**UNIVERSITATEA DIN BUCUREȘTI**



**FACULTATEA**

**DE**

**MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICA**

**Lucrare de licență**

**Aplicație Web pentru căutarea destinațiilor de vacanță destinată grupurilor de turiști**

**Absolvent**

**RADU DENIS FLORIAN**

**Coordonator ştiinţific**

**IONESCU RADU**

**București, februarie 2021**

**Rezumat**

Aplicația web pentru căutarea destinațiilor de vacanță destinată grupurilor de turiști este o aplicație ce are ca obiectiv formarea si sugerarea celor mai potrivite destinații turistice pe care o persoană sau un grup de persoane le poate avea.

Ideea în sine a plecat de la situațiile reale pe care le au oamenii în momentele în care își doresc sa planifice o excursie sau o călătorie, dar intervin factori precum bugetul alocat, clima dorită, țara aleasă, s.a.m.d. Realizarea unor sugestii pentru dorințele unei singure persoane este relativ ușor, aceasta este stăpână pe modul în care își structureaza interesele și nevoile în ceea ce privește locația pentru care optează, dar problema adevărată intervine în momentul în care un grup de persoane cu interese diferite apar și acestea nu reușesc să cadă de comun acord pe o destinație anume, cauzele putând fi multiple și diversificate. În situația acesta, aplicația realizează pe baza unui algoritm, filtrarea intereselor fiecarei persoane si caută cele mai apropiate rezultate de alegerile fiecăruia per total.

Situațiile speciale în care niciun rezultat nu a fost găsit pentru un grup, algoritmul va reanaliza filtrele si va extinde aria de căutare, avand posibilitatea sa intervină cu sugestii ce pot ieși din aria de interes pe care persoanele din grup le-au sugerat, făcând posibilă propunerea unor destinații pe care aceștia nu le-au luat în vedere, dar care prezintă anumite criterii de selecție per individ și care pot fi incluse in rezultatul final ca opțiuni secundare.

În paralel, persoanele unui grup au posibilitatea de a comunica prin intermediul unei mesagerii, iar fiecare conversație va fi disponibilă si diferită pentru fiecare grup în parte, dand posibilitatea unei planificări mai ușoare si avantajul de a putea actualiza lista de interese pentru una sau mai multe persoane.

Fiecare persoană dispune de o pagină principală care la un anumit interval de timp va fi actualizată și vor fi prezentate multiple recomandări de destinații turistice, obținute pe baza setărilor de interese generale pe care aceasta le-a specificat.

Un sistem de notificări permite utilizatorilor să fie înștiințați cu privire la noi invitații în grupuri, excluderea lor de către un administrator dintr-un grup si noi cereri de prietenie de la alți utilizatori.

Aplicația este adaptată pentru toate dispozitivele si este fiabilă pentru orice browser, iar tehnologiile utilizate sunt actualizate si compatibile cu versiunea limbajului pe care ruleaza, atat pe partea de front-end, cat si pe cea de back-end.

**Summary**

The web application built for searching holiday destinations for groups of tourists is an application that aims to form and suggest the most suitable tourist destinations a person or a group of people can have.

The idea itself has been based on the real situations people have when they want to plan a trip or a trip, but factors such as the budget allocation, the desired climate, the chosen country, and so on. Making suggestions for the wishes of a single person is relatively easy, it has ownership of how it structures its interests and needs in terms of the location of its choice, but the real problem arises when a group of persons with different interests emerge, and they fail to agree on a particular destination, the causes can be multiple and diverse. In this situation, the application performs on the basis of an algorithm, filtering the interests of each individual and seeking the closest results to each individual's choices in total.

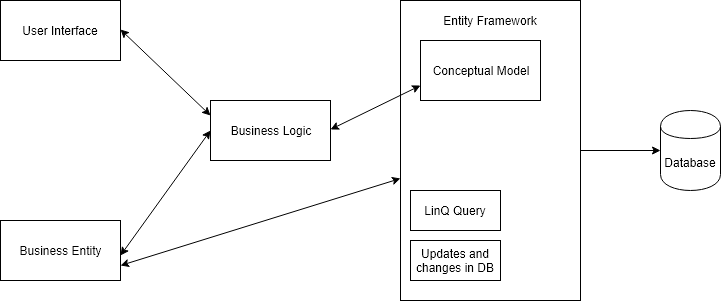
Special situations where no results have been found for a group, the algorithm will re-analyze the filters and extend the search area, with the possibility of intervening with suggestions that may leave the area of interest suggested by the people in the group, making it possible to propose destinations which they have not considered but which have certain selection criteria per individual and which can be included in the final result as secondary options.

In parallel, people in a group have the opportunity to communicate via a messaging, and each conversation will be available and different for each group, allowing easier planning and the advantage of being able to update the list of interests for one or more people.

Each person has a home page that will be updated at a certain time frame and multiple tourist destination recommendations will be presented based on the general interest settings it has specified.

A notification system allows users to be notified of new invitations in groups, their exclusion by an administrator from a group, and new friend requests from other users.

The application is tailored for all devices and is reliable for any browser, and the technologies used are up-to-date and compatible with the version of the language on which it is running, both front-end and back-end.

1. Introducere
   1. Tipul lucrării și subdomeniul specific în care se încadrează tema  
      Lucrarea este o aplicație web, structurată pe cele două părți importante ale programării, partea de front-end și partea de back-end, astfel încât construcția acesteia să fie încadrată tiparului general de aplicație, cel putin cand ne referim la arhitectura oricărei aplicații sau oricărui site web.   
      Decizia de a realiza o aplicație web a pornit în urma perioadei de stagiu într-o companie românească, la care am descoperit demersurile și necesitățiile principale pentru realizarea unei astfel de aplicații. În plus, pe perioada stagiaturii, am înțeles care pot fi diferențele între cele 2 părți ale unei aplicații și cum pot fi dezvoltate astfel încât sa fie păstrate convențiile stabilite la nivel global pentru acestea.  
        
      Dorința de a pune bazele unei astfel de aplicații a fost accentuată si mai tare la următorul loc de muncă, la care am descoperit cu adevărat ce înseamnă o aplicație web și de la care am „furat” idei si cunoștiințe, oferindu-mi toate resursele necesare pentru a construi din toate punctele de vedere aplicația ce urmează a fi prezentată. Aici am învățat cum creez un proiect, cum îl configurez, cum îl structurez și cu ce ustensile și framework-uri pot sa lucrez pentru a obține o aplicație web cât de cât profesionist realizată.  
      Idea pe baza căreia am început dezvoltarea a venit dintr-o necesitate generală pe care majoritatea oamenilor o manifestă și care vine sub forma unui mic asistent în momentul în care suntem indeciși, avem dubii sau poate chiar nu știm care este varianta favorabilă pentru ce ne dorim in momentul acela.  
        
      Tema aleasă este incadrată turismului, combinat cu social-media, deoarece putem spune că ideea aplicației se bazează pe rezolvări și răspunsuri ce se încadrează turismului, dar oferind un mediu prietenos de socializare între utilizatori.
   2. Prezentarea generală, scopul și motivația alegerii temei  
      Aplicația web pentru căutarea destinațiilor de vacanță adresată grupurilor de turiști, este o aplicație bazată pe nevoia utilizatorilor și dorințelor lor, astfel încât acestora să li se pună la dispoziție un număr mare de opțiuni pentru a alege cât mai ușor o destinație turistică.   
      Idea de aplicație pentru generarea unor destinațîi turistice a pornit de la nevoia comună pe care o au oamenii atunci când doresc să calatorească sau să petreacă o vacanță în anumite locații pe care fie nu le-au vizitat vreodată, fie despre care nu au auzit până în acel moment, dar care au o șansă destul de mare să fie pe placul acestora.   
        
      Cu ce vine totuși diferit această aplicație față de altele care sunt deja cunoscute și existente pe piață de mult timp? Ei bine, aplicația această spre deosebire de aplicațiile existente pentru generarea unor sugestii bazate pe interesele și opțiunile alese de utilizatori, vine în plus cu un mecanism prin care le este oferită posibilitatea utilizatorilor să creeze grupuri prin care să poată invită alți oameni să participe la generarea unor destinațîi turistice prin selectarea unor interese și opțiuni dorite. Mecanismul vine în ajutorul familiilor, grupurilor de prieteni, excursiilor școlare, taberelor și multor altor adunări de cel puțîn 2 oameni în care există o posibilitate foarte mare că cel puțîn o persoană să aibă interese diferite și să își dorească vizitarea altor atracții turistice, dar totuși să se poată ajunge la un consens între membrii grupurilor pentru alegerea celor mai favorabile destinații turistice încât să ii poată mulțumi pe aproape toți participanții.   
        
      Pe lângă mecanismul principal în jurul căruia fost construită ideea acestei aplicații, mai există și opțiunea de a creea un set de destinațîi turistice pentru o singură persoană, dar asupra căruia nu vor putea fi adăugate și alte persoane pentru modelare. De asemenea, pe pagină principala vor există mereu destinații turistice sugerate automat de către server pentru utilizatori, în funcție de setările pe care aceștia le vor face în pagină de administrare a contului.  
        
      Fiecare grup dispune de o pagină modală de administrare a grupului, prin care administratorii desemnați vor putea invită alți utilizatori să se alăture grupului și vor putea să dea afară utilizatori nedoriți, iar oricare membru al grupului va avea posibilitatea de a resetă lista de interese aleasă și de a părăși grupul.  
      Un lucru foarte important fără de care am considerat că nu poate există un grup pe această aplicație, a fost un chat pe care l-am alocat fiecărui grup în parte, făcând posibilă o comunicare deschisă între membri, să poată discuta cât mai ușor despre destinațiile și interesele dorite și alese.   
        
      Pentru a putea gestiona invitațiile primite, a fi înștiințați de grupurile din care au fost dați afară și cererile de prietenie primite, utilizatorilor le este pus la dispoziție un sistem de notificări, care este actualizat de fiecare dată când aceștia reîncarcă orice pagină de pe aplicație.  
        
      În cazul în care utilizatorii doresc să revadă atât grupurile cât și sugestiile generate din acestea, sau drumețiile pe cont propriu, fiecare dispune de o pagină în care sunt memorate toate acestea și sunt accesibile pentru a fi vizitate. Totuși, în cazul în care un utilizator nu mai face parte dintr-un grup sau a abandonat crearea drumeției pe cont propriu, acesta nu va mai putea accesa paginile acestora, ci doar vor fi înștiințați că au făcut parte din ele.   
        
      Înainte de a putea utiliza aplicația, utilizatorii vor fi nevoiți să își creeze un cont pe baza căruia vor fi stocate toate interesele acestora.
   3. Contribuția proprie in realizarea lucrării  
      Realizarea lucrării a fost făcută pe cont propriu, astfel toate apoartele asupra arhitecturii, modul de organizare a structurii, separarea codului în multiple fișiere, decizia legată de limbajul folosit pentru partea de server, cât și framework-urile aferente, limbajul pentru front-end cât și librăriile utilizate, au fost alese bazându-mă pe cunoștiintele pe care le dețineam până la momentul începerii acestui proiect.   
        
      Ulterior proiectul a suferit modificări majore, de la structura gândită inițial, până la regândirea interfeței pe partea de client. Cel mai semnificativ lucru a fost implementarea unor design pattern-uri pentru modelarea proiectului într-o manieră eficientă și ușor de modificat sau alterat ulterior.
   4. Structura lucrării  
      În momentul începerii aplicației, această a fost construită pe baza a două proiecte C# și un proiect backup în care era ținută structura bazei de date și alterată cu fiecare modificare adusă acesteia.   
      Proiectul principal reprezenta serverul aplicației și totodată partea de client, fiind un serviciu web bazat pe conexiuni REST API și construit folosindu-mă de modelul arhitectural „Model-View-Controller” pentru a permite apelarea requesturilor HTTP de către client către partea de server. Framework-ul principal pentru dezvoltarea serviciului web a fost .NET Core 2.2, asupra căruia au fost setate diferite configurări pentru cuplarea corectă cu partea de client. De asemenea, pentru partea de front-end, am utilizat framework-ul AngularJs împreună cu Javascript pur pentru funcționalitate, dar împreună cu framework-ul C# Răzor pentru a putea folosi diferite metode ale serverului în interiorul interfeței.   
        
      Acest proiect a fost secționat pe mai multe foldere precum Controllers, Models, Views, AppViews, AppStyle, App, Images, Scripts, Workers acestea fiind asociate următoarelor lucruri:  
                  - Controllers -> folderul principal unde se află toate fișierele în care se regăsesc clasele intermediare dintre client și server, acestea oferind prin diferite metode accesul către server prin requesturi HTTP.  
                  - Models -> aici se regăsesc toate entitățile folosite că intermediari pentru preluarea unor date ce vin în requesturi către server sau utilizate că răspuns pentru metode pentru a trimite obiectul json copie al entității către partea de client.  
                  - Views -> reprezintă zona principala unde se regăsesc fișierele Răzor cu extensia CSHTML, ele fiind la baza de fapt fișiere HTML, dar care permit utilizarea sintaxelor C# în acestea. Acest folder este obligatoriu să existe, deoarece framework-ul .NET Core împreună cu arhitectură MVC, asociază toate fișierele controller din folderul Controllers cu view-urile aferente din acest folder, folosind convenția de nume în care folderul are numele controllerului, iar fișierul CSHTML are denumirea index.  
                  - AppViews -> în acest folder există toate fișierele HTML create pentru interfață aplicației, secționate în funcție de zonele de care aparțîn. Aceste fisirele sunt utilizate în interiorul unui fișier global CSHTML care reprezintă părintele aplicației la nivel de interfață, astfel aplicația se încadrează categoriei de o site cu o singură pagină, deoarece orice trecere către alt pagină, de fapt reprezintă alt html ce înlocuiește cel curent fără a apela un nou DNS request.  
                  - AppStyle -> folderul în care stilizarile css ale paginilor html se regăsesc, secționate pentru a putea fi diferențiate în funcție de nevoie.  
                  - App -> reprezintă folderul vital pentru funcționarea corectă a interfeței, deoarece aici se află toate fișierele javascript cu angularjs, fișiere ce au rolul de controllere, request-uri către server, fișiere ajutătoare ce conțîn metode reutilizabile, directive și fișiere de constante. În cazul în care un fișier html nu ar avea asociat un fișier controller js, nu ar avea nicio funcționalitate și ar ffi un fișier static.  
                  - Images -> așa cum spune și numele folderului, acesta conține imagini utilizate pe interfață. Acestea au fost salvate local, deoarece erau criticale, iar pierderea lor ar fi stricat interfață, dar există și imagini ce sunt încărcate din mediul online, acestea fiind suplimentare și fără să fie nevoie de un backup.  
                  - Scripts -> folder în care există librării și framework-uri externe precum cel de angularjs, bootstrap pentru css, librării adiționale pentru angularjs sau javascript, evitând astfel legături către librării din mediul online în cazul în care ar există o problema de conexiune. De asemenea, sunt utilizate și librării externe, dar care sunt suplimentare și nu pot cauza o discrepanța.  
                  - Workers -> toate funcționalitatea serverului, mai precis metodele și variabilele claselor ce oferă funcționalitatea serverului, erau păstrate în acest folder, dar nu orice funcționalitate, ci doar cea în care erau folosite metodele principale ce se vor regăsi în proiectul ce urmează să fie prezentat mai jos.  
        
      Proiectul secundar poartă denumirea de DataLayer, zona unde sunt executate toate comenzile principale către baza de date, cât și query-uri mai complexe, dar toate având că scop interacțiunea cu baza de date, precum metode de înserare, ștergere, modificare și retragere a datelor din baza de date. Pe lângă aceste funcționalități, tot în acest proiect se regăseau toate interfețele, modelele și clasele utilizate în proiectul principal.  
      Specificat și anterior, aplicația folosește o baza de date gratuită de la Microsoft, mai precis SQL Server, instalată pe calculatorul local. Pentru a putea funcționa aplicația, această verifică la runtime dacă conexiunea către baza de date poate fi realizată conform unui text ce conține conexiunea către sql server.  
        
      Pe parcursul modelării aplicației, această a suferit schimbări majore, printre care și reorganizarea proiectelor, astfel că proiectul de DataLayer a fost împărțit în 3 proiecte sub denumirile Domain, DataLayer, Helpers. Asemenea proiectului principal i-au fost aduse schimbări ce au rezultat în separarea de proiectul principal a functionalitatilor din folderul Workers ce au fost duse într-un nou proiect denumit BusinessLogic. Formarea acestei separări a venit de-a lungul procesului meu de învățare la compania la care am lucrat pe parcursul dezvoltării acestei aplicații, unde am înțeles nevoia unei mai bune și eficiente organizări, astfel am aplicat design pattern-ul ce poartă denumirea de “Repository Pattern”. Acest pattern presupune consolidarea următoarei arhitecturi:  
                  - Crearea proiectului Domain în care vor fi ținute toate modelele/entitățile utilizate pentru maparea datelor din sau spre baza de date, temporare pentru diferite prelucrări sau diferite clase în care sunt ținute constante sau enum-uri.  
                  - Crearea proiectului Helpers în care vor fi ținute toate clasele ce conțîn metode ajutătoare, mai precis metode și clase statice sau diferite constante ce vor fi refolosite fie în metodele ajutătoare, fie în restul aplicației.  
                  - Păstrarea proiectului DataLayer în care rămân doar interfețele și clasele cu metodele primare ce accesează baza de date pentru modelarea tabelelor sau preluarea datelor din ele.  
                  - Crearea proiectului BusinessLogic ce conține doar logică de prelucrare a datelor venite prin controller, utilizând toate ustensilele puse la dispoziție de proiectele Domain, Helpers și DataLayer.  
        
      Folosind pattern-ul menționat, permite aplicației să fie scalabila și poată fi create și alte servicii web precum proiectul principal, care la rândul lor să poată utiliza clasele, metodele și variabilele din proiectele ajutătoare.  
        
        
        
        
      Pentru conexiunea către baza de date, am ales să utilizez framework-ul “Entity Framework” care mi-a fost recomandat de ceilalți colegi de la locul de muncă, dar pe care îl cunoșteam la rândul meu și știam exact cum trebuie implementat. Avantajele acestui framework este simplitatea sintaxei și posibilitățile oferite pentru prelucrarea datelor, folosindu-mă de metode predefinite în acest framework ce utilizau în spate query-uri optimizate către SQL Server. Pe lângă Entity Framework, am fost nevoit să utilizez și librăria de .ADO NET, o librărie ce permite crearea unei conexiuni directe cu baza de date și interogarea acesteia prin query-uri de SQL create manual. Motivul principal pentru care am ales să adaug și această modalitate, a fost imposibilitatea de a lucra simultan cu două conexiuni de entity framework pornite în paralel, motiv pentru care această rezultă mereu într-o excepție aruncată.  
        
      Ulterior, proiectul a migrat de la versiunea .NET Core 2.2 către .NET Core 3.1, deoarece această venea cu îmbunătățiri și solutionari la diferite probleme ale framework-ului. Odată cu această modificare, au apărut și modificări la nivelul configurarii făcute în proiectul principal, care au forțat și schimbarea anumitor parametrii ale unor metode din fișierele controller existente.
   5. Repere istorice și rezultate cunoscute  
      În trecut oamenii planificau excursiile în funcție de locațiile de care auziseră sau văzuseră în poze, sau chiar în funcție de zonele pe care deja le vizitaseră. Rareori se întâmplă că cineva să meargă într-o locație fără să știe un minim de detalii despre această, astfel multe atracții turistice ale pământului rămâneau nevizitate. Evoluția tehnologiei a început să acopere aceste lipsuri și să creeze noi aplicații web sau disponibile pentru orice tip de dispozitiv, că oameni să se poată ajută de ele pentru a fi informați despre zone turistice existente, în care puteau să vizualizeze poze cu peisaje, să descopere tradițiile, climatul, modul de transport și multe alte repere turistice legate de locația aleasă. Aplicații Web precum Booking, Trivago și multe altele, au pornit acest concept prin care expun utilizatorilor diferite locații și oferă informații referitoare la condițiile și beneficiile pe care le vor avea turiștii.   
        
      Deși la nivel de turist sunt prezentate diferite recomandări, personal nu am descoperit o aplicație web care să prezinte recomandări pentru grupuri de oameni, care să își pună problema dacă aceștia au individual recomandări diferite și vor totuși să obțînă sugestii care să fie plăcute pentru toate taberele. Lipsa unui astfel de mecanism limitează foarte mult grupurile de oameni, mai ales în perioada sărbătorilor în care site-urile de traveling au un trafic imens pentru că familii și grupuri de prieteni doresc să plece în diferite locații, desigur aceștia având cel mai probabil interese diferite.  
        
      Aplicația această acoperă lipsul menționat, dar vine și ea cu lipsuri ce sunt acoperite de celelalte aplicații menționate, precum lipsa unor cazări și rezervarea acestora, achiziționarea unor modalități de transport către acele locații și numărul limitat de atracții turistice menționate, deoarece capacitatea de date stocată este limitată la posibilitatea personală de a găși și adaugă referințe turistice, în comparație cu aplicațiile celelalte care dispun de un grup de oameni ce completează aceste informații.
2. Tehnologii
   1. Tehnologii Back-End  
      Alegerea tehnologiilor pentru partea de back-end a aplicatiei a fost relativ intuitiva, deoarece majoritatea dintre acestea imi erau familiare la momentul inceperii proiectului, avand experienta precedenta cu ele la locul de munca. Desi le cunosteam si stiam la ce sunt utile, provocarea a venit in momentul in care a trebuit sa le integrez, sa le configurez si sa le utilizez cat mai eficient. Pentru realizarea aplicatiei, am decis sa folosesc IDE-ul Visual Studio Community 2017, ulterior migrand la Visual Studio Community 2019, fiind cel mai recomandat pentru dezvoltarea aplicatiilor web in C# in conditiile in care acesta era si este in continuare gratuit, destinat pentru studenti si proiecte personale.  
      Serverul este creierul unei aplicatii, iar acesta trebuie sa fie cat mai bine organizat si sa se foloseasca de cat mai multe lucruri benefice pentru a obtine o performanta cat mai buna, evitarea a cator mai multe exceptii si formarea unei scalabilitati eficiente pentru marirea numarului de functionalitati si diminuarea greselilor de buna practica pentru limbajul C#.  
        
      Asa cum am mentionat si anterior, serverul este bazat pe limbajul C#, iar framework-ul principal pentru a fi posibila crearea unui server web, este .NET Core. Prima versiune pe care am abordat-o si utila la momentul acela, a fost versiunea 2.2, deoarece era ultima disponibila si aparuta, dar si compatibila cu toate framework-urile utilizate, si desigur cea mai potrivita din punct de vedere al optimizarilor aduse framework-ului si rezolvarilor de erori pe care le avusese. Desi existau alte alternative precum framework-ul .NET Standard 4.5, acesta nu era gratuit si open-source pentru a fi accesibil codul oricarui dezvoltator, facand limitat accesul spre imbunatatiri si idei ce ar fi putut crea din acesta un framework cautat si usor de utilizat. In plus, comunitatea persoanelor ce utilizau .NET Standard era destul de restransa in comparatie cu cea pentru .NET Core, astfel solutiile si parerile referitoare la acesta erau putine si de multe ori, fara un raspuns la anumite probleme.  
      Pentru a putea compila solutia pentru partea de back-end, au fost instalate fisierele SDK si Runtime ce apartineau de .NET Core si aferente versiunii utilizate. Fisierul SDK este utilizat pentru a avea acces la librarii si ustensile oferite de framework-ul .NET Core pentru a putea realiza aplicatia sau cel putin partea de back-end si conexiunea catre partea de front-end, dar in special permite compilarea solutiilor. Fisierul Runtime este utilizat pentru a permite rularea aplicatiilor in urma compilariilor facute, fara de care nu ar putea fi accesibile spre utilizare.  
        
      Datorita framework-ului .NET Core, am utilizat librarii precum Microsoft.AspNetCore.Http si Microsoft.AspNetCore.Http.Abstractions pentru a putea crea sau primi request-uri HTTP, fie prin intermediul conexiunii cu partea de client, fie prin alte modalitati precum programul Postman, pentru depanarea problemelor si/sau a codului implementat. Programul Postman este extrem de utilizat in randul programatorilor, utiliatatea sa primara fiind creearea si interograrea oricarui tip de request HTTP catre un server anume pe baza unor multiple aspecte, precum IP-ul sau DNS-ul calculatorului pe care se afla serverul, port-ul asociat acestuia, protocolul utilizat pentru request (GET,POST,PUT,DELETE,UPDATE) si multe alte configurari de acces atribuite in mod normal unui request HTTP. Astfel, cu ajutorul acestui program am putut vizualiza daca serverul receptiona apelurile facute si returna mesajul dorit sau daca exista vreo problema ce ar fi putut periclita buna functionare, dar in plus ofera o alternativa in cazul in care nu exista partea de front-end sau nu este posibil accesul catre aceasta, deoarece simuleaza un request ce ar putea fi trimis de catre client catre server.   
      Pe langa cele mentionate, cele doua librarii ofera posibilitatea realizarii unor clase middleware, ce mai poarta numele de “omul in mijloc” sau “clasa intermediara”, aceasta avand scop interceptarii request-urilor inainte ca acestea sa ajunga la controllerele existente si la functionalitatea serverului, in mod normal pentru a putea fi validate si/sau prelucrate.  
        
      Un alt framework foarte important pe care l-am utilizat este Entity Framework si este un cadru de mapare obiect-relațional open source pentru ADO.NET, fara de care nu ar fi fost posibila aproape toata conexiunea si interogarea bazei de date. Acesta face permite crearea conexiunii catre baza de date in functie de un string de conexiune. Pentru recunoasterea tabelelor, framework-ul recunoaste automat maparea catre acestea in functie de o conventie de nume utilizata in cod pe care o atribuim unor seturi de obiecte ce se leaga fiecare de cate un tabel. In general pentru interograrea unei baze de date sunt folosite scripturi sau query-uri SQL, dar entity framework foloseste query-uri predefinite ce nu sunt accesibile pentru modificare, dar optimizate si reutilizabile pentru multiple tipuri de interogari. Query-urile vin sub forma unor metode c#, facand mai simpla implementarea acestora fara a stii in mod direct limbajul SQL pentru a putea intelege cum sunt prelucrate datele intr-o baza de date relationala. Astfel framework-ul este recunoscut pentru simplitatea sintaxei si usurintei pentru configurarea acestuia. In ultimul rand, framework-ul ofera o ustensila ce are ca scop crearea unor fisiere de migrare pentru crearea automata a bazei de date si a tabelelor, in functie de clasele create si mapate, reducand timpul si necesitatea mentenantei bazei de date, dar si portabilitatea acesteia pe baza unui back-up.   
        
      Cu toate acestea, personal nu am utilizat aceasta ustensila, ci am folosit o ustensila a IDE-ului prin care baza de date a fost replicata in interiorul folderului proiectului, dar pe care am updatat-o treptat folosindu-ma de aceeasi unealta. Folosind aceasta varianta de replicare si back-up, pot vizualiza codul sursa pentru generarea intregii baze de date si alterarea acesteia fara a impacta codul serverului, deoarece uneori maparile nu se schimba, dar apar mici modificari insesizabile asupra bazei de date, iar aceasta modalitate permite modificarea bidirectional.  
        
        
        
      O tehnologie care completeaza un pattern pe care l-am implementat ulterior si datorita careia am redus semnificativ utilizarea memoriei, este framework-ul Ninject, pe care l-am folosit impreuna cu pattern-ul Composition Root pentru a implementa o clasa reutilizabila si in acelasi timp singletone, utilizata pe baza depedency injection, astfel incat aceasta leaga orice implementare a claselor “worker” sau “repository”, fara a aloca mereu o noua adresa de memorie datorita instantierii continue atunci cand aveam nevoie de utilizarea unor metode din cadrul acestora. Astfel folosind acest framework, am legat orice interfata de clasa in care era implementata si am facut posibila injectarea automata a implementarii in diferite workere, prin intermediul constructorului worker-ului specific. Utilizand acest pattern, se va aloca de la lansarea proiectului in memorie spatiu pentru clasa Composition Root, impreuna cu implementarile instantiate, pe care o vom injecta mereu prin request-uri catre controllerele aferente ce o vor folosi, urmand ca in interiorul acestora sa utilizez implementarile mentionate doar in cazul in care e nevoie de acestea si fara a aloca spatiu suplimentar in memorie.   
      Acest pattern este recunoscut in cadrul multor companii si utilizat foarte mult in proiecte, deoarece ofera o buna organizare si o usoara utilizare a logicii din spatele oricarui server.  
        
      Anterior am prezentat tehnologia folosita pentru a conecta serverul la baza de date si a putea prelua, insera si altera seturi de date din aceasta, aceasta avand numele Entity Framework. In continuare vorbesc despre SQL Server, baza de date aleasa pentru stocarea informatiilor aplicatiei. Alegerea a venit ca urmare a utilizarii acesteia si pe parcursul anilor universitari pentru diferite proiecte scolare, cat si pentru proiecte personale, ca ulterior sa fie aprofundata la stagiul dintre anii II si III, si mai apoi in locurile de munca ce au mai urmat. MS SQL Server este o baza de date relationaal oferita de Microsoft, ce vine atat in varianta gratuita, cat si contra cost, si care poate fi accesata atat prin intermediul unor comenzi in terminalul sistemului de operare, dar si prin aplicatia pusa la dispozitie de catre acestia ce poarta denumirea de Sql Server Management Studio. Aplicatia ofera o interfata vizuala pentru prelucrarea bazei de date si configurarea sa oferind o usoara interactiune cu aceasta. Personal am lucrat si cu alte baze de date relationale, precum MySQL, PostgreSQL, Oracle, cat si nerelationale cum ar fi MongoDB sau ElasticSearch, dar aceasta ofera avantajul unei conexiuni mult mai usoare cu orice aplicatie ce este realizata pe baza framework-ului .NET Core. De asemenea, IDE-ul Visual Studio ofera o ustensila mentionata anterior pentru crearea locala a unui back-up pentru structura bazei de date, sau pentru migrarea sa. Din acest motiv si adaugand experienta anterioara, am decis ca ma voi folosi de ea pentru realizarea intregii aplicatii, optand pentru varianta gratuita, deoarece oferea majoritatea optiunilor pe care le detinea si varianta contra cost.  
        
      Pe langa toate tehnologiile mentionate, am mai incercat integrarea si a altor framework-uri, care ulterior s-au dovedit a fi ineficiente pentru tipul acesta de aplicatie si care nu ar fi adus un plus de performanta sau de scalabilitate.
   2. Tehnologii Front-End  
      Tehnologiile pentru partea de client au reprezentat o provocare, deoarece ma consider o persoana ce indrageste si stapaneste mai mult partea de server si logica a unui proiect, astfel realizarea interfetei a fost un impediment initial si a reprezentat un timp de gandire si research, ca intr-un final sa descopar framework-ul AngularJs pentru Javascript. Initial acesta a fost destul de greu de inteles, sintaxa si conceptele sale nu erau prietenoase cu o persoana incepatoare, dar avantajul a fost usurinta integrarii, documentatiile si solutiile propuse de comunitatea ce dezvolta in limbajul C# partea de server si la randul lor au putut realiza aplicatii web folosindu-se de AngularJs. Astfel o perioada de timp a constat in acomodarea cu acest framework, incercarea de a-l intelege si a gandi o arhitectura pentru a fi cat mai scalabila si eficienta.   
      Folosind acest framework, nevoia de a folosi cod Javascript pur a disparut, dar fundamentele acestuia au ramas si la nivelul acestui framework, deoarece sintaxa si algoritmica ce stateau la baza framework-ului erau aceleasi ca la Javascript. Pot spune ca Javascript nu imi era necunoscut, deoarece l-am utilizat inca din perioada liceului si chiar si pe perioada facultatii la anumite materii pentru realizarea proiectelor sau la laboratoarele acestora. Adaugand suplimentar, o persoana ce nu ar fi cunoscut fundamentele Javascript-ului nu ar fi avut o intelegere usoara pentru acest framework si cel mai probabil ar fi fost aproape imposibil de integrat, deoarece asa cum spuneam si anterior, conceptele propuse de AngularJs sunt destul de complexe si necesita o intelegere mai buna.  
      AngularJS spre deosebire de utilizarea propriu-zis a limbajului Javascript pur, ofera dinamicitate interfetei si permite o conexiune cu paginile HTML extrem de rapida. In plus, posibilitatea creeri unor validari si alterari ale interfetei in timp real fac ca acesta sa fie o alegere buna pentru acest tip de aplicatie web.   
        
      Interfata paginilor web a fost realizata folosind HTML si CSS, avand un impuls mai mare datorita framework-ului Bootstrap, pe care l-am utilizat pentru prima data in primul an de facultate pentru realizarea unui proiect personal si pe care ulterior am ajuns sa il refolosesc la orice alta aplicatie web pe care am dezvoltat-o sau o dezvolt si in prezent, deoarece acesta ofera o multitudine de stilizari predefinite in CSS pentru a crea o interfata receptiva si cu un aspect cat mai placut. Cu toate acestea, stilizarile oferite de Boostrap nu erau suficiente si am fost nevoit sa recurg la implementari aditionale in CSS pentru crearea unor stilizari sau alterarea celor oferite de framework.   
        
      Pe parcursul proiectului au aparut foarte multe fisiere ce contineau cod de AngularJs sau cod CSS, facand imposibila depanarea problemelor direct din consola pentru dezvoltatori din toate browse-rele, astfel am folosit o ustensila in Visual Studio ce compila pe baza unui fisier de configurare toate fisierele de AngularJS sau JS selectate, intr-un singur fisier principal, iar pentru CSS aceeasi procedura, doar ca fisierele de CSS trebuiau separate in mai multe fisiere pentru a nu suprapune denumiri identice de stilizari.  
      Folosind aceasta metoda minimalista, calitatea depanarii pentru diferite probleme intampinate in timpul dezvoltarii a fost sporita si a redus din timpul alocat.
3. Perspectiva utilizatorului  
   alocat.