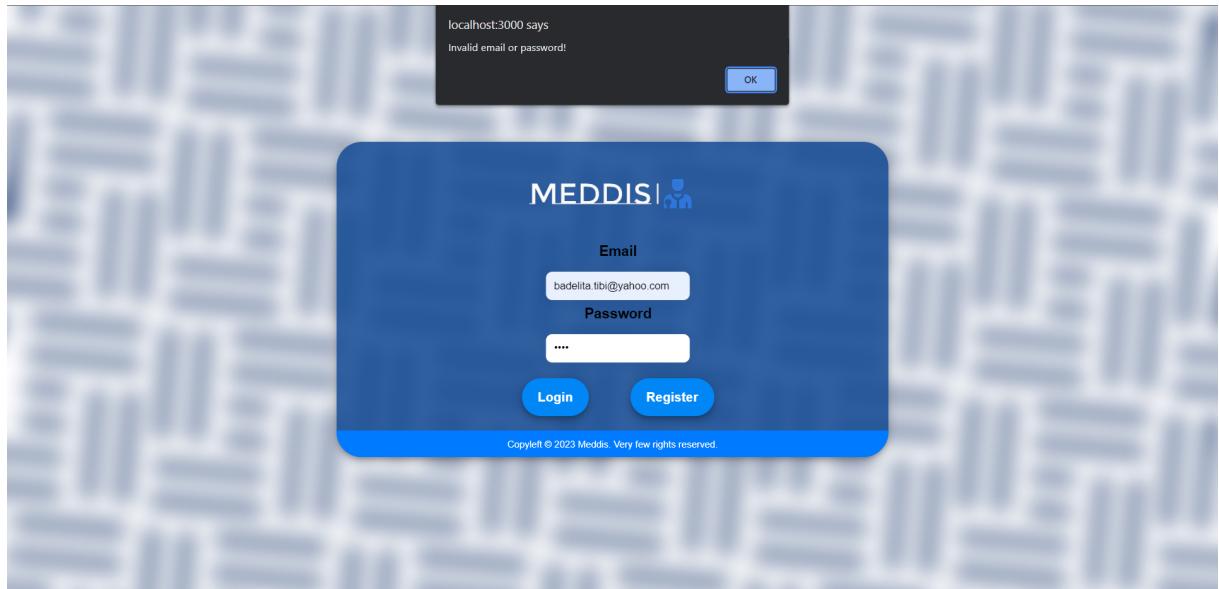
4.1 Pagina de logare

Această pagină este destinată logării utilizatorilor și este totodată pagina de pornire a aplicației pentru un utilizator ce nu este înregistrat sau logat în această aplicație.



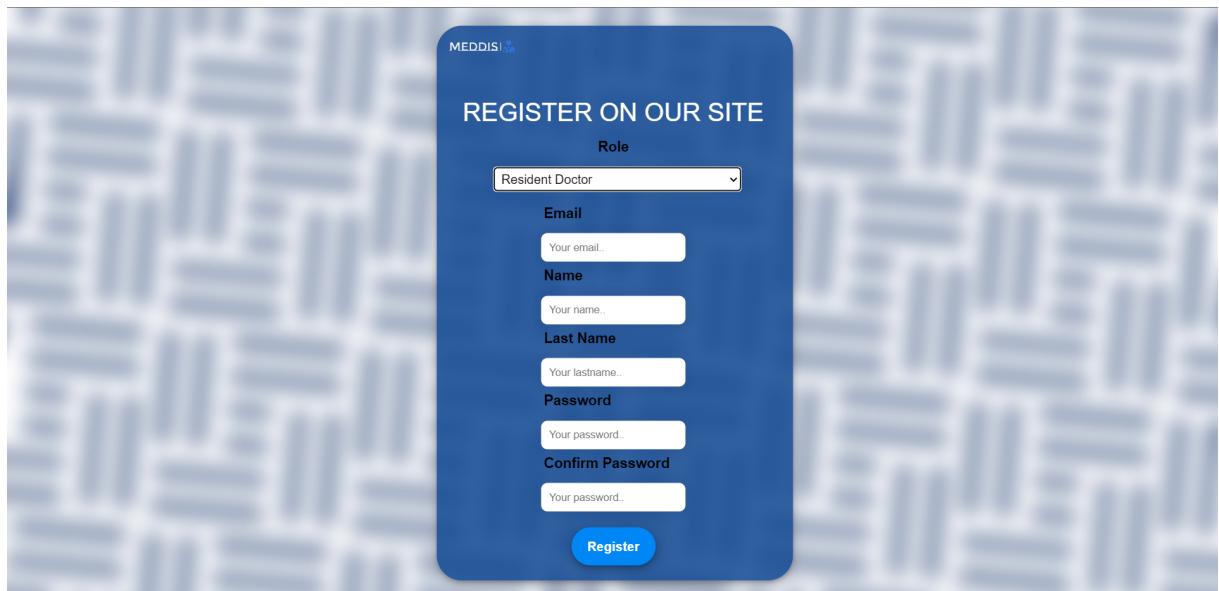
În cazul logării cu un cont existent, utilizatorul va fi redirectionat către pagina principală, iar în cazul logării cu un cont inexistent sau cu date de logare invalide va

primi un mesaj de avertizare. Dacă cineva dorește să acceseze o pagină prin URL fără a fi logat, va fi redirectionat către pagina de login.



4.2 Pagina de înregistrare

Pagina de înregistrare este destinată pentru înscrierea unui utilizator ca medic rezident sau ca spital în cadrul MEDDIS. Rolul de administrator poate fi atribuit unui cont, doar de către un alt administrator deja existent. În urma înregistrării completând corect toate câmpurile , utilizatorul va primi un mesaj de înregistrare reușită, fiind redirectionat către pagina de login.

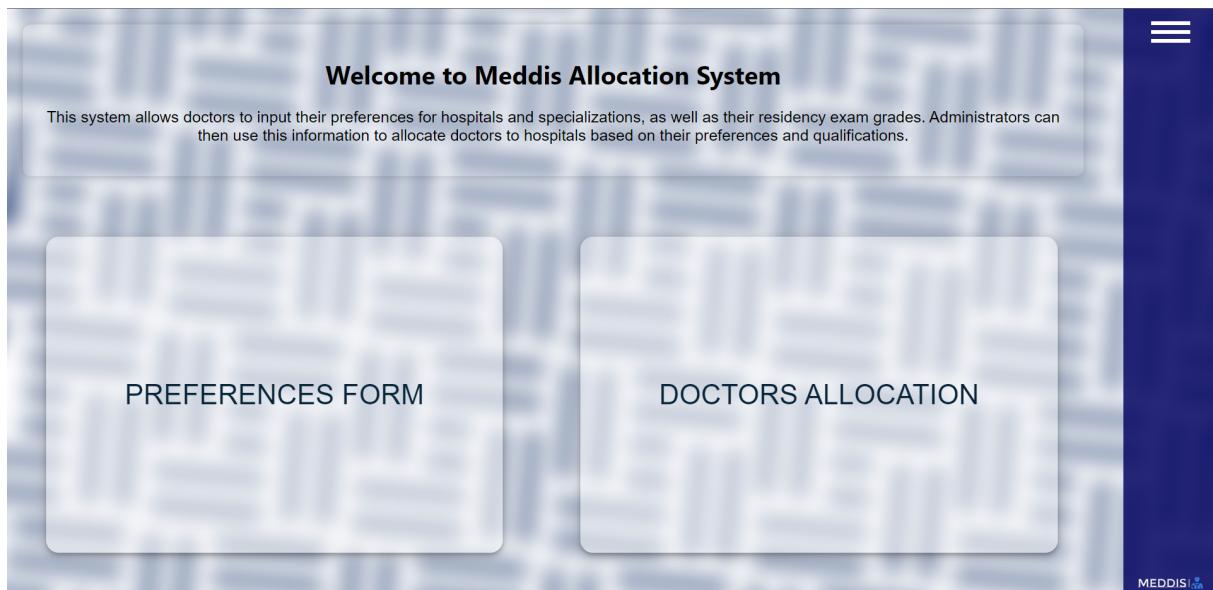


4.3 Pagina principală

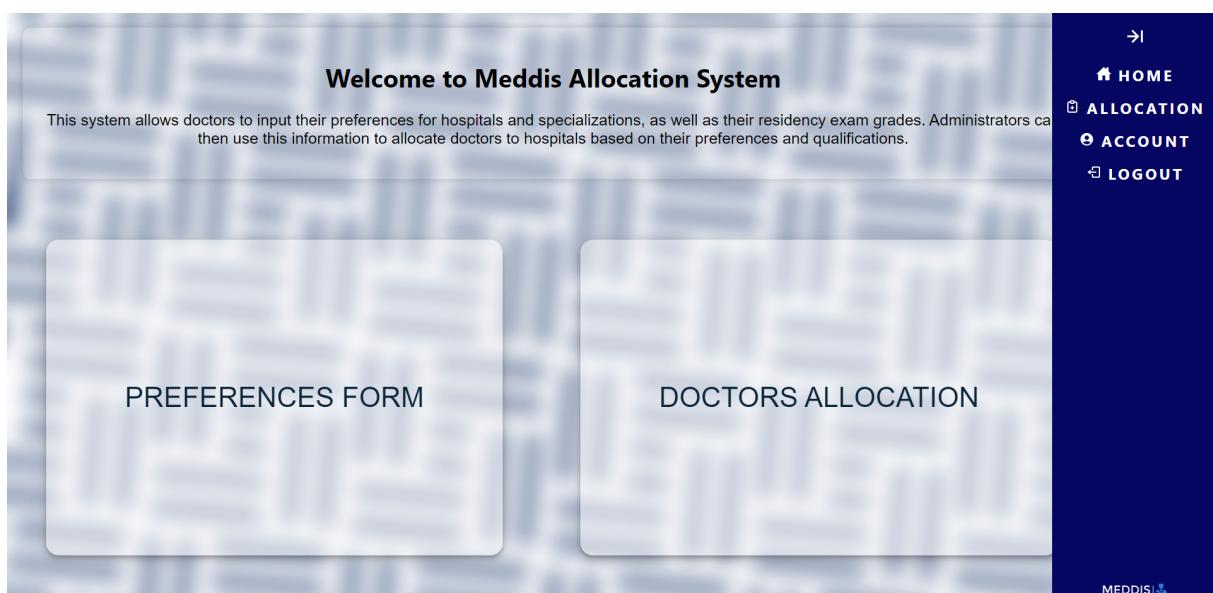
Pagina principală este comună pentru rolurile de doctor și spital, dar diferită pentru administrator.

Afișare pagină pentru doctor/spital

Aici vom vedea o scurtă descriere asupra modului cum funcționează sistemul de repartizare MEDDIS și două butoane către principalele pagini din cadrul site-ului: pagina de completare a preferințelor și pagina de vizualizare a repartizărilor.



În partea dreaptă avem meniul care va fi afișat în momentul unui click dat pe cele trei linii orizontale. Acesta va apărea în toate paginile după momentul logării ca medic sau spital. Din cadrul acestuia putem naviga între pagini sau ne putem deloga.



Afișare pagină pentru administrator

Pagina principală a administratorului reprezintă un dashboard în care poate vizualiza toți utilizatorii din cadrul aplicației. De asemenea, poate căuta după nume, poate șterge, poate actualiza pe fiecare în parte. Alte operații pe care administratorul le poate efectua sunt:

- import de date în format CSV;
- setarea datei limită de completare și transmitere a formularului cu preferințe;
- selectarea algoritmului de repartizare și generarea repartizărilor cu algoritmul selectat;

The screenshot shows the Admin Dashboard with a table of users. The columns are Name, Role, and Actions (Delete, Update). The users listed are Hospital NewYork (Hospital), Spitalul Clinic de Urgență București (Hospital), Tiberiu-Gabriel Badelita (Doctor), Vitalie Bocicov (Doctor), Hospital Suceava (Hospital), Spitalul Clinic de Recuperare (Hospital), and Ana Pop (Doctor). Below the table are buttons for Import Data, Set Form Deadline (Current Deadline: 2023-07-01), Select an algorithm, and Generate Allocations.

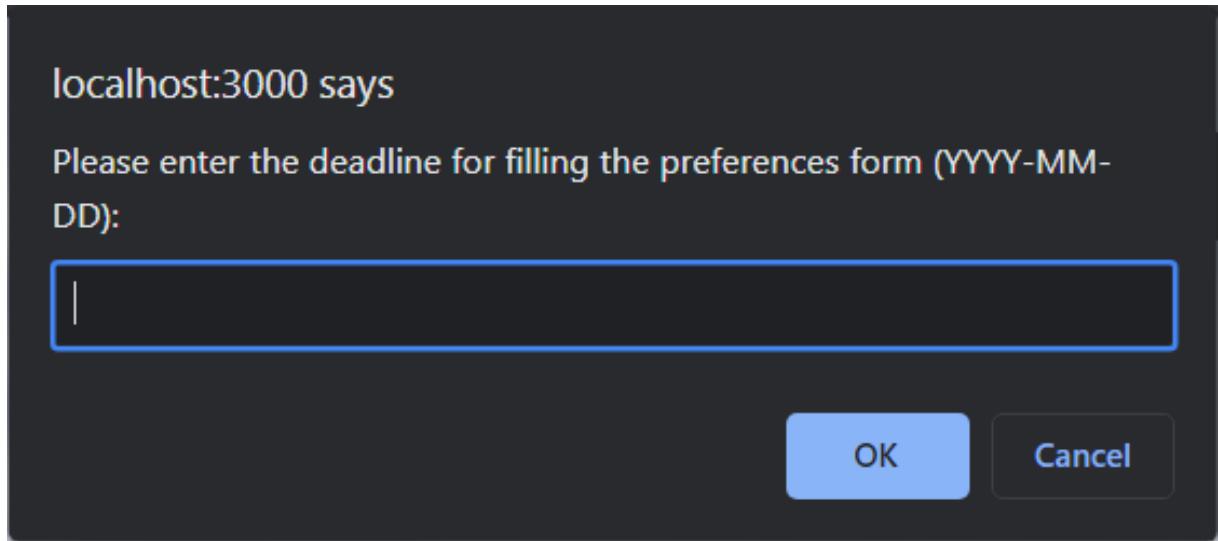
| Name | Role | Actions |
|--------------------------------------|----------|---|
| Hospital NewYork | Hospital | <button>Delete</button> <button>Update</button> |
| Spitalul Clinic de Urgență București | Hospital | <button>Delete</button> <button>Update</button> |
| Tiberiu-Gabriel Badelita | Doctor | <button>Delete</button> <button>Update</button> |
| Vitalie Bocicov | Doctor | <button>Delete</button> <button>Update</button> |
| Hospital Suceava | Hospital | <button>Delete</button> <button>Update</button> |
| Spitalul Clinic de Recuperare | Hospital | <button>Delete</button> <button>Update</button> |
| Ana Pop | Doctor | <button>Delete</button> <button>Update</button> |

Actualizarea datelor unui utilizator sau promovarea acestuia ca admin sunt posibile prin intermediul unui modal în care modificăm datele dorite.

The screenshot shows the Admin Dashboard with a modal window for updating a user. The modal has fields for Name (Hospital's Name: Hospital Suceava), Role (dropdown: Hospital, Admin, Hospital selected), and Email (suceava@yahoo.com). A blue button labeled "Update" is at the bottom right of the modal.

Importul de date se face prin încărcarea unui fișier cu formatul csv , ce conține pe fiecare linie numele, prenumele, rolul, email-ul, preferințele (scrise conform modelului din capitolul anterior) fiecărui utilizator ce se dorește a fi introdus astfel.

Setarea datei limită de completare și transmitere a formularului cu preferințe se face prin introducerea datei respectând formatul menționat. Se fac validări asupra datei introduse, asigurând astfel introducerea corectă a acesteia.



O ultimă operație ce poate fi făcută din postura de administrator este cea de generare a repartizărilor în momentul depășirii zilei deadline-ului. După generare , acestea vor fi afișate automat în pagina de vizualizare a repartizărilor.

4.4 Pagina de completare a preferințelor

Pagina de completare a preferințelor este o pagină în care atât doctorii , cât și spitalele își trimit preferințele. Paginile acestora sunt diferite datorită faptului că tipul de preferințe al acestora este diferit. De asemenea, trebuie de menționat că utilizatorii pot să-și trimită și să-și actualizeze preferințele până la data limită stabilită de către administrator. Această dată va fi afișată în partea de sus a paginii.

Afișare pagină pentru doctor

The screenshot shows a form for a doctor to enter their preferences. At the top, a message says "Complete your preferences form below. You can change your preferences at any time before the deadline: 2023-07-01." Below this, under "Your Specialization:", there is a dropdown menu set to "Cardiology". Under "Hospital Preferences:", there are three sections for "Hospital 1", "Hospital 2", and "Hospital 3", each with a dropdown menu. "Hospital 1" is set to "Spitalul Clinic de Urgență București", "Hospital 2" to "Spitalul Clinic de Recuperare", and "Hospital 3" to "Hospital NewYork". There are also minus and plus buttons between Hospital 2 and Hospital 3. Below these, an "Exam Grade" field shows "9.76" with a "SEND" button next to it. The MEDDIS logo is in the bottom right corner.

Afișare pagină pentru spital

The screenshot shows a form for a hospital to enter its preferences. At the top, a message says "Complete your preferences form below. You can change your preferences at any time before the deadline: 2023-07-01." Below this, there are four sections for "Specialization 1", "Specialization 2", "Specialization 3", and "Specialization 4", each with a dropdown menu and a "Places Needed" field. "Specialization 1" is set to "Dermatology" with "Places Needed: 3". "Specialization 2" is set to "Oncology" with "Places Needed: 2". "Specialization 3" is set to "Neurology" with "Places Needed: 1". "Specialization 4" is set to "Psychiatry" with "Places Needed: 1". There are also minus and plus buttons between Specialization 3 and Specialization 4. A "SEND" button is located below the fields. The MEDDIS logo is in the bottom right corner.

4.5 Pagina de vizualizare a repartizărilor

Această pagină este creată cu scopul de a vedea rezultatul generării repartizărilor. Medicii și spitalele au această pagină comună. În cadrul acestei pagini se pot face filtrări în funcție de spitale, în funcție de specializări sau se poate căuta direct prin intermediul căsuței search ceea ce se dorește : numele unui medic, numele unui spital sau unei specializări.

The screenshot shows a table titled "Doctors Distribution to Hospitals". The table has three columns: "Doctor Name", "Hospital Name", and "Specialization". The data is as follows:

| Doctor Name | Hospital Name | Specialization |
|------------------------------|---|--------------------------------------|
| Dr. Tiberiu-Gabriel Badelita | Hospital NewYork | Cardiology |
| Dr. Ahmed Maggio | Hospital Miss Jaclyn Nicklaus Mitchell | Neurology |
| Dr. Alda Cremin | Hospital Gavin Zulauf | Physical medicine and rehabilitation |
| Dr. Jeanne Quigley | Hospital Jaida Hodkiewicz | Otolaryngology |
| Dr. Melyssa Corkery | Hospital Lucas Schumm | Obstetrics and gynecology |
| Dr. Mackenzie Terry | Hospital Ms. Rachael Howell Schroeder | Otolaryngology |
| Dr. Armani Bode | Hospital Lester Tom Waelchi Sr. | Urology |
| Dr. Helen O'Keefe | Hospital Jaida Hodkiewicz | Cardiology |
| Dr. Anya Hoeger | Hospital Miss Barrett Darrel Gislason Jr. | Hematology |
| Dr. Vivian Larson | Hospital Dr. Sabina Weber Sr. | Oncology |

At the bottom right of the page, there is a small logo for MEDDISION.

În cazul în care nu s-a depășit data limită de completare a formularelor de preferințe, iar repartizările nu au fost generate, va apărea un mesaj de informare cu privire la acest lucru în momentul accesării paginii.

The screenshot shows a message centered on the screen: "Allocations are not available yet". At the bottom right of the page, there is a small logo for MEDDISION.

4.6 Pagina de vizualizare a contului

În cadrul paginii de vizualizare a contului (pagina ce este comună pentru medici și spitale) putem vizualiza și edita numele, prenumele și mail-ul utilizat la înregistrare. De asemenea, se pot vizualiza și ultimele preferințe trimise de către utilizator, în cazul în care acesta dorește să verifice ceea ce a trimis, astfel putând retrimitre formular cu preferințe.

Afișare pagină pentru doctor

The screenshot shows a mobile application interface for a doctor's profile. At the top right is a vertical dark blue sidebar with a white three-line menu icon. The main content area has a light gray background. A large, semi-transparent rectangular overlay is centered, titled "Profile". Inside the overlay, there are three input fields: "First Name" (Tiberiu-Gabriel), "Last Name" (Badelita), and "Email" (badelita.tibi@yahoo.com). To the left of these fields, the text "Your last preferences set:" is displayed. To the right, under "Hospital preferences:", are listed "Hospital NewYork", "Hospital Miss Jaclyn Nicklaus Mitchell", and "Hospital Suceava". Below this, "Doctor's specialization:" is listed as "Cardiology" and "Exam grade: 9.75". At the bottom of the overlay is a blue "Save" button.

Afișare pagină pentru spital

The screenshot shows a mobile application interface for a hospital profile. It features a similar layout to the doctor profile, with a light gray background and a central semi-transparent "Profile" overlay. On the left, "Hospital's Name:" is set to "Suceava" and "Email:" is "suceava@yahoo.com". On the right, "Your last preferences set:" is shown. Under "Specialization Preferences:", it lists "Dermatology : 1 place available" and "Anesthesiology : 2 places available". A blue "Save" button is at the bottom of the overlay. The vertical dark blue sidebar with a three-line menu icon is visible on the right side of the screen.

Concluzii și direcții viitoare

Lucrarea de licență cu titlul "Aplicație de repartizare a medicilor la spitale" din cadrul Facultății de Informatică Iași prezintă o soluție eficientă pentru gestionarea resurselor umane în domeniul medical. Prin implementarea unei astfel de aplicații, se urmărește optimizarea procesului de repartizare a medicilor la spitale, având ca rezultat o distribuție echitabilă și optimă a personalului medical în funcție de nevoile fiecărei unități sanitare în acest mod îmbunătățindu-se și condițiile din sistem.

Direcții viitoare

În cadrul aplicației web propuse de mine se pot aduce numeroase îmbunătățiri și se poate extinde în perspectiva următoarelor direcții viitoare:

- Reținerea arhivelor fiecărei sesiuni de repartizări;
- Notificarea prin SMS și prin email a utilizatorilor în momentul finalizării repartizărilor;
- Folosirea unor algoritmi mai precisi de repartizare ce țin cont de poziția geografică a spitalelor și de mariajul dintre doi medici;
- Îmbinarea acestui sistem cu alte sisteme din cadrul domeniului medical, cum ar fi un sistem de programări medicale;
- Implementarea unui sistem de feedback și evaluare a performanței;
- Utilizarea de smart contracts pe blockchain pentru a permite automatizarea acordurilor și reglementărilor între medici și spitale.

Bibliografie

- *Stable Matching and Gale Shapley*, Stanford Winter 2022
<https://stanford-cs161.github.io/winter2022/assets/files/lecture17-notes.pdf>
- *Gale-Shapley Algorithm*. Wikipedia
<https://en.wikipedia.org/wiki/Gale-Shapley-Algorithm>
- *Variants of Stable Marriage Problem*
<https://iq.opengenus.org/variants-of-stable-matching/>
- *Layered Architecture*. Medium
<https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057>
- *CQRS pattern*. Microsoft
<https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs>
- *CQRS and Mediator pattern*. Code-maze
<https://code-maze.com/cqrs-mediatr-in-aspnet-core/>
- *.NET Documentation*. Microsoft
<https://learn.microsoft.com/en-us/dotnet/>
- *React Documentation*
<https://react.dev/learn>