UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL TEHNOLOGIA INFORMAȚIEI

Platformă de evaluare rapidă pentru programe de internship

TIE – Tool for Internal Evaluation

Lucrare de licentă

Absolvent: **Sergiu-George Zăgrean**

Coordonator științific: **Asis. Ing. Cosmina Ivan**

2016

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DECAN,
Prof .dr. ing. Liviu Miclea

Director departament,
Prof. dr. ing. Rodica Potolea

Absolvent: **Sergiu-George Zăgrean**

Platformă de evaluare rapidă pentru programe de internship

1. **Enunțul temei:** Proiectul își propune realizarea unui sistem menit să faciliteze întreg procesul de evaluare a unor posibili candidați în vederea angajării în candrul unei companii. Pentru eficientizarea și scăderea timpului necesar procesului de recrutare, sistemul ofera o alternativa la metodele clasice și își propune să ofere posibilitatea de management rapid al testelor, corectarea si acordarea rapida a calificativelor și viualizarea eficientă a rezultatelor și a statisticilor.
2. **Conținutul lucrării:** Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză si Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare, Validare si Evaluare, Concluzii, Bibliografie, Anexe.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Tehnologia Informației
4. **Consultanți:** Asis. Ing. Cosmina Ivan
5. **Data emiterii temei:** 1 Noiembrie 2015
6. **Data predării:** 30 Iunie 2016

Cuprins

Capitolul 1. Introducere

1.1 Contextul proiectului

În societatea actuală, în contextul real de business, se pune din ce în ce mai mult accent asupra programelor de internship sau de inițiere într-un anumit domeniu. Datorită faptului că majoritatea programelor de inițiere sunt organizate în aceeași perioadă, cererile sunt tot mai mari, iar resurele sunt insuficiente, ceea ce poate duce la întârziere în luarea unor decizii. Este vital ca o companie să fie capabilă să facă față unui număr mare de cereri și în același timp să poată veni cu o soluție sau concluzie în cel mai scurt timp, astfel acest lucru poate duce la pierderea candidaților potriviți pentru companie.

Proiectul propus se încadrează în contextul mai larg al platformelor de evaluare. O platformă de evaluare este o aplicație (utilizată de regulă în spațiul Web sau intranet) complexă, creată cu scopul testării cunoștințelor acumulate într-un domeniu. Platformele de evaluare prezintă un trend ascendent pe piața din România, ele fiind utilizate nu doar în scop educațional, dar mai ales în domenii precum medicia, inginerie și multe altele. Practic, aproape că nu există domeniu al vieții științifice ori sociale, în care evaluarea cu ajutorul unei aplicații – platformă să nu fie prezentă. Ele pot fi utilizate în cadru organizat, în interiorul unor grupuri de lucru, cu un scop comun sau individual.

Sistemul prezentat este o platformă de evaluare rapidă a cărui obiectiv principal este să optimizeze procesul de recrutare a candidaților în vederea obținerii unei poziții într-o companie, școală, asociație sau în alte entități similare. Eficientizarea procesului este realizată prin susținerea computerizată a testelor, evaluare rapidă și centralizare a informațiilor. Procesul de recrutare presupune evaluare cunoștințelor candidaților, ordonarea și clasificarea acestora, precum și contactarea lor în vederea comunicării și a stabilirii unor decizii. Beneficiarii acestui sistem sunt personalul tehnic care este responsabil cu elaborarea modelelor de teste și cu corectarea testelor susținute de către candidați precum și personalul departamentului de resurse umane care este responsabil cu organizarea evenimentelor de recrutare, cu programarea candidațiilor la examen și cu comunicarea ulterioară a rezultatelor către candidați.

1.2 Motivația

Ținând cont de timpul necesar realizării tuturor activităților de recrutare și a resurselor fizice necesare, apare nevoia unui sistem care să înglobeze aceste activități într-un sistem compact, precum și să ușureze și să scurteze timpul necesar pentru derularea acestor activități. Urmărind inițial majoritatea platformelor de evaluare existente pe piață, se constată cu ușurință că acestea sunt destinate pentru învățarea și evaluarea unor grupuri stabile de candidați, cum ar fi clasele de elevi sau cursanți, necesitând un efort suplimentar pentru gestionarea conturilor fiecărui candidat

în parte, crearea unor clase sau grupuri de candidați, etc. Acest tip de platforme nu sunt potrivite pentru procesele de recrutare, unde fluxul de candidați este foarte mare, iar în majoritatea cazurilor un candidat este evaluat o singură data. Crearea și gestionarea unor conturi pentru un număr mare de candidați, precum și creare unor grupuri și adăugarea fiecărui candidat în parte la un grup (clasă) de care aparține, ar însemna un efort suplimentar și inutil în procesul de recrutare. Folosirea unei astfel de platforme ar însemna crearea a unui număr foarte mare de conturi și grupuri de conturi, care în majoritatea cazurilor ar fi folosite o singură data, după care acestea din urmă vor fi eliminate din sistem. Acest lucru nu duce la eficientizarea procesului de recrutare, ci din potrivă, la îngreunarea acestuia. Este nevoie de o platformă de evaluare rapidă, unde fiecare candidat este responsabil cu introducerea datelor personale înainte de susținerea testului, cu un mecanism simplu și rapid de corectare a testelor și de acordare a calificativelor, precum și posibilitatea de vizualizare a unui clasament, efectuarea unor căutări sau sortări candidaților după anumite criterii.

1.3 Conținutul lucrării

În următoarele paragrafe se va prezenta structura lucrării pe capitole, precum și o scurtă descriere pentru fiecare din ele.

Capitolul 1 – Introducere – Acest capitol conține o scurtă descriere a contextului problemei și a motivației care au dus la dezvoltarea aplicației prezentate.

Capitolul 2 – Obiectivele Proiectului – În acest capitol sunt descrise și prezentate obiectivul principal și obiectivele secundare care au fost propuse spre implementare pentru sistemul prezentat.

Capitolul 3 – Studiu Bibliografic – În acest capitol sunt prezentate principalele aspecte și concepte ale dezvoltării aplicațiilor web. Tot în acest capitol, este prezentată o comparație a sistemului care s-a realizat și a sistemelor similare disponibile pe piață.

Capitolul 4 – Analiza si Fundamentare Teoretica – Acest capitol conține o descriere a tehnologiilor utilizate în dezvoltarea sistemului precum și motivele pentru care acestea au fost alese, și conceptele folosite în dezvoltare. Tot în acest capitol sun prezentate cerințele funcționale, cerințele non-funcționale și cazurile de utilizare ale sistemului, cele mai semnificative fiind prezentate într-un mod mai detaliat.

Capitolul 5 – Proiectare de Detaliu și Implementare – În acest capitol va fi prezentat modul în care sistemul a fost proiectat. Se va prezenta schema arhitecturală a sistemului, arhitectura bazei de date și structura tabelelor, diagrama de deployment și vor fi detaliate anumite componente ale aplicației considerate relevante în înțelegerea a felului în care sistemul este proiectat.

Capitolul 6 – Testare, Validare și Evaluare – Acest capitol conține o prezentare a principalele procese de testare care au fost realizate asupra sistemului propus. Tot în acest capitol

sunt prezentate rezultatele evaluării interfeței grafice a aplicației de către un grup de utilizatori țintă.

Capitolul 7 – Manual de Instalare și Utilizare – În acest capitol sunt descrișii pașii care trebuie urmați pentru instalarea cu succes a componentelor sistemului pe o mașină locală sau pe un server într-o rețea locală. Tot în acest capitol vor fi prezentate resursle software și hardware necesare rulării, precum și un manual de utilizare a aplicației.

Capitolul 8 – Concluzii - În acest capitol sunt aduse concluzii asupra sistemului dezvoltat. Se vor trecere în revistă realizările și obiectivele care au fost atinse prin acest proiect, urmată de o descriere a posibilităților de dezvoltare ulterioară.

Capitolul 2. Obiectivele Proiectului

În acest capitol sunt descrise obiectivele propuse pentru sistemul realizat. Scopul sistemului este de a ușura și a eficientiza procesul de recrutare în cadrul companiilor, prin centralizarea activităților de creare, susținere și corectare a testelor, și prin capabilitatea de a efectua căutări, sortări și filtrări asupra informațiilor.

2.1 Obiectivul principal

Obiectivul principal al acestui proiect este definirea și implementarea unui sistem care să ofere suport și să eficientizeze procesele de recrutare din cadrul companiilor în vederea reducerii resurselor materiale și a timpului necesar acestor procese. Acest obiectiv principal al proiectului poate fi îndeplinit prin realizarea obiectivelor secundare care vor fi prezentate în cele ce urmează.

2.2 Obiective secundare

2.2.1 Obiective generale

Eficiența în utilizare este unul din cele mai importante obiective propuse pentru acest proiect. Această caracteristica este realizată prin oferirea unei interfețe "prietenoase" userului (user-friendly interface) care să fie ușor de înțeles, învățat și utilizat. Caracteristicile unei interfețe "prietenoase" sunt[1] :

- Simplă - o interfață ușor de utilizat, nu este prea complexă, dar în schimb este simplă, oferind acces rapid la funcții sau comenzi comune.
- Curată – o interfață utilizator bună este bine organizată, făcând ușoară localizarea diferitelor instrumente și opțiuni
- Intuitivă – pentru a fi ușor de utilizat, o interfață trebuie să aibă sens pentru utilizatorul mediu și ar trebui să necesite explicații minime pentru modul în care trebuie folosită.

Al doilea obiectiv general care se dorește a fi realizat în aplicația prezentată este securitatea. Se dorește evitarea situațiilor în care un candidat sau o altă persoană neautorizată să aibă acces la resursele care trebuiesc protejate, vizualizându-le sau modificându-le. Acest lucru se dorește a fi realizat prin autentificare și autorizare.

Al treilea obiectiv general al acestui proiect este realizarea unui sistem care să fie extensibil, adică ușor de supus unor dezvoltări ulterioare prin modificarea sau adăugarea de funcționalități noi. În general, niciun sistem nu este complet, tot timpul acesta poate fi îmbunătățit sau extins, și trebuie să țină pasul cu schimbările care au loc în timp.

[1] User-friendly Definition - http://techterms.com/definition/user-friendly

Sistemul trebuie să fie capabil să ofere posibilitatea de gestiune asupra întrebărilor, secțiunilor, templateurilor, userilor și a sesiunilor de testare, și posibilitatea de susținere, corectare și vizualizare a testelor, precum și sortarea și filtrarea acestora. Sistemul trebuie să poată fi utilizat de mai multe timpuri de utilizatori. Tipurile de utilizator în această aplicație vor fi administratorul, hr și candidatul. Fiecare dintre acești utilizatori trebuie să beneficieze de anumite funcționalități pe care sistemul trebuie să le ofere. Tipurile diferite de utilizatori implică acces diferit la resursele sistemului, unul din obiectivele secundare generale propuse este ca sistemul să aibă capacitatea de filtrare a resurselor disponibile în funcție de tipul de utilizator care le solicită.

2.2.2 Managementul intrebărilor

Sistemul trebuie să fie capabil de a adaugă, modifica, vizualiza și șterge întrebările. De asemenea, sistemul trebuie să valideze faptul că atunci când se dorește ștergerea unei întrebări, această să nu fie asignata la o secțiune, altfel să interzică utilizatorului ștergerea întrebării. Pe lângă conținutul și tipul întrebării, sistemul trebuie să mai fie capabil de a stoca și o imagine explicativă pentru o întrebare, dacă este nevoie.

2.2.3 Managementul secțiunilor

Sistemul trebuie să fie pe lângă efectuarea operațiilor CRUD asupra secțiunilor, să asigneze întrebări unei unei secțiuni specifice sau să elimine întrebări de la o secțiune. Sistemul trebuie să mai conțină o validare care va permite ștergerea unei secțiuni, doar dacă aceasta nu este asignată nici unui template, în caz contrar ștergerea secțiunii să nu fie posibilă.

2.2.4 Managementul templateurilor

Pe lângă capabilitatea de a efectua operații CRUD pe templateuri, sistemul trebuie să fie capabil de a asigna secțiuni unui template, și de a seta o pondere pentru fiecare secțiune, pondere care ulterior va fi folosită pentru calcularea notei finale. Fiecare template trebuie să aibă asociat un timp de testare specificat în ore și minute, și sesiunea de testare de care aparține. Se poate specifica care template va fi folosit în testarea candidaților prin specificarea lui că template implicit pentru sesiunea de testare respectivă. Fiecare sesiune de testare va avea câte un template implicit. Sistemul trebuie să conțină două validări în cazul templateurilor: un user nu trebuie să fie capabil să șteargă un template dacă acesta a fost deja folosit pentru testarea candidaților, iar a doua validare nu trebuie să permită setarea de două templateuri default pentru aceiași sesiune de testare.

2.2.5 Acordare/vizualizare review-uri

Un user care folosește sistemul trebuie să fie capabil să vizualizeze sau să adauge un review pentru teste, în cazul în care acestea nu au unul deja. Pentru eficientizare, testele vor fi vizualizate în funcție de sesiunea de testare în care au fost susținute. Tot pentru eficientă, prima dată ar trebui afișate testele pentru care nu a fost deja acordat un calificativ, iar apoi cele

corectate. Sistemul va mai conține capabilitatea de căutare unui anumit test după numele candidatului sau a templateului, de a configura numărul de teste vizibile pe o pagină și alte caracteristici care duc la eficientizarea procesului.

2.2.6 Vizualizare rapoarte

Sistemul trebuie să furnizeze capabilitatea de a genera rapoarte în funcție de calificativul testului, sesiunea de testare sau dată, să permită ordonarea testelor după anumite atribute cum ar fi anul de studiu și universitatea (dacă este cazul), nota finală, sesiunea de testare și să ofere posibilitatea de a vizualiza fiecare test în detaliu și de a contacta un candidat. Aplicația va mai fi capabilă să afișeze statistici cum ar fi distribuția notelor pentru o anumită sesiune de testare.

2.2.7 Gestionare sesiuni de testare

Un alt obiectiv al sistemului este posibilitatea de a gestiona sesiunile de testare. Pe lângă adăugarea și ștergerea acestora, va exista un mecanism de activare/dezactivare acestora, astfel încât în momentul în care o sesiune de testare este dezactivată, nu se vor putea adăugă sau susține teste pentru aceasta. Sistemul va conține și un mecanism care nu va permite ștergerea unei sesiuni de testare, dacă pentru acestea există templateuri definite sau teste susținute.

2.2.8 Gestionare useri

Administratorul aplicației va fi capabil de a adăuga noi useri cu oricare din rolurile administrator, hr sau candidat, să șteargă sau să dezactiveze useri.

Capitolul 3. Studiu bibliografic

3.1 Aplicaţiile Web

Aplicaţiile web moderne sunt sisteme software complexe, iar dezvoltarea acestora necesită o abordare metodologică. Similar cu proiectarea aplicaţiilor software, proiectarea web implică utilizarea unei abordări sistematice şi cuantificabile pentru realizarea specificaţiilor, implementării, operaţiilor şi întreţinerii aplicaţiilor web de calitate superioară. Din punct de vedere al istoricului dezvoltării şi complexităţii distingem anumite tipuri de aplicaţii web: orientate pe documente, interactive, tranzacţionale, cu trăsături ale web-ului semantic. Cerinţele particulare ale proiectării aplicaţiilor web rezultă din caracteristicile lor speciale din sfera produselor software, precum şi din dezvoltarea şi utilizarea lor. World Wide Web are o influenţă enormă şi permanentă asupra vieţii noastre. Economia, industria, educaţia, sănătatea, administraţia publică, divertismentul – majoritatea componentelor vieţii noastre au fost pătrunse de World Wide Web. Motivul acestei omniprezenţe constă în special în natura web-ului, caracterizată prin disponibilitatea globală şi permanentă dar şi prin accesul omogen la informaţiile distribuite la nivel global sub forma paginilor web.

Deşi iniţial web-ul a fost proiectat ca un mediu pur informaţional, în prezent el evoluează într-un mediu al aplicaţiilor. Aplicaţiile web de astăzi sunt sisteme software complexe care oferă servicii interactive şi personalizabile accesibile prin intermediul diferitelor dispozitive; ele oferă posibilitatea realizări tranzacţiilor între utilizatori şi de obicei stochează datele într-o bază de date. Elementul distinctiv al aplicaţiilor web comparativ cu aplicaţiile software tradiţionale este modul în care este utilizat web-ul: tehnologiile şi standardele sale sunt utilizate ca o platformă de dezvoltare şi ca platformă utilizator în celaşi timp. O aplicaţie web poate fi definită astfel: O aplicaţie web este un sistem software bazat pe tehnologiile şi standardele consorţiului World Wide Web (W3C), care oferă resurse web specifice (conţinut şi servicii) prin intermediul unei interfeţe utilizator numită browser web. Această definiţie include în mod explicit tehnologiile şi interacţiunea cu utilizatorul. De aici putem deduce că tehnologii precum serviciile web nu sunt aplicaţii web, dar pot fi o parte a acestora, iar siturile web lipsite de componente software (cum sunt paginile HTML statice) nu sunt considerate aplicaţii web.

3.2 Single Page Application

Una dintre cele mai noi tendințe în dezvoltarea web actuală o reprezintă Aplicațiile pe o singură pagină (Single Page Application, SPA). SPA este o aplicatie web sau un website care este incadrat intr-o siugura pagina cu scopul de a furniza o experienta user mult mai fluida, similara cu a unei aplicatii desktop. Marea diferență dintre o *SPA* și o aplicație web standard o constituie faptul că în cadrul celei din urmă fiecare navigare la o altă pagină necesită o generare completă a paginii, adică fiecare pagină este complet (re)încărcată de pe serverul web. În cadrul

unei SPA întreg codul necesar (HTML, CSS si JavaScript) este încărcat o singură dată, atunci cand site-ul este accesat pentru prima data, dupa care nu mai este niciodată încărcată complet o pagină de pe server. Site-ul va avea mai multe pagini (de ex. "Home", "About", "Contact" etc.), dar acestea sunt încărcate (împreună cu diverse date) ca fragmente HTML, fără a se reîncărca complet o pagină.

Interfața single-page web este compusă din componente individuale care pot fi actualizate sau înlocuite independent, astfel încât nu trebuie reîncărcată întreaga pagină la fiecare acțiune a unui user. Există câțiva termeni cheie care ajută la descrierea SPA:

- **Web interface** – folosită pe web, se concentrează pe interfețele utilizator
- **Componente individuale** – Este divizată in componente mai mici care interacționează untele cu altele
- **Actualizări și înlocuiri** – o componenta poate fi oricând actualizată sau înlocuită cu o altă componenta
- **Reîncărcare** – niciodată nu este reîncărcată întreaga pagină, chiar dacă va fi încărcat conținut nou în unele secțiuni
- **Acțiuni user** – SPA este responsabilă cu gestionarea acțiunilor utilizatorului cum ar fi input de la mouse sau tastatură

Un beneficiu este faptul că o SPA este mult mai rapidă decăt o aplicație standard: nu trebuie mereu ca o pagină să fie încărcată complet când este accesată. Prin aceasta, o SPA se aseamănă mai mult cu o aplicație desktop: tranzitia paginiilor este realizata mai ușor, fără a aștepta de fiecare dată ca serverul web să reîncarce toate resursele unei pagini înainte de a o reda.

Alt beneficiu mai puțin vizibil, este faptul că mare parte din funcționalitatea site-ului este executată pe partea de client și nu pe partea de server (cum este într-o aplicație web standard) și deci, timpul de așteptare pentru un răspuns al serverului este mai scurt. Partea de client este realizată în JavaScript pur sau cu ajutorul unor frameworkuri pe bază de Javascript.

Exista o varietate mare de framework-uri disponibile precum EmberJs, Knockout SPA, Facebook React sau Angular JS. Una dintre tehnologiile de bază folosite in dezvoltarea aplicației web a fost AngularJS, un framework open source dezvoltat și menținut de Google. Trăsătura lui caracteristică o constituie faptul că extinde HTML-ul cu etichete și proprietăți noi. Cu ajutorul directivelor se pot crea elemente HTML noi.

3.3 Url, rute și stări

În dezvoltarea aplicațiilor web, structura URL (Uniform Resource Locator) este un aspect fundamental de considerat. Pentru a furniza un URL partajabil și funcțional este important că URL-ul să fie unic pentru starea curentă a aplicației. Dacă același URL ar fi folosit pentru mai multe stări ale aplicației, acesta nu ar fi accesibil din bara de adrese a browserului, lucru care ar fi nepractic dacă userul dorește să stocheze sau să distribuie URL-ul curent. Aplicația trebuie să fie capabilă să mapeze URL-ul la o anumită stare, lucru care este adesea realizat cu ajutorul unei

componente de rutare. Această componenta primește un URL de la user pe care îl decodează într-o anumită stare, astfel încât componentele corespunzătoare sunt inițializate și prezentate la user. Componenta de rutare trebuie să fie capabilă să schimb URL-ul, dacă starea aplicației s-a schimbat. HTML5 furnizează un history API care permite unui client JavaScript să schimbe URL-ul la runtime fără a reîncărca pagină[2].

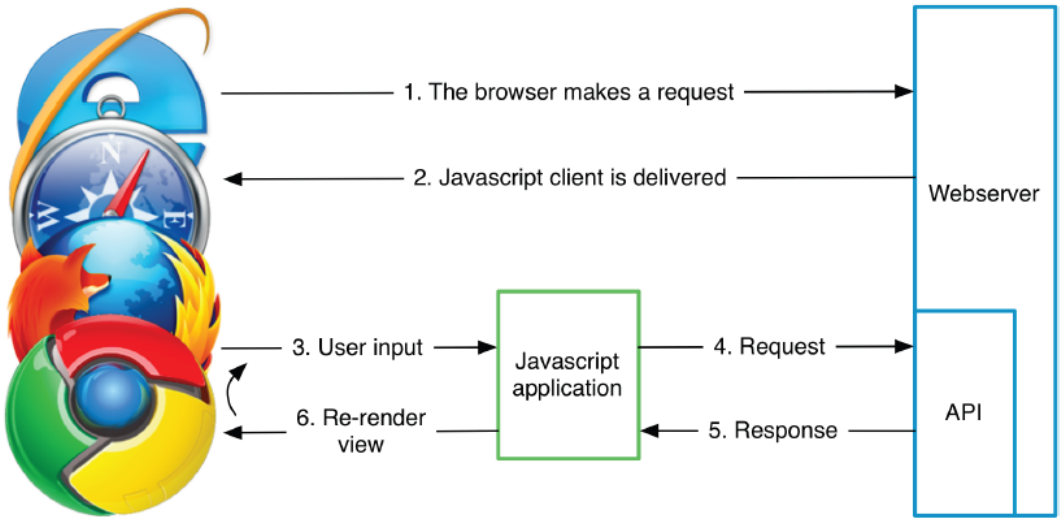

Fig. 3.1 Comunicarea între browser și web server

3.4 Legăturile de date (data bindings)

Legăturile de date sunt folosite pentru a lega un set de date la un view corespunzător. One-way data bindings permite unui view să fie randat din nou în mod automat când datele se schimbă. Unele framework-uri, precum și cel folosit în aplicația prezentată, AngularJS, suportă two-way data binding care permite de asemena ca datele să fie actualizate când au loc schimbări în view . Această reprezintă o tehnică puternică, deoarece toată logica din spatele unui view este administrată de către framework, ceea ce face munca mult mai ușoară pentru un dezvoltator de aplicații web. Important de menționat este faptul că nu este eliminată logica și complexitatea, ci aceasta este doar mutat în framework. Totuși, un framework trebuie să suporte o varietate mare de elemente HTML, chiar dacă acestea un sunt folosite în aplicație, lucru care poate afecta performanța aplicației în unele cazuri.

[2] W3C, HTML5 - A vocabulary and associated APIs for HTML and XHTML, 2012-03-29 http://www.w3.org/TR/2012/WD-html5-20120329/history.html#history [2016-06-07]

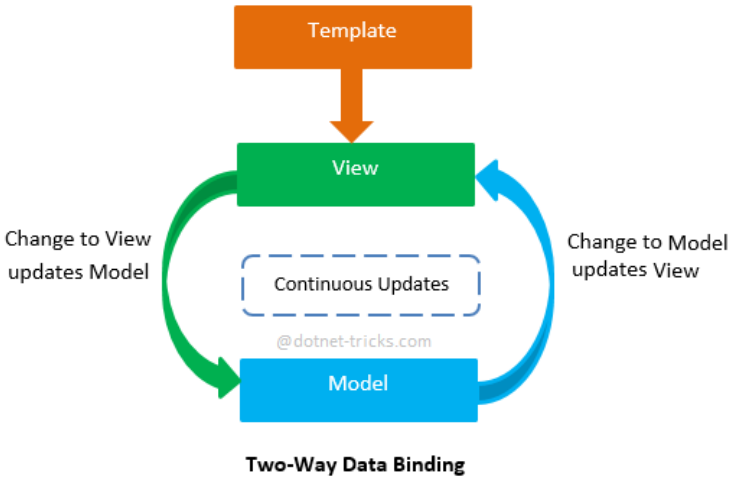

Fig. 3.2 AngularJS - Two-way data binding

3.5 Securitatea aplicațiilor web

Securitatea în mod fundamental se referă la protejarea bunurilor. Bunurile pot fi obiecte tangibile precum o pagină web sau o bază de date sau pot fi obiecte mai puțin tangibile precum reputația unei companii. Securitatea este o cărare, nu o destinație. Pe măsură ce este analizată infrastructura și aplicația, se pot identifica potențiale amenințări de securitate și se constată ca fiecare amenitare prezintă un grad de risc. Securitatea se referă la managementul riscurilor și la implementarea unor contramăsuri eficiente.Securitatea se bazează pe următoarele elemente[3]:

- Autentificarea – este procesul de identificare unică a clienților aplicației web sau a serviciilor. Autentificarea răspunde la întrebarea "Cine ești?"
- Autorizarea – este procesul care reglementează resursele și operațiunile la care un user autentificat are dreptul de acces. Resursele pot include fișiere, baze de date, tabele, înregistrări din tabele, etc, împreună cu resursele de sistem cum ar fi cheile registru și datele de configurare. Autorizarea răspunde la întrebarea "Ce poți face?"
- Confidențialitate – este procesul de a ne asigura că datele rămân private și confidențiale, și că nu pot fi vizualizate de către utilizatorii neautorizați sau de către persoanele care monitorizează fluxul de trafic în rețea. Criptarea este frecvent utilizată pentru a pune în aplicare confidențialitatea. Listele de control al accesului (ACL-uri) sunt un alt mijloc de aplicare a confidențialității

[3] Microsoft – Web Application Security Fundamentals
https://msdn.microsoft.com/en-us/library/ff648636.aspx#c01618429_002 [2016-06-15]

- Integritatea – este garanția faptului că datele sunt protejate de modificarea accidentală sau deliberată. Integritatea datelor în tranzit este de obicei asigurată folosind tehnici de hashing și coduri de stare a mesajelor
- Disponibilitatea – din perspectiva securității, disponibilitatea înseamnă că sistemul rămâne disponibil pentru userii legitimi. Scopul atacatorilor care folosesc atacuri DOS (Denial Of Service) este să prăbușească aplicația sau se asigure că aplicația este copleșită de request-uri astfel încât să nu mai fie disponibilă pentru useri.

3.6 Sisteme similare

Sistemul propus se încadrează în contextul mai larg al platformelor de evaluare. O platformă de evaluare este o aplicație (utilizată de regulă în spațiul Web) complexă creata cu scopul testării cunoștințelor acumulate într-un domeniu.

Platformele precum Proiectul INSAM, TAO, GoConqr sau ProProfs pun accentul pe gestionarea mai strictă și detaliată a candidaților testelor, făcută de către administratori (aplicații destinate mai mult profesorilor). De asemenea unele din aceste tool-uri oferă și suport pentru învățare online. Aplicația web prezentată folosește o abordare diferită, unde fiecare dintre candidați este responsabil să furnizeze datele personale înainte de începerea testului. Această decizie a fost luată deoarece, în programele de inițiere/internship, rulajul de candidați este mult mai mare decât într-o clasă obișnuită. Astfel, organizatorii procesului de selecție nu vor pierde timp esențial cu introducerea fiecărui candidat în parte, sau cu administrarea lor. De asemenea, aplicația nu se adresează unui anumit grup stabil de candidați, deoarece ei diferă de la un sesiune de testare la alta.

3.6.1 Proiectul INSAM

Proiectul INSAM[4] (Instrumente digitale de ameliorare a calității evaluării în învățământul preuniversitar) are ca obiectiv dezvoltarea și implementarea de instrumente și mecanisme digitale de îmbunătățirea a proceselor evaluative și de autopoziționare/autoevaluare a elevilor din învățământul preuniversitar liceal.

Proiectul vizează sistemul de învăţământ preuniversitar, cu o focalizare specială pe nivelul liceal, unde nevoile de evaluare/autoevaluare/autopoziţionare sunt mai acute, din perspectiva diversificării traseelor de învăţare şi a oportunităţilor existente la final de ciclu. Proiectul urmăreşte să abordeze aspecte problematice legate de procesele evaluative prin oferirea unui sistem informatic performant de evaluare accesibil tuturor participanţilor la procesul de educaţie (elevi, cadre didactice) şi stimularea acestora în procesul unei educaţii adaptate nevoilor pietei muncii si societatii bazate pe cunoastere. INSAM vine sa sustina aceste deziderate prin modernizarea şi diversificarea modului de furnizare a educaţiei de la nivel preuniversitar, prin

[4] Formare specialisti in evaluare INSAM – din cadrul poriectului INSAM
https://ticlimbimoderne.wikispaces.com/file/view/Suport_curs_formare_INSAM_2.pdf [2016-06-18]

utilizarea resurselor şi serviciilor digitale care vor constitui o baza pentru dezvoltarea rutelor flexibile de învăţare şi pentru asigurarea calităţii procesului educational.

3.6.2 GoConqr

GoConqr[5] este o platformă de evaluare care combină beneficiile instrumentelor de învățare cu sprijinul unei comunități active. Fiecare user poate să creeze conținut de învățare și teste pentru diferite domenii de activitate și să le partajeze cu alți useri. Userii pot face parte din grupuri de interes comun, unde pot avea discuții, partaja conținut și își pot evalua cunoștințele. Fiecare user trebuie să își seteze nivelul de experiență Student, Educator sau Professional, fiecare nivel având mai multe caracteristici după care grupurile sau conținutul este filtrat. GoConqr permite atât crearea de conturi noi cât și logarea cu conturi de Google+ sau Facebook.

3.6.3 TAO

TAO[6] este o platformă open-source profesională pentru evaluare atât în domeniul educației cât și pentru angajarea în sectorul public. Tao oferă flexibilitatea de a administra testele de pe un laptop, un server, sau din cloud. Datorită faptului că folosește o bază de date de tip NoSql, Tao suportă până la 100,000 de useri concurenți. Tao poate fi folosit și de pe tablete sau smartphone-uri, ajutând administratori testelor să salveze resurse implementând soluții BYOD (Bring your own device) de testare.

Acest sistem oferă posibilitatea de a definii roluri și responsabilități pentru useri, de a definii și stoca teste, de a programa teste online, și de vizualizare eficientă a rezultatelor și a rapoartelor.

3.6.4 PEDb 2.0

Platforma PEDb 2.0[7] este un sistem computerizat de evaluare psihologică construit după principiul utilizării mai multor metode și a mai multor surse de informare (multi-method & multi-informant assessment), principiu considerat în cercetarea științifică actuală drept standardul de aur în evaluarea copiilor şi adolescenților. PEDb 2.0. este o aplicație software ce integrează o serie de teste psihologice computerizate (etalonate și validate pe populația românească) și resurse pentru promovarea sănătății mintale, foarte utile pentru:

- Evaluarea, prevenția și remedierea problemelor de sănătate mentală;
- Consiliere psihologică;
- Orientare școlară și profesională la copii și adolescenți.

Această platformă oferă posibilitate de gestionare a subiecților și a subiectelor de testare, precum și vizualizarea de rapoarte și statistici.

[5] GoConqr - https://www.goconqr.com/en/info/about-us/
[6] TAO Data Sheet - http://www.taotesting.com/wp-content/uploads/2016/01/tao_datasheet_lowres.pdf
[7] PEDb 2.0 - http://www.cabinetpsihoterapieiasi.ro/evaluare-psihologica/platforma-de-evaluare-a-dezvoltarii-pedb-2-0-67-19-ani/

3.6.5 ProProfs

ProProfs[8] este un sistem profesional de evaluare online permite atât crearea de teste, cât și de training-uri, sondaje, studii și chiar și jocuțe care stimulează gândirea precum puzzle-uri, trivia sau joc de cuvinte încrucișate. Această platformă poate fi folosită pentru o gamă largă de domenii deoarece nu se adresează unui domeniu anume, putând fi adaptată pentru orice tip de activitate de învățare și testare cum ar fi testări periodice, evaluare la angajare, training-uri în cadrul companiilor, stimularea gândirii și a logicii, etc.

3.6.6 Comparație

În tabelul de mai jos este prezentată o comparație a sistemelor similare descrise mai sus și a sistemului prezentat, comparație făcută după anumite criterii precum capabilitatea de a genera clasamente sau de a seta anumite constrângeri pentru un test.

[8] ProProfs – About - http://www.proprofs.com/about.shtml

Funcționalitate\Platforma	Proiectul INSAM	TIE	TAO	PEDb 2.0	GoConqr	ProProfs
Aplicație Web	X	X	X		X	X
Aplicație Desktop			X	X		
Definire/modificare/ștergere teste	X	X	X	X	X	X
Generarea de rapoarte	X	X	X	X	X	X
Gestionare evaluari	X	X	X	X	X	X
Generare clasamente	X	X		X	X	X
Cotare automata a raspunsurilor	X		X	X	X	X
Cotare manuala a raspunsurilor	X	X				X
Suporta mai multe tipuri de intrebări (true and false, checkboxes, blank spaces, raspuns text, etc.)	X		X		X	X
Manage and Control User Access	X	X	X	X		X
Mobile Applications (Android and/or iOS)			X		X	X
Export rezultate	X		X			X
Import întrebari/teste	X		X			
Constrângeri de timp	X	X	X		X	X
Suport pentru e-learning (knowledge base, flashcards, trainings, courses) – una sau mai multe variante	X				X	X
Open Source			X			
Firends/Chat					X	X
Creare sondaje de opinie					X	X
Logare folosind conturi din rețelele sociale (Google+, Facebook, Twitter, etc.)					X	X

Tabel 3.1 Comparatie - sisteme similare

Capitolul 4. Analiză şi Fundamentare Teoretică

În acest capitol vor fi prezentate tehnologiile care au contribuit la implementarea aplicației web precum si motivele pentru care acestea au fost alese. Printre acestea se numără: ASP.NET Web API, .Net Entity Framework, Open Web Interface for .Net (OWIN) and Katana, etc. Se descriu detaliile considerate necesare în vederea întelegerii codului sursă, cu referire către literatura de specialitate, care detaliaza tehnologiile enumerate. Tot în acest captitol mai sunt prezentate cerințele functionale si cele non-functionale ale sistemului si cazurile de utilizare ale acestuia.

4.1 Tehnologii și concepte utilizate pentru dezvoltarea aplicației

4.1.1 ASP.NET Web API

Web API este un framework care face ușoara construirea serviciilor HTTP care pot fi consumate de o gamă largă de clienți, incluzând browserele, telefoanele mobile, tablete, etc. ASP.NET Web API este o platformă ideală pentru construirea aplicațiilor RESTful în .Net.

Termenul API provine de la "Application Programming Interface".În lumea web development-ului, termenul API este sinonim cu servicii web online pe care aplicațiile client le pot folosi pentru a extrage și a actualiza date. Fără un API, într-o arhitectură de bază, fiecare aplicație client ar trebui să aibă prorpia logică de business incorporată. Web API oferă o arhitectură cu un API central care încorporează toată logică de business.

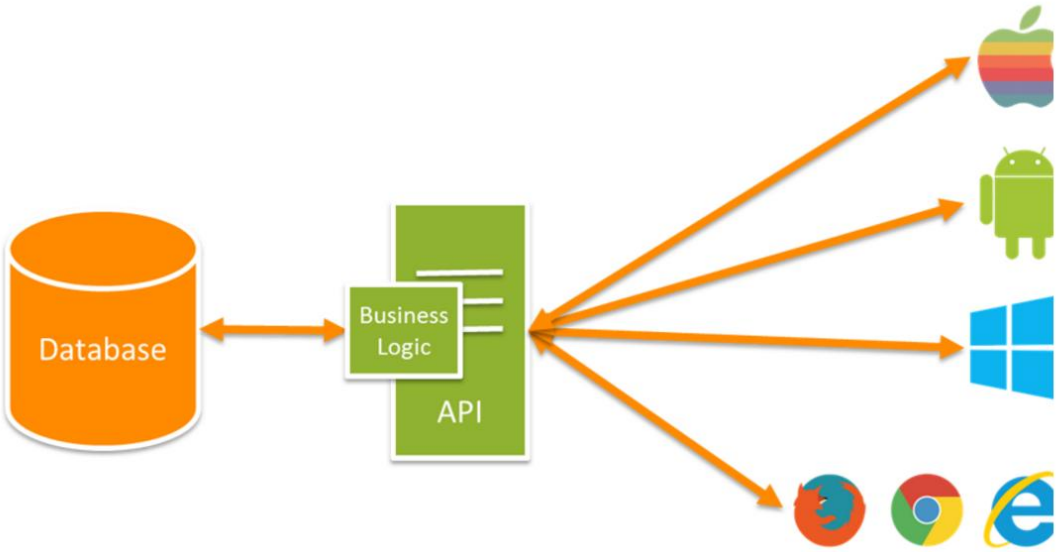

Fig. 4.1 Fluxul de date în Web API

Web API este construit pe modelul pipeline .net modular, la care pot fi conectate componente independente, cum ar fi componente pentru securitate (OWIN Katana). În momentul în care un server care găzduiește serviciile web api primește un request, acesta îl pasează mai întâi prin pipeline-ul .net pentru requesturi. Acest lucru permite adăugarea cu ușurință a propriilor module în cazul în care capacitățile implicite nu sunt suficiente pentru nevoile aplicației.

Acest framework folosește conceptele de Controller și Action. Resursele sunt mapate direct la controlere. În mod tipic, există câte un controller pentru fiecare entitate de date (User, Question, etc.). Web API folosește mecanismul de rutare .net pentru a mapa controlerele la URL-uri. Acțiunile sunt folosite pentru a maparea la diferite verbe specifice HTTP, cum ar fi Get sau Post.

Tehnologia Web API a fost aleasă din următoarele motive:

- Funcționează pe baza protocolului HTTP. Acest lucru înseamnă că poate fi apelata și consumata de orice fel de aplicație precum aplicații mobile, desktop sau web.
- Această tehnologie centralizează logica de business, care duce la diminuarea eforturilor în implementarea mai multor clienți. Ca rezultat, informația legată de business poate fi menținută într-un mod consistent
- Folosește date de dimensiuni reduse (XML, JSON), ceea ce îl face ideal în special pentru aplicațiile mobile și web. Cu toate acestea, poate folosii și alte tipuri de date dacă este nevoie precum fișiere binare, documente, fișiere video sau imagini
- Dezvoltarea cu Web API în sine presupune o separare de UI, clientul poate fi schimbat cu ușurință

4.1.2 ADO.Net Entity Framework

Entity Framework este un framework ORM (Object/Relațional Mapping) care permite programatorilor să manipuleze date relaționale precum obiecte specifice domeniului aplicației, eliminând nevoia codului „greoi” pe care programatorii sunt de obicei nevoiți să îl scrie. Utilizând Entity Framework, dezvoltatorii emit interogări folosind LINQ (Language-Integrated Query) pentru a prelua și manipula datele ca obiecte puternic tipizate. Acest ORM furnizează servicii cum ar fi change tracking, identity resolution, lazy loading și query translatation, astfel încât dezvoltatorii să se poată concentra pe logica de business a aplicației, nu pe fundamentele de acces la date.

Există trei abordări pentru folosirea EF:

- Database first approach – crearea Entity Data Model, a contextului și a claselor entitate dintr-o bază de date existentă. Entity Data Model este actualizat ori de câte ori schema bazei de date este modificată. Oferă suport pentru proceduri stocate, vederi, etc.
- Model first approach – crează entitățile, relațiile și ierarhiile acestora direct din utilitarul de proiectare EDMX, după care generează baza de date din acest model.

- Code first approach – evită folosirea utilitarului vizual pentru modelare EDMX. Se scriu în cod clase POCO (Plain Old CLR Objects), după care se crează baza de date din aceste obiecte. Această abordare va fi folosită în aplicația prezentată.

Fig. 4.2 Arhitectura conceptuală Entity Framework

- Entity Client – Abstract DBMS Providers pentru Entity Framework
- Object Services – contextul obiectului (Entity Container, Object Querry) si serviciile obiectului (Entity Connection, Metadata Workspace, Object State Manager)
- Entity Data Model – se ocupă cu entitațiile si relațiile pe care acestea le folosesc, set de obiecte care descriu structura datelor de business și sunt mapate la baza de date

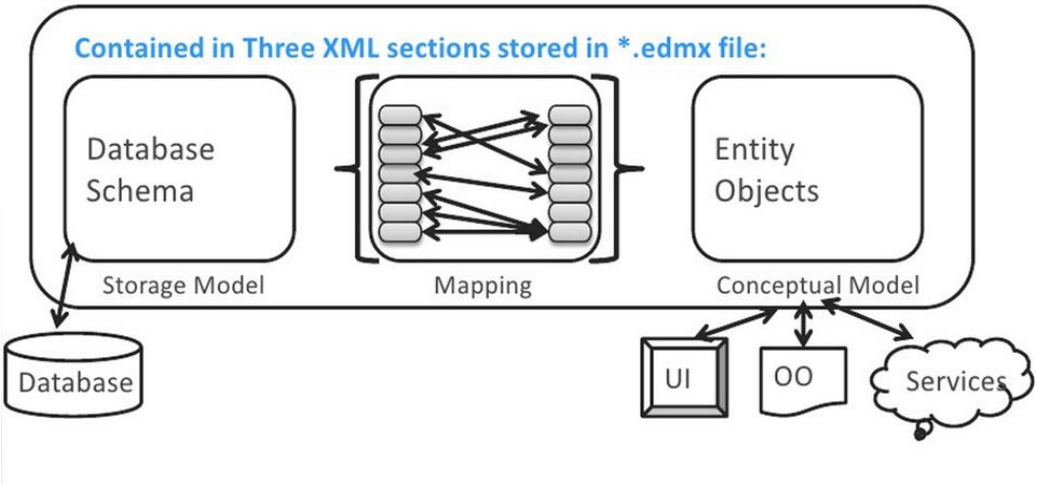

Fig. 4.3 Fluxul de date in Entity Framework

Entity Framework a fost ales deoarece oferă o posibilitate rapidă și eficientă de a construi layerul pentru acces la date. Este independent de baza de date, ceea ce înseamnă că nu trebuie scris cod specific unei baze de date. Totodată, simplifica manipularea datelor, are grijă de dependențele dintre tabele și construiește query-uri complexe, oferă mentenanța tranzacțiilor și a conexiunilor, oferă suport pentru concurență. În concluzie, este ușor de întreținut și crește productivitatea.

4.1.3 OWIN și Katana

Owin (Open Web Interface for .Net) definește o interfață standard între serverele web .Net și aplicațiile web. Rolul interfeței Owin este de a decupla serverul de aplicație, și de a încuraja dezvoltarea modulelor .net web development, și, fiind un standard deschis, stimulează ecosistemul open source a uneltelor .net pentru dezvoltare web.

Katana este un set de componente open source folosite pentru dezvoltarea și găzduirea aplicațiilor web bazate pe owin, întreținute de Microsoft Open Technologies Group. Katana furnizează o implementare a specificației Owin, o varietate mare de componente middleware ready-to-use, care pot fi folosite în aplicațiile web bazate pe specificația Open Web Interface for .Net. De asemenea oferă și o modalitate standardizată pentru a gestiona autentificarea.

Fluxul request/response acționează ca un pipeline, unde requesturile trebuie să treacă prin diferite componente middleware înainte să ajungă la aplicație. Există posibilitatea ca requestul să nu ajungă niciodată la aplicație, dacă o componentă middleware răspunde în locul ei (de exemplu o componentă middleware pentru autentificare).

Fig. 4.4 Owin request pipeline

Unul dintre principalele motive pentru care a fost aleasă această tehnologie este faptul că ea decuplează serverul web de frameworkul aplicației. Deși acesta este un principiu simplu, are o implicație mare: deoarece serverul este decuplat de frameworkul aplicației, se poate folosi aplicația cu diferite servere web sau alte componente.[9] Caracteristicile care definesc OWIN sunt:

[9] Understanding Owin and Katana – CodeProject
http://www.codeproject.com/Articles/826757/Understanding-OWIN-and-Katana [2016-06-12]

- Flexibilitate – avem opțiunea de a alege între diferite componente și de a înlocui ușor o componenta cu alte componente. Se pot folosi combinații diferite de componente astfel încât să se potrivească nevoilor proiectului
- Portabilitate – Owin permite conectarea componentelor existente cu noi componente care nu sunt restricționate la un anumit furnizor. Se poate înlocui hostul, serverul sau middleware-ul în orice moment este nevoie
- Eficiența – Owin are o arhitectură ușoară: furnizează capabilitatea de a adăuga și folosi caracteristicile de care avem nevoie. Deoarece există componente diferite pentru a îndeplini responsabilități specifice, se pot folosi doar componentele de care este nevoie, evitând deținerea componentelor care nu vor fi folosite niciodată.

4.1.4 Angular JS

AngularJS este un framework structural pentru aplicații web dinamice. Acesta permite folosirea HTML-ului ca un limbaj șablon și permite extinderea sintaxei HTML pentru exprimarea componentelor aplicației clar și succint. AngularJS manipulează combinația de cod DOM și AJAX într-o structură bine definită, cod care altfel ar trebui scris de mână. Acest framework furnizează tot ce este necesare pentru construirea unei aplicații web CRUD (Create, Read, Update, Delete): Data-binding, directive pentru templateing, form validation, rutare, componente refolosibile și dependency injection.

Un prim avantaj al acestui framework este conceptul de “data-binding”. Acesta reprezintă sincronizarea automată a datelor între model si componentele unei vederi. AngularJS permite dezvoltatorului să trateze modelul ca un obiect primar, ce o să fie mai apoi expus în view. Orice modificare care se face pe model, va fi reflectată si pe vedere. Acest lucru este posibil folosind Two-Way Data Binding. În acest caz, prima dată, șablonul (template-ul HTML), care reprezintă codul HTML nerandat, este randat in browser. Imediat după randare, se produce o vedere live (live view), acesta fiind o reprezentare directă a datelor. Orice modificare care apare în vedere, se reflectă imediat si asupra modelului sau invers. Din cauza că vederea este doar o proiecție a modelului, controlerul este total separat de vedere și nici nu știe de existența acestuia. Acest lucru duce la decuplarea componentelor in structura aplicatiei si ajută mult la testare [10].

În Angular JS, un controler este definit de o funcție JavaScript constructor. Când un controler este ataşat DOM-ului, folosind directiva ng-controller, AngularJS va inițializa un obiect nou de tipul Controller, folosind funcția de construcție specifică [11]. Serviciile furnizate de AngularJS reprezintă obiecte substituibile legate împreună folosind injecția de dependințe. Acestea pot fi instantiate doar in momentul in care sunt utilizate (lazy instantiated); AngularJS va instanția serviciul numai în momentul în care exista o componentă care depinde de ea sau

[10] https://docs.angularjs.org/guide/databinding
[11] https://docs.angularjs.org/guide/controller

instante unice (singletons) - fiecare componentă dependentă de un serviciu primește o referință către o singură instanță garantată de fabrica de servicii [12].

AngularJS folosește serviciul $http, un serviciu de bază, ce facilitează comunicarea între un server HTTP remote, folosind JSONP. Acest serviciu este o funcție ce primește un singur argument, un obiect din configurații, ce este folosit pentru a genera o cerere HTTP și returnează o promisiune. Răspunsul primit are următoarele proprietăți [13]:

- data - corpul răspunsului transformat
- status - codul de răspuns HTTP
- headers
- config - obiectul de configurații folosit pentru generarea cererii
- status text - statusul răspunsului HTTP (acesta poate să fie între 200-299, în cazul în care cerera a fost cu success sau orice altă valoare, reprezentând o eroare)

AngularJS se folosește de servicii REST expuse de serverul de back-end pentru a primii și tansmite datele. Acest framework se folosește de protocolul HTTP pe baza portului 8080, protocolul primar pentru comunicarea dintre un server și un browser, pentru a consuma serviciile. Se folosesc metodele GET/POST/PUT pentru cereri de resurse, sau cereri speciale cum ar fi salvarea sau editarea. La baza cererilor HTTP stă conceptul de request-response, unde clientul face o cerere (request) și serverul îi răspunde (responde).

Arhitectura AngularJS este prezentată în figura de mai jos:

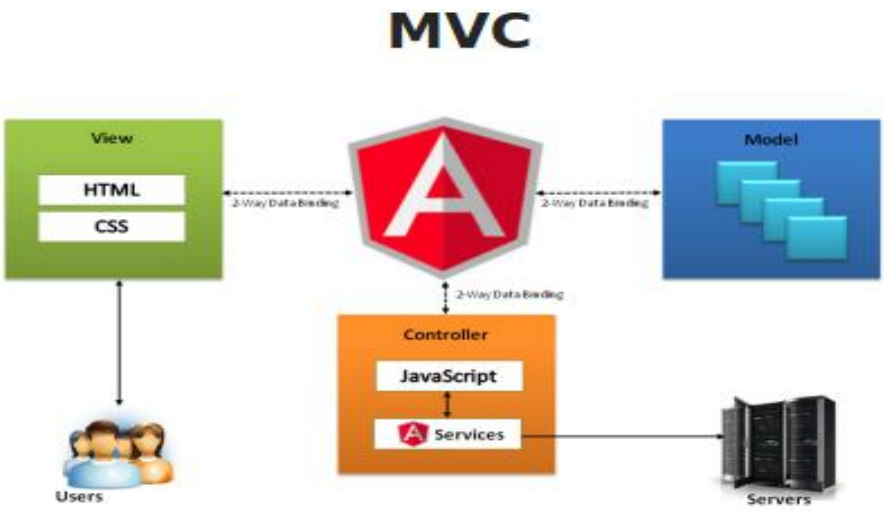

Fig. 4.5 Arhitectura Angular JS

[12] https://docs.angularjs.org/guide/service
[13] https://docs.angularjs.org/api/ng/service/$http

AngularJs a fost ales deoarece este o soluție completă și rapidă pentru dezoltarea front-end. Nu sunt necesare alte plugin-uri sau framework-uri pentru a construi o aplicație web bazată pe date. Angular folosește modelul MVC (Model-View-Controller) care duce la o organizare eficientă a codului și oferă posibilitatea de dezvoltare ulterioară, folosește Data binding și Dependency injection care duce la decuplarea componentelor, oferă posibilitatea de a extinde HTML-ul prin adăugarea de noi directive precum și de a crea template-uri pentru modele de date.

4.1.5 ASP.Net Identity

Identity este un API creat de Microsoft care ajută la administrarea userilor în aplicațiile .Net și poate fi folosit cu orice framework ASP.NET cum ar fi ASP.Net MVC, Web Forms, Web API sau SignalR. Identity poate fi folosit în construirea aplicațiilor web,mobile, de tip storage sau hibride. Autentificarea ASP.Net este bazată pe middleware-ul Open Web Interface for .Net (OWIN) și poate fi folosită pe orice host bazat pe OWIN.

Acest API a fost ales deoarece prezintă următoarele beneficii:

- Un singur sistem de identitate – poate fi folosit pentru toate famework-urile ASP.NET
- Controlul persistenței – în mod implicit, Identity stochează toate informațiile userilor într-o bază de date. Aceasta folosește Entity Framework Code-First pentru a implementa toate mecanismele de persistență. Se pot adăuga cu ușurință diferite mecanisme de stocare cum ar fi SharePoint, Azure Storage Table Service, baze de date de tip NoSql, etc. fără a fi necesare schimbări majore.
- Furnizor de roluri – restricționează accesul la anumite părți din aplicație în funcție de roluri.
- Bazat pe revendicări – Identity suportă autentificarea bazată pe revendicări, unde identitatea userului este reprezentată că un set de revendicări (claims), care ajută dezvoltatorii să fie mult mai expresivi în descrierea identității.
- Social login providers – pot fi adăugați cu ușurință furnizori pentru social log-în cum ar fi Conturi Microsoft, Facebook, Twitter, Google la aplicație, și permite stocarea datelor specifice userilor în aplicație
- Integrarea cu OWIN – autentificare bazată pe OWIN, poate fi folosit pe orice host bazat pe Open Web Interface for .Net.

4.1.6 Baza de date

O bază de date reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea regăsirii rapide a acestora. Adesea, o bază de date este memorată într-unul sau mai multe fișiere. Bazele de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date.

Cel mai răspândit model de date este modelul relațional, în care datele sunt memorate în tabele. Pe lângă tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, vederi, mecanisme de securitate și de gestiune a tranzacțiilor, etc.

4.1.6.1 Microsoft SQL Server

MS SQL Server este un sistem de management al bazelor de date relaționale. Arhitectură MS SQL Server este împărțită în 3 componente:

- SQLOS care implementează serviciile de bază necesare serverului SQL, incluzând managementul thread-urilor, al memoriei și a I/O
- Motorul relațional (relational engine), care implementează componentele relaționale ale bazei de date, incluzând suport pentru bazele de date, tabele, interogări și proceduri stocate
- Protocol layer care expune funcționalitatea serverului SQL.

Principala unitate de stocare este baza de date, care este o colecție de tabele cu coloane tipizate. SQL Server suportă diferite tipuri de date, incluzând tipuri primare (Integer, Float, Char, Decimal, Varchar, Text, Binary). Permite deasemenea tipuri de date definite de utilizator. O bază de date mai poate conține vederi, proceduri stocate, indecși, constrângeri.

4.1.7 Unity Container

Unity Container (Unity) este un dependency injector container extensibil. Facilitează construirea aplicatilor slab cuplate și furnizează dezvoltatorilor următoarele avantaje:

- Crearea simplificată a obiectelor, în special a structurilor ierarhice de obiecte și a dependințelor
- Abstracția cerințelor; aceasta permite dezvoltatorilor să specifice dependințele la run-time sau în configurație

Dependency injection injectează dependențele unei clase în timpul rulării. Acest lucru crează o cuplare slabă între clase, deoarece o schimbare într-una din clase nu are niciun impact pe celelalte clase dependente.

4.2 Cerințele sistemului

Cerințele unui sistem pot fi clasificate între cerințe funcționale și cerințe non-funcționale. Cerințele funcționale prezintă o descriere completă a funcționalităților pe care sistemul trebuie să le îndeplinească, ce să se poată realiza utilizând sistemul. Cerințele non-funcționale dictează proprietăți și constrângeri asupra sistemului și sunt specificate atribute de calitate pe care sistemul trebuie să le dețină.

4.2.1 Cerințe funcționale

În ingineria software, o cerința funcțională definește o funcționalitate a sistemului, sau a unei componente a sistemului. O funcționalitate poate fi descrisă că un set de variabile de intrare, un comportament specific funcționalității și un set de variabile de ieșire. Cerințele funcționale pot fi calcule, detalii tehnice, manipulări sau procesări de date, sau alte funcționalități specifice care definesc ce ar trebui să realizeze un sistem .

Pentru o gestiune mai usoară a cerințelor funcționale ale sistemului, acestea vor fi prezentate în subcapitolele următoare în funcție de tipurile de utilizatori care pot beneficia de acestea.

Identificator	Descrierea cerinței funcționale	Utilizator
CF 1	Autentificare	Toti utilizatorii
CF 2	Administrare conturi utilizator	Admin
CF 2.1	Administrare conturi candidati	Admin
CF 2.2	Administrare conturi HR	Admin
CF 2.3	Administrare conturi Admin	Admin
CF 3	Administrare sesiuni de testare	Admin
CF 3.1	Adăugare sesiuni noi	Admin
CF 3.2	Dezactivare/activare sesiuni	Admin
CF 3.3	Ștergere sesiuni de testare	Admin
CF 4	Administrare întrebari	Admin
CF 4.1	Ștergere întrebari	Admin
CF 4.2	Adăugare întrebari	Admin
CF 4.3	Modificare întrebari	Admin
CF 4.4	Asignare întrebari la secțiuni	Admin
CF 5	Administrare secțiuni	Admin
CF 5.1	Adaugare secțiune	Admin
CF 5.2	Ștergere secțiune	Admin
CF 5.3	Asignare secțiune la template	Admin
CF 5.4	Asignare întrebari la secțiune	Admin
CF 6	Administrare template-uri	Admin
CF 6.1	Adăugare template	Admin
CF 6.2	Ștergere template	Admin
CF 6.3	Asignare secțiuni la template	Admin
CF 7	Gestionare test-uri	Admin, Hr
CF 7.1	Vizualizare test	Admin, Hr
CF 7.2	Acoradre de review la test	Admin, Hr
CF 7.3	Ștergere test	Admin, Hr
CF 8	Vizualizare rapoarte	Admin, Hr
CF 8.1	Sortare teste dupa anumite criterii	
CF 8.2	Vizualizare detaliată a notelor	Admin, Hr
CF 8.3	Vizualizare de statistici	Admin, Hr
CF 9	Contactare candidat	Admin, Hr
CF 10	Susținere test	Candidat

Tabel 4.1 Cerințele funcționale ale sistemului

Precum sunt definite în ingineria cerințelor, cerințele funcționale specifică rezultatele particulare ale unui sistem. Acestea sunt în contrast cu cerințele non-funcționale care specifică caracteristicile generale ale sistemului cum ar fi costul sau fiabilitatea.

4.2.2 Cerințe non-funcționale

În ingineria sistemelor și ingineria cerințelor, o cerința non-funcțională este o cerința care specifică criterii care pot fi folosite pentru a judeca funcționarea unui sistem, mai degrabă decât un comportament specific. Planul de punere în aplicare a cerințelor de bază non-funcționale este detaliat în arhitectura sistemului, deoarece acestea sunt, de obicei, cerințe semnificative arhitectural[5]. Cerințele non-funcționale sunt de asemenea numite și atribute de calitate a unui sistem.

Utilizabilitatea unui sistem este dată de ușurința cu care acesta poate fi învățat sau utilizat, astfel sistemul propus își dorește ca timpul de acomodare și de învățare necesar să fie cât mai redus. O astfel de cerință non-funcțioanlă poate face diferența între un sistem de real succes, sau un sistem care eșuează. Utilizabilitatea sistemului este dată și de designul interfeței grafice, dar și intuitivitatea acesteia, plasarea elementelor în locații specifice, locații cu care utilizatorul este obișnuit de la sistemele pe care le utilizează frecvent. O aplicație cu o interfață grafică cât mai prietenoasă și intuitivă, chiar incompletă din punctul de vedere al funcționalităților poate fi preferată de utilizatori în detrimentul unei aplicații complete care acoperă multe din cerințe funcționale de interes utilizatorului, dar care nu beneficiază de o interfață grafică adecvată. Utilizabilitatea este susținută și de eficiența pe care o au utilizatorii în îndeplinirea diferitelor taskuri, eficiență care se garantează prin accesul rapid la resursele disponibile.

Disponibilitatea sistemului reprezintă măsura în care sistemul trebuie să funcționeze pentru utilizatori. Ținând cont de faptul că sistmul dorește să devină o alternativă procesului manual de management al documentelor, activităților și evaluării dintr-o școală, disponibilitatea este una din cele mai importante cerințe pe care sistemul trebuie să le acopere. Disponibilitatea maximă care se impune este ca sistemul să fie disponibil utilizatorilor 24 de ore din 24, iar timpul de revenire într-o stare stabilă a sistemului în cazul unui eșec să nu depășască 30 de minute, maxim o oră. Disponibilitatea unui sistem este susținută și de cât este de sigur un sistem, cât este de robust și de tolerabil la erorile care pot să intervină. Validarea datelor reprezintă cheia către un sistem robust și tolerant, datele introduse de utilizatori pot să fie adesea incorecte sau neformatate corect, lucru care sistemul trebuie sa-l detecteze.

Performanța unui sistem se rezumă în cele mai multe cazuri la timpul maxim de răspuns care este pus la dispoziție sistemului pentru a oferi feedbackul necesar în urma unei acțiuni declanșate de utilizator. Depinzând de context, preformanta poate implica una sau mai multe din următoarele aspecte:

- Timp de răspuns mic pentru un request

- Rata de procesare a informației
- Utilizarea redusă a resurselor calculatorului
- Compresarea/decompresarea sau codificare/decodificarea rapidă a datelor
- Timp scurt de transmitere a datelor

În cazul sistemului prezentat, performanța este data atât de timpul scurt necesare procesării unui request, cât și de dimensiunea redusă a datelor folosite în comunicarea client-server (formatele x-www-form-urlencoded și json), care duce la transferuri rapide de date și utilizare scăzută a capacității canalului de comunicație.

Securitatea în mod fundamental se referă la protejarea bunurilor. Bunurile pot fi obiecte tangibile precum o pagină web sau o bază de date sau pot fi obiecte mai puțin tangibile precum reputația unei companii. Securitatea în cazul sistemului descris este asigurată prin introducerea autorizării server-side pe bază de token-uri de tip Bearer, generate de o componenta middleware situată între client și server. User-ul trebuie să își furnizeze userul și parola o singură data, după care primește un token trebuie transmis împreună cu fiecare request făcut către server. Tot odată, securitatea mai este asigurată prin hashuirea parolelor de către ASP.NET Identity cu algoritmul de hashing Password-Based Key Derivation Function 2 (PBKDF2), care aplică o funcție pseudoaleatoare cum ar fi hash criptografic sau HMAC pe parolă, împreună cu o valoare salt, și repetă procesul de mai multe ori producând o cheie derivată[14]. Pe partea client-side, securitatea este asigurată prin validarea rutelor, verificând dacă userul curent are dreptul de a accesa ruta respectivă, evitând astfel accesarea sau modificarea unor resurse la care userul nu are acces.

Extensibilitatea în ingineria software este un principiu de design al sistemului, unde implementarea este făcută luând în considerare posibilitatea de creștere ulterioară a sistemului. Este o măsură sistemica a capacității de a extinde un sistem și nivelul de efort necesar pentru a pune în aplicare sau a implementa extinderea. Extensibilitatea mai poate fi definită ca și abilitatea unui sistem da a avea noi funcționalități sau de a le extinde pe cele existente, unde structura internă a sistemului și fluxul de date sunt afectate minim sau deloc, în special recompilarea sau schimbarea codului sursă original nu este necesară la adăugarea unor componente sau la schimbarea comportamentului sistemului[15].

Extensibilitatea în sistemul prezentat este realizată prin frameworkului Web API care centralizează logica de business a aplicației pe web server, expunând un API care poate fi consumat de varietate largă de clienți precum browsere, aplicații desktop, aplicații mobile (Android, IOS, Windows Phone), astfel un nou client putând fi adăugat cu ușurință, și fără modificarea codului sursă. Totodată, extensibilitatea mai este asigurată prin folosirea frameworkului AngularJs care folosește modelul MVC (Model-View-Controller), unde interfață

[14] PBKDF2 - https://en.wikipedia.org/wiki/PBKDF2
[15] Johansson, Niklas, and Anton Löfgren. Designing for Extensibility, University of Gothenburg, 29 May 2009 https://gupea.ub.gu.se/bitstream/2077/20561/1/gupea_2077_20561_1.pdf [2016-06-19]

și logica sunt complet separate, putând fi schimbate independent una de cealaltă. Posibilitatea de a scrie cod independent de o anumită bază de date, beneficiu oferit de frameworkul EntityFramework, asigură independența de un anumit sistem de gestiune a bazelor de date, și schimbarea acestora fără a fi nevoie de modificarea codului sursă, constituind o altă caracteristică de extensibilitate oferită de sistemul prezentat.

4.3 Cazuri de utilizare

În ingineria software și a sistemelor, un caz de utilizare este o listă de acțiuni sau etape de evenimente, care definesc în mod tipic interacțiunile dintre un rol, cunoscut în UML (Unified Modeling Language) ca actor, și un sistem, pentru a atinge un obiectiv. Actorul poate fi un om sau un sistem extern. Cazurile de utilizare au menirea să ofere o perspectivă globală asupra comportamentului și funcționalităților puse la dispoziție de către un sistem pentru utilizatori.

4.3.1 Actorii sistemului

Tipurile de utilizatori care se găsesc in sistem sunt:

- Administrator – responsabil cu managementul întrebărilor, secțiunilor, templateurilor, sesiunilor de testare și a userilor
- Hr – responsabil cu vizualizarea de rapoarte și cu contactarea candidaților
- Candidat – candidatul care susține testul

4.3.2 Cazuri de utilizare

În cele ce urmează vor fi prezentate cazurile de utilizare pentru fiecare tip de utilizator în parte și anume administrator, hr și candidat.

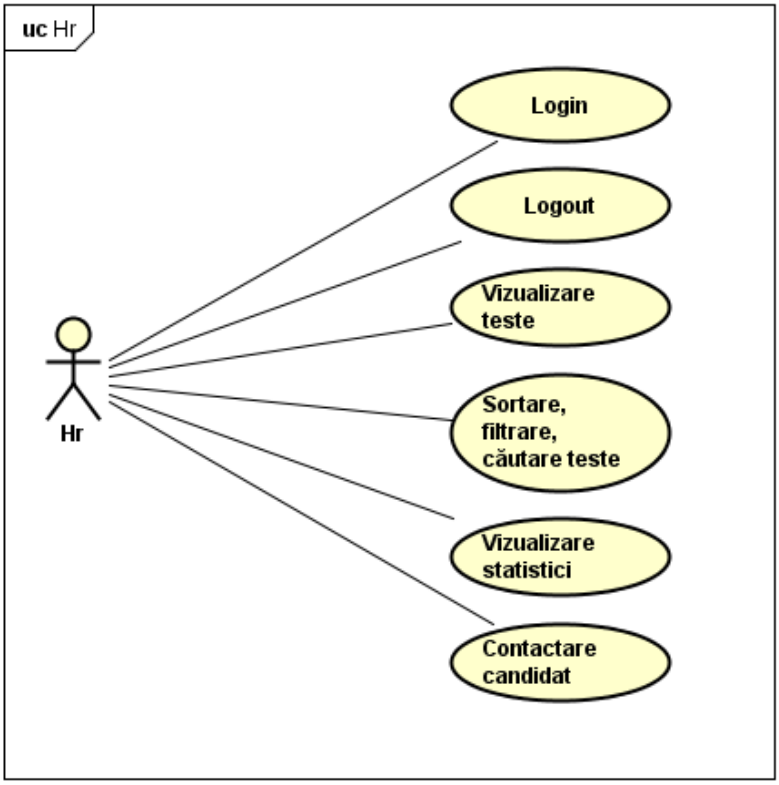

Fig. 4.6 Cazuri de utilizare pentru Hr

Figura anterioară, Fig. 4.6, prezintă cazurile de utilizare actorului care are rolul de Hr. În figura următoare, Fig. 4.7 , sunt prezentate cazurile pentru actorul cu rolul de Candidat.

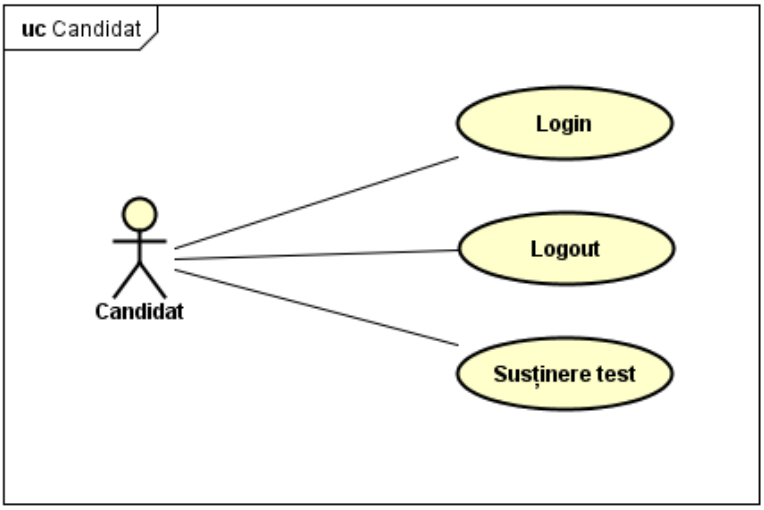

Fig. 4.7 Cazuri de utilizare pentru Candidat

În figura de mai jos, Fig. 4.8, vor fi prezentate cazurile de utilizare pentru actorul cu rolul de Administrator, rol care reprezintă actorul principal al acestui sistem.

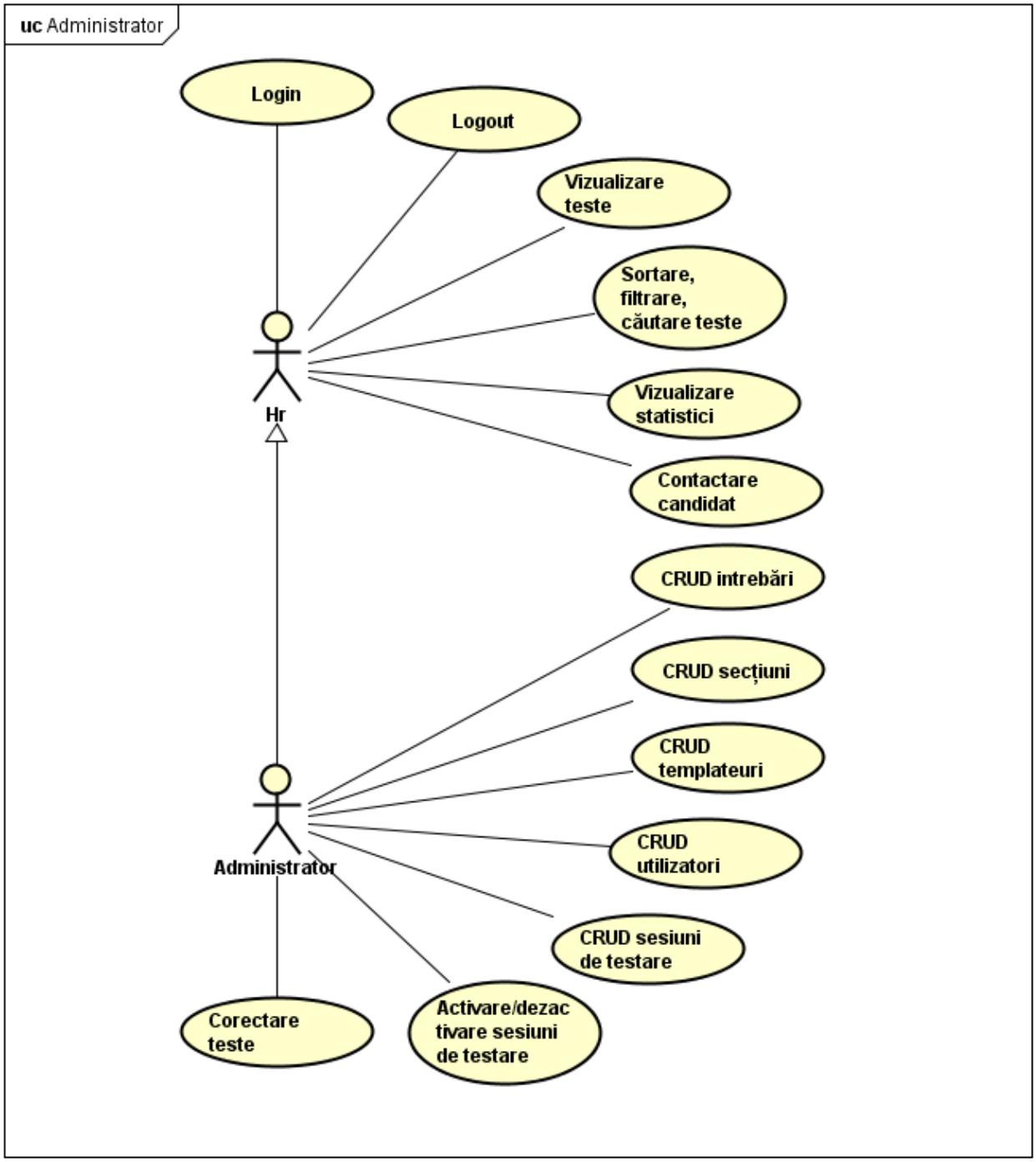

Fig. 4.8 Cazuri de utilizare pentru Administrator

4.3.2.1 Descriere detaliata a cazurilor de utilizare

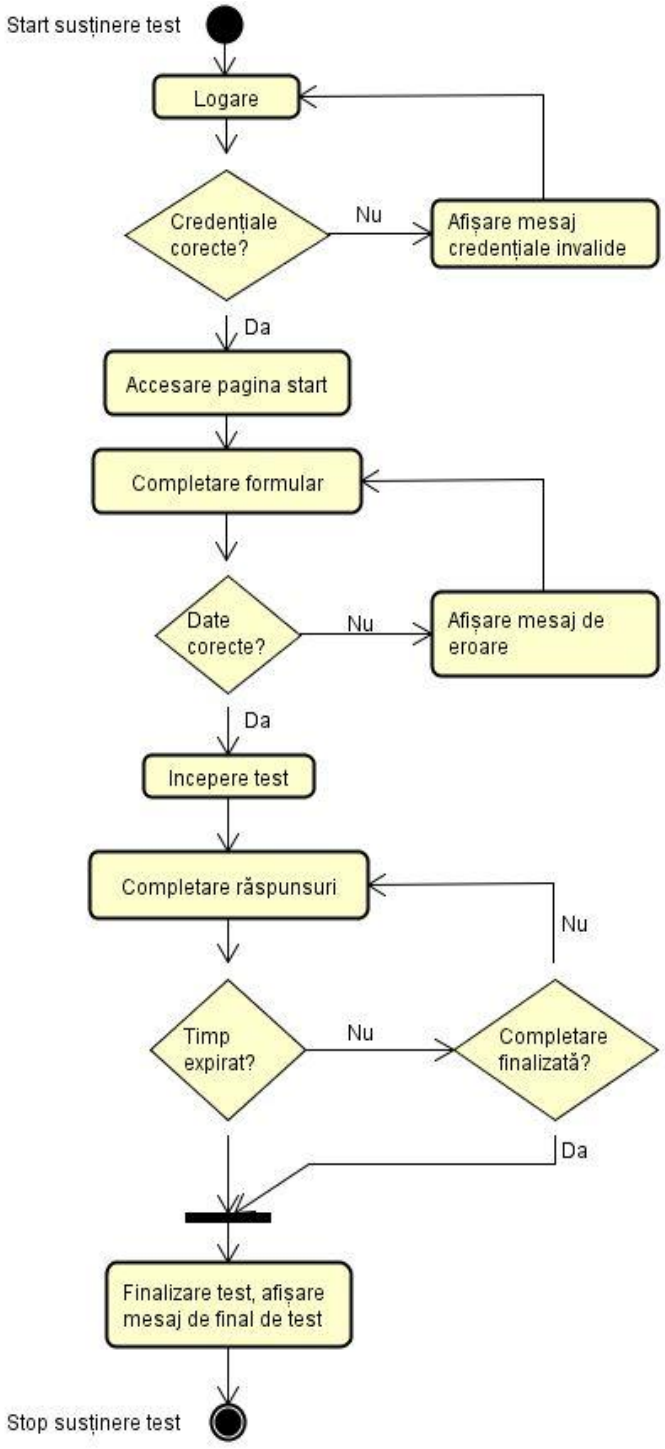

Fig. 4.9 Diagramă flow-chart pentru susținere test

Diagrama anterioară prezintă în detaliu fluxul de evenimente declanșat la activitatea cazului de utilizare în momentul momentul susținerii testului de către un candidat.

CU1

Numele cazului de utilizare: Susținere test

Actor principal: Candidat

Parți interesate (Stakeholders):

- Candidatul: dorește să susțină un test în vederea calificării pentru o anumită poziție în cadrul companiei
- Administratorul: dorește să aibă acces la toate testele care au fost susținute pentru a ține o evidența ale acestora, pentru a le putea corecta și a oferi o un calificativ, și a face anumite remarci asupra testelor
- Hr: dorește să aibă acces la evidența testelor corectate, pentru a putea contacta candidații și a planifica următoarele întâlniri, sau pentru a le comunica faptul că rezultatul testului a dus la respingerea acestora

Precondiții: Nu exista precondiții pentru acest caz de utilizare

Postcondiții: Testul este salvat cu succes în baza de date, lucru care oferă accesul stakeholderilor la el. Din acest moment un test poate fi corectat și vizualizat.

Scenariul de succes:

1. Candidatul se logheaza in aplicatie
2. Candidatul completează datele corecte in formularul de identificare a candidaților si selecteaza sesiunea de testare pentru care aplică
3. Candidatul acceptă termenii și conditile testului, după care apasă butonul de incepere a testului
4. Sistemul salvează datele candidatului, încărca template-ul folosit pentru sesiunea de testare a candidatului și setează constrângerea de timp aferentă testului
5. Candidatul navighează pe diferitele secțiuni ale testului, completând răspunsurile în căsuțele aferente întrebărilor. Acesta nu închide browser-ul sau nu încearcă să acceseze alte pagini prin intermediul browserului
6. Dupa ce candidatul termină de completat răspunsurile, timpul de testare încă nu s-a epuizat
7. Candidatul apăsa pe butonul „Submit” pentru finalizarea testului, primind un mesaj de informare că testul a fost finalizat cu succes și că urmează să fie contactat în cel mai scurt timp cu rezultatul.
8. Sistemul salvează răspunsurile testului în baza de date, după care redirecționează userul la pagina principală a aplicației

Scenarii alternative:

- În timpul susținerii testului, candidatul încearcă părăsirea paginii curente prin apăsarea butonului „Inapoi” al browserului sau prin modificare URL-ului din bara de adrese:
    1. Acesta este avertizat că nu are voie să părăsească testul, iar dacă o va face, testul va fi închis
    2. Dacă candidatul acceptă părăsirea paginii, testul este închis iar răspunsurile sunt salvate în baza de date
    3. Dacă candidatul nu acceptă părăsirea paginii, acest lucru fiind făcut accidental, testul este continuat în condiții normale
- În timpul susținerii testului, timpul alocat pentru testare expiră, în timp ce candidatul completează răspunsurile fără a mai apuca să apese butonul „Submit”:
    1. Sistemul informează candidatul de faptul că testul a luat sfârșit iar răspunsurile sunt salvate în baza de date, testul fiind finalizat cu succes
- Sistemul nu poate salva răspunsurile în baza de date, din cauza indisponibilității acesteia:
    1. Sistemul afișează un mesaj de eroare

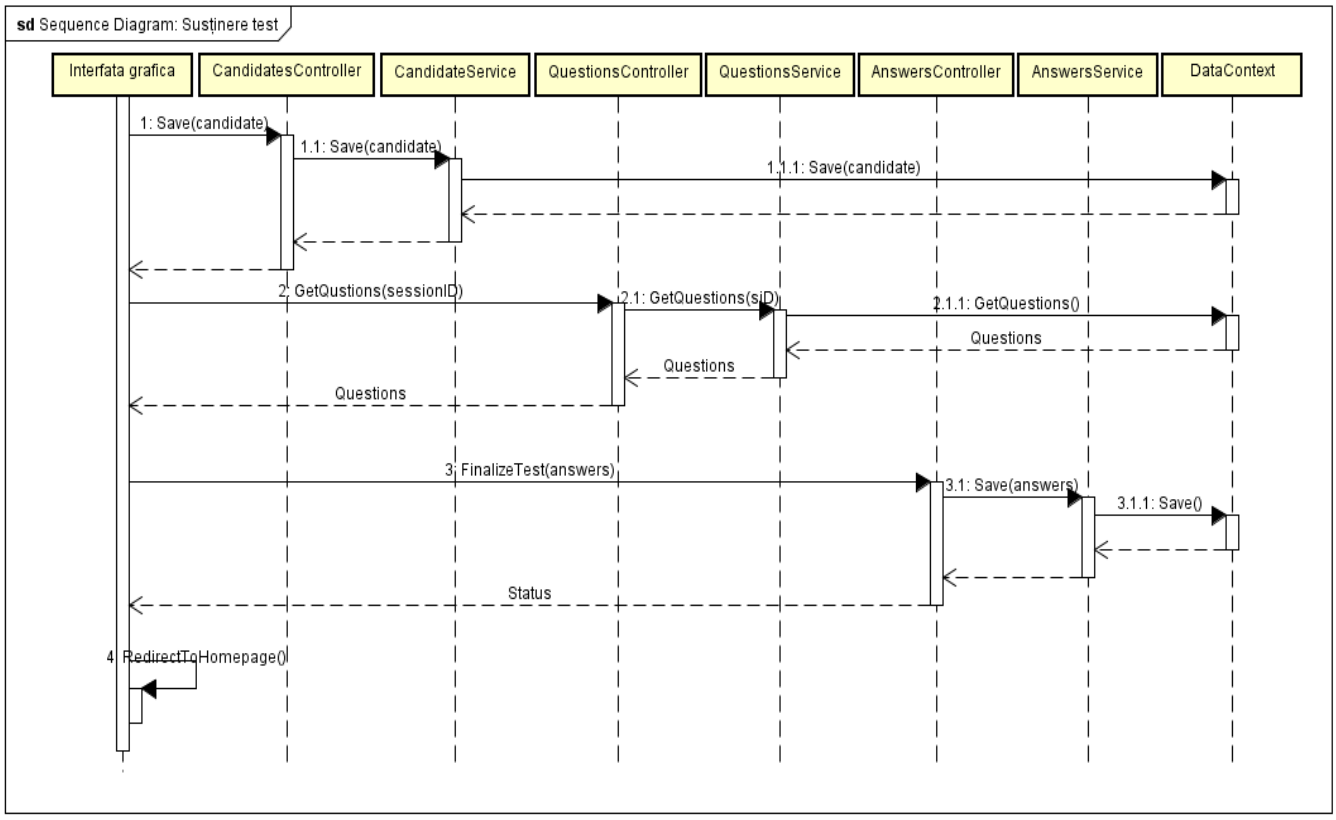

Fig. 4.10 Diagramă de secvență pentru susținere test

Diagrama de secvența prezentată anterior evidențiază comunicarea dintre componentele sistemului pentru realizarea cazului de utilizare ce are ca actor principal un candidat care dorește să susțină un test. La accesarea paginii pentru evaluare, candidatului îi este prezentat un formular unde trebuie să își introducă datele personale cum ar fi numele, e-mail sau telefon și sesiunea de testare la care participă. După apăsarea butonului start, acesta va fi redirecționat la pagina de susținere a testului, unde îi va fi prezentat testul structurat pe secțiuni, unde fiecare tab reprezintă o secțiune.

Următorul caz de utilizare prezentat ilustrează scenariul pentru adăugarea și configurarea unui template. Scenariul complet presupune pe lângă crearea templateului, și crearea de secțiuni și întrebări pentru acesta.

CU2
Numele cazului de utilizare: Adăugarea unui template
Actor principal: Administratorul
Parți interesate (Stakeholders):

- Administratorul: dorește să adauge și să configureze un nou template care va fi ulterior folosit pentru o anumită sesiune de testare

Precondiții: Administratorul este autentificat in sistem
Postcondiții: Template-ul și toate resursele suplimentare care au fost adăugate pentru acesta sunt salvate cu succes în baza de date.

Scenariul de succes:

1. Administratorul accesează pagina responsabilă pentru managementul întrebărilor
2. Administratorul adaugă întrebarile care doreste să apară in template
3. Sistemul salvează întrebarile în baza de date
4. Administratorul accesează pagina responsabilă pentru managementul secțiunilor
5. Administratorul adaugă secțiunile aferente templateului, și configurează fiecare secțiune cu întrebările dorite
6. Sistemul salvează secțiunile în baza de date
7. Administratorul accesează pagina responsabilă pentru managementul templateurilor
8. Administratorul adaugă templateul

9. Administratorul configurează timpul dorit de testare, sesiunea de testare căruia i se adresează, și adagua secțiunile care dorește să apăra în template, selectând pentru fiecare ponderea, care apoi va fi folosită la calcularea notei finale
10. Se selectează template-ul creat ca template implicit pentru sesiunea de testare căruia i se adresează
11. Sistemul salvează toate datele legate de template în baza de date

Scenarii alternative:

- În orice pas al scenarilui de succes, sistemul se blochează sau baza de date devine indisponibilă:
    1. Sistemul afișează un mesaj de eroare prin care avertizează administratorul
    2. Administratorul se reautentifica în sistem, iar cazul de utilizare se repetă
- În momentul în care administratorul setează template-ul creat ca implicit, deja există alt template implicit
    1. Sistemul afișează un mesaj de informare prin care avertizează administratorul de faptul că deja există un template default pentru acea sesiune de testare indicând și numele templateului
    2. Administratorul accesează template-ul respectiv, și debifeaza setarea „Default template”, după care salvează modificările făcute
    3. Administratorul revine la template-ul creat, și îl setează ca implicit după care salvează modificările
    4. Sistemul salvează modificările făcute cu succes
- Administratorul trebuie să întrerupă activitatea deoarece apare o urgență în companie:
    1. Administratorul anulează pasul curent la care se află, având posibilitatea de al relua după întoarcere, scenariul nu se reia de la început.

Următorul caz de utilizare prezintă scenariul de corectare a unui test prezentând pașii care trebuie urmați pentru realizarea cu succes a scenariului. Actorul principal în acest caz de utilizare va fi tot administratorul.

CU3
Numele cazului de utilizare: Corectare test
Actor principal: Administratorul
Parți interesate (Stakeholders):

- Administratorul: dorește să corecteze și să acorde un calificativ testului, calificativ pe baza căruia se va lua ulterior o decizie

- Hr: dorește să poată vizualiza o listă de teste corectate, pe baza căruia să poată contacta ulterior candidații care au reușit să promoveze testul pentru stabilirea unor întâlniri ulterioare

Scenariu de succes:

1. Administratorul accesează pagina responsabilă pentru corectarea testelor
2. Administratorul selectează sesiunea de testare în care testul a fost susținut
3. Sistemul furnizează o listă cu toate testele susținute în sesiunea de testare respectivă
4. Administratorul identifică în lista de teste afișată testul care se dorește a fi corectat și apăsa pe butonul „Add review”
5. Pentru fiecare întrebare în parte, administratorul oferă o notă
6. Administratorul navighează la tabul final remaks, unde introduce anumite observații pe care le are de făcut asupra testului, selectează un calificativ după care apasă butonul „Submit”
7. Sistemul adaugă în baza de date notele și calificativul acordat testului

Scenarii alternative:

- În orice pas al scenariului de succes, aplicația sau browser-ul se blochează, sau se iese accidental din browser
    1. Administratorul se reautentifica în sistem, reluând din nou scenariul.
- Administratorul trebuie să întrerupă activitatea de corectare a testului
    1. Administratorul salvează modificările care le are făcute până acum
    2. După revenire, administratorul accesează testul din nou, reluând activitatea.

Capitolul 5. Proiectare de Detaliu și Implementare

În acest capitol va fi prezentat modul în care sistemul a fost proiectat. Se va prezenta schema arhitecturală a sistemului, arhitectura bazei de date și structura tabelelor, diagrama de deployment și vor fi detaliate anumite componente ale aplicației considerate relevante în înțelegerea a felului în care sistemul este proiectat.

5.1 Arhitectura sistemului

La baza arhitecturii sistemului stă modelul arhitectural pe trei nivele (Three-tier). Arhitectura three-tier este un model de arhitectură software client-server în care interfața cu utilizatorul (prezentare), logica de process funcțional (reguli de business) și logica de stocare a datelor împreună cu accesul la date sunt dezvoltate și mentiunte ca module independente, de cele mai multe ori pe platforme separate[4]. Modelul three-tier al aplicației conține următoarele nivele: nivelul de prezentare (presentation tier), nivelul logic sau nivelul aplicație (business tier) și nivelul de stocare a datelor (data tier). Nivelul de prezentare, reprezentat de aplicația client, interacționează nivelul logic prin protocolul HTTP. Nivelul logic înglobează cea mai mare parte din logica aplicației, transformând cererile venite de la client în interogări sau modificări ale bazei de date și convertind datele extrase din baza de date în modele de date care să poate fi interpretate de către client. Acesta la rândul sau interacționează cu nivelul de stocare al datelor, în vederea efectuări unor operațiuni asupra bazelor de date.

Nivelul de prezentare reprezintă nivelul superior al aplicației, și este nivelul care se ocupă cu afișarea informațiilor simplu și intuitiv pentru useri și cu tratarea interacțiunilor pe care un user le are cu sistemul. Nivelul de prezentare este reprezentat de aplicația client, o aplicație pe o singură pagină (SPA), implementată având la baza framework-ul AngularJS, structurat pe modelul Model-View-Controller. Controlerele interacționează cu nivelul logic prin intermediul mesajelor de tip request/response, în urma cărora au loc modificări asupra modelului, care vor fi reflectate în UI. Acest modul interacționează direct cu utilizatorii, oferindu-le posibiltatea de a adăuga date noi în bază de date, de a modifica datele stocate și de a vizualiza informațiile existențe făcând anumite căutări sau filtrări.

Nivelul logic numit și nivelul aplicație, înglobează toată logica de business corespunzătoare sistemului. Acest nivel este responsabil cu procesarea cererilor venite de la aplicația client, validarea datelor și efectuarea deciziilor care țin de logica sistemului. Acest nivel mai poate fi numit și nivelul de mijloc, deoarece acesta face legătura între nivelul de prezentare și nivelul de acces la date. Integrând logica de business la acest nivel, face înlocuirea clientului existent sau adăugarea de clienți noi, precum și schimbarea sistemului de stocare a datelor să necesite un nivel minim de efort, oferă o caracteristică de extensibilitate sistemului.

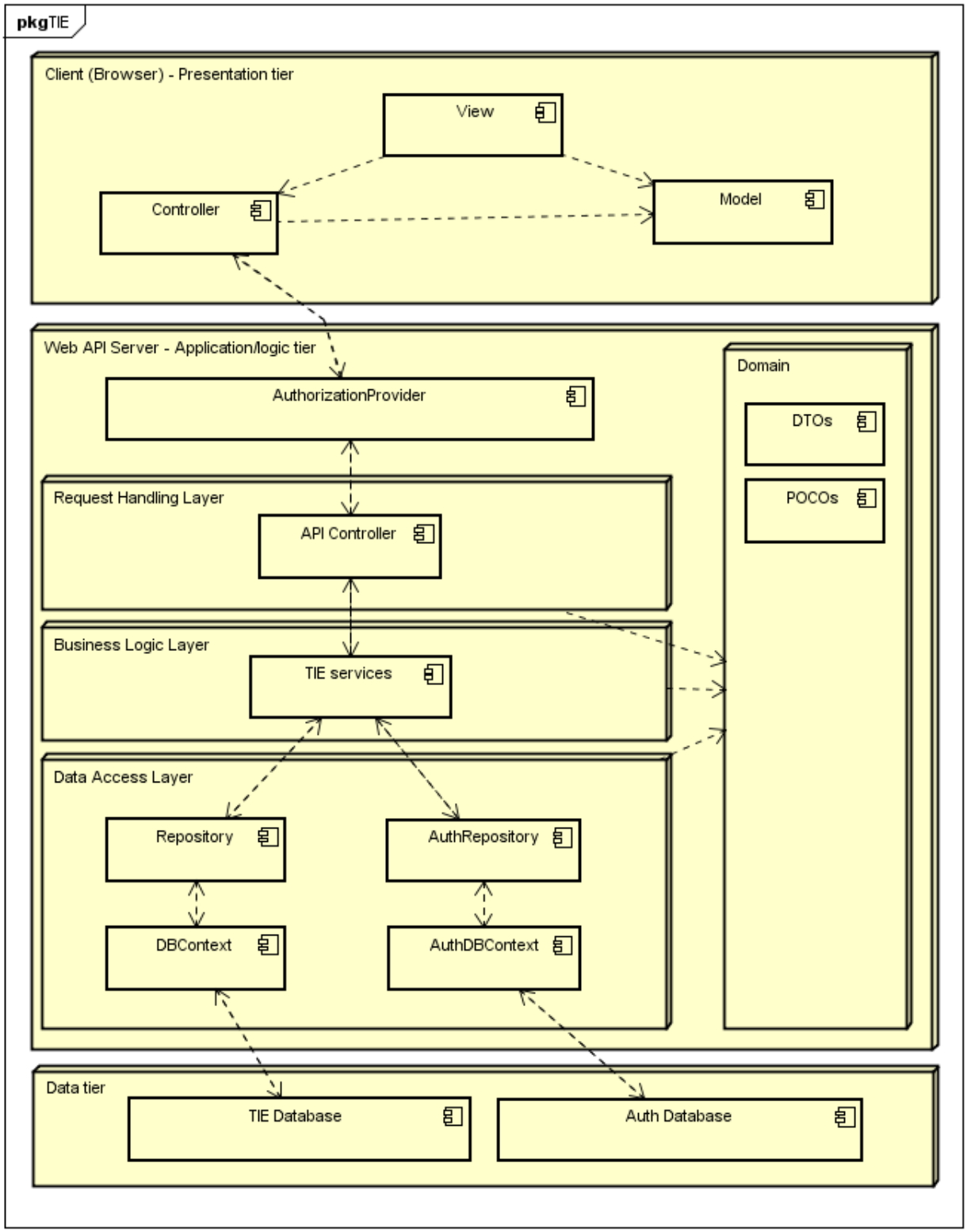

Fig. 5.1 Diagrama arhitecturală a sistemului

Nivelul de stocare a datelor este reprezentată de serverul de management al bazei de date. La acest nivel este stocată toată informația aferentă aplicației. Baza de date este organizată că și o colecție de date, o colecție de scheme, tabele, interogări, vederi și alte obiecte. La acest nivel se execută operații de inserare, actualizare, ștergere sau extragere a datelor. Nivelul de stocare comunică cu nivelul logic (nivelul aplicație), căruia îi furnizează datele disponibile în bază de date sau la cererea căruia efectuează inserări, modificări sau ștergeri.

Aplicația a fost dezvoltată folosind IDE-ul Microsoft Visual Studio 2013 care pune la dispoziție un set de unelte care ajută dezvoltatorul în scrierea programelor. Pot fi identificate patru mari componente în cadrul soluției sistemului:

- Tie.SPA – este componenta client a sistemului
- Tie.API – reprezintă serverul Web API al sistemului, server hostat în Internet Information Service (IIS)
- Tie.Business – componenta responsabilă cu logica de business a sistemului, conține și serviciile de acces la bază de date
- Tie.Core – componenta care specifică domeniul aplicației. Această componenta conține atât modelul și interfețele utilizate în aplicație cât și un container responsabil de inversarea controlului în aplicație

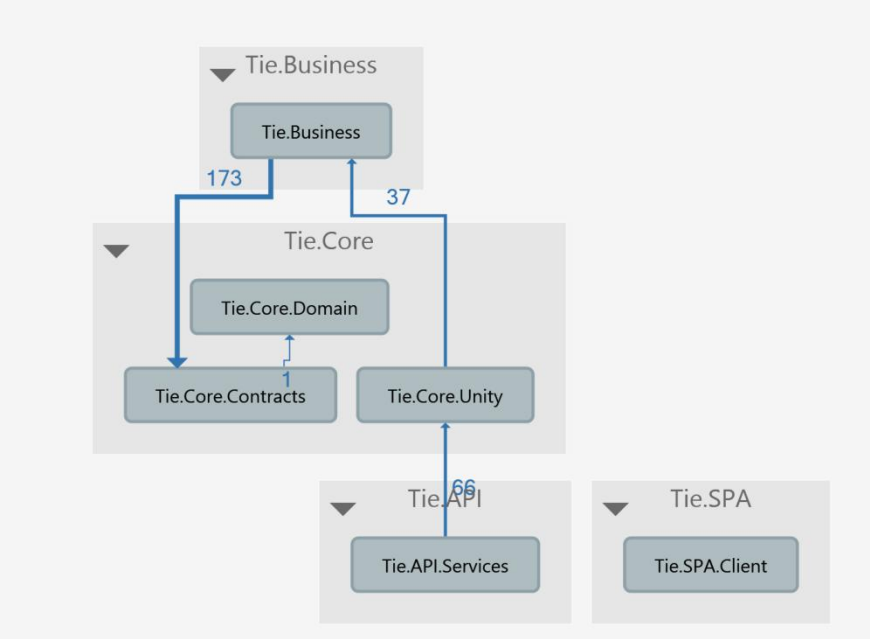

Fig. 5.2 Diagrama de dependențe a sistemului

În figura prezentată mai sus, Fig.5.2, se poate observa o delimitare clară a nivelului de prezentare, care conține componenta Tie.SPA, și a nivelului logic, care înglobează componentele Tie.API, Tie.Business și Tie.Core.

Tot în figura de mai sus, iese în evidența folosirea principiului de design „Dependency Injection”. Componenta Tie.Core.Unity este responsabilă cu instanțierea obiectelor în aplicație, făcând posibil faptul ca o clasă să poate fi modificată sau înlocuită cu ușurință, fără a fi necesară modificarea tuturor claselor care depind de această. În cele ce urmează, va fi ilustrat în exemplu de folosire a principiului de injecție a dependințelor în aplicație:

Mai întâi, o dependință trebuie înregistrată în containerul de injecție a dependințelor, specificând interfața obiectului, clasa care o implementează și dependințele acesteia dacă este cazul, și un nume asociat dependinței:

```
> _container.RegisterType(typeof(IManagementService<>),
                          typeof(QuestionManagementService),
                          "QuestionService",
                          new InjectionConstructor(typeof(QuestionConverterService)));
```

Clasele care vor avea dependință specificată mai sus, nu vor instanța direct clasa, ci se vor folosi de containerul de injecție a dependințelor pentru a obține o instanță a clasei, specificând numele dependinței dorite:

```
> public IManagementService<QuestionDTO> _questionService;
> _questionService = (IManagementService<QuestionDTO>)UnitySetup.Container
      .Resolve(typeof(IManagementService<QuestionDTO>),"QuestionService");
```

În cazul în care se dorește ca dependință „QuestionService” să nu mai returneze o instanță de tip QuestionManagementService, tot ce trebuie făcut este specificarea unei alte clase dorite pentru dependință, iar schimbarea se va reflectă în toate clasele care folosesc această dependință.

5.1.1 Presentation Tier

Aplicația client respectă modelul arhitectural Model-View-Controller oferit de frameworkul AngularJS, unde View-ul este reprezentat de template-urile HTML, Controller-ul este implementat prin controlerele Angular care sunt funcții JavaScript, iar modelul este reprezentat de serviciul $scope, care este injectat în controlere, care notifica view-ul să se reactualizeze atunci când au loc schimbări în model, sau este actualizat, atunci când se produc schimbări în view.

Client-ul este împărțit în mai multe sub-module. Această decizie a fost luată pentru o organizare modulară a codului, modulele fiind independente unele de altele. Aplicația conține următoarele sub-module: Login, Admin, Evaluation, Questions, Sections, Templates, Reports și Reviews. Pe lângă aceste module, mai există un modul comun, folosit de toate modulele enumerate precedent, care conține configurații, directive, filtre și servicii comune modulelor. De asemenea, fiind o aplicație SPA, aceasta mai dispune de o componenta de rutare, a cărui rol este

de a citi URL-ul furnizat de user, și în funcție de valoarea lui, va încărca un anumit modul, sau un anumit template și un controller. Pentru fiecare rută se specifică în parte un template HTML și un controller care va fi încărcat la accesarea rutei, controller care va îngloba funcționalitatea pentru pagina respectivă. Dacă ruta pe care un user încearcă să o acceseze nu este înregistrată în această componenta, acesta va fi redirecționat la o rută implicită, care în cazul aplicației este pagina „Home”. În figura de mai jos, este prezentată arhitectura aplicației client văzută la nivelul întregii aplicații:

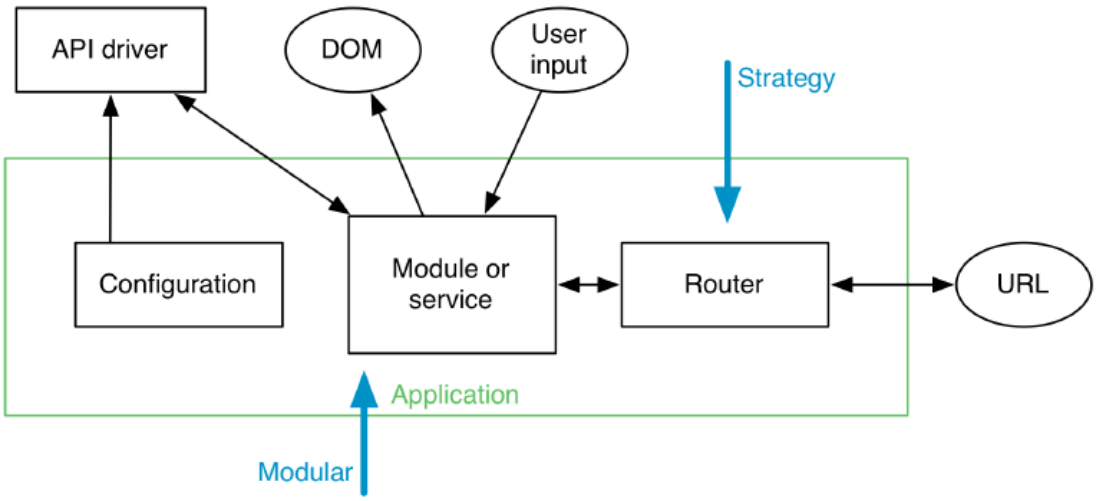

Fig. 5.3Arhitectura aplicației client văzuta de la nivelul aplicației

Un modul Angular este un container pentru diferite părți ale aplicației. Fiecare modul vine împreună cu propriile sale controlere, servici, template-uri HTML sau alte componente, și poate avea propriile sale dependințe. Fiecare din modulele aplicației depinde de modulul „Common”, în care se găsesc configurațiile aplicației, anumite filtre și directive comune modulelor, precum și servicii comune cum ar fi serviciul care este responsabil de setarea token-ului în header-ul unui request.

Fiecare user al aplicației, în funcție de rolul pe care îl are, are acces la anumite module:

- Administrator – are acces la modulele toate modulele aplicației, mai puțin la modulul Evaluation, care este responsabil cu procesul de testare
- Hr – are acces la modulul Reports
- Candidate – are acces la modulul Evaluation

Modulul de login este tot timpul disponibil, acesta fiind responsabil cu logarea/delogarea userilor. De asemenea acest modul va încărca dinamic meniurile disponibile, în funcție de rolul pe care fiecare user îl îndeplinește.

În figura de mai jos, Fig. 5.4, este prezentată arhitectura aplicației văzută din perspectiva unui modul:

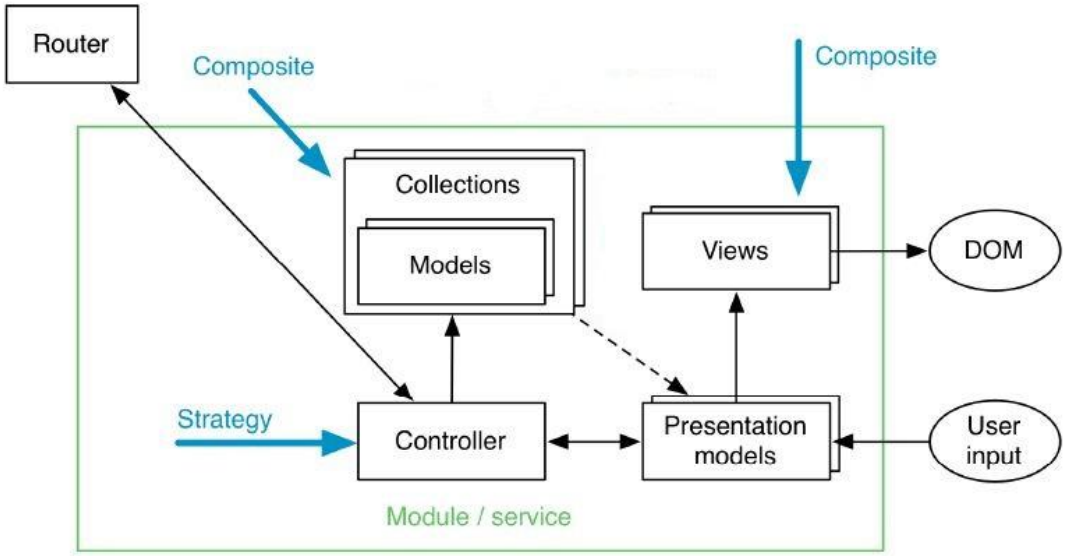

Fig. 5.4 Arhitectura aplicației client din perspectiva unui modul

5.1.2 Business Tier

Acest tier corespunde unui server web implementat cu ajutorul framework-ului ASP.Net Web API, care expune un set de servicii REST aplicațiilor client. Serviciile sunt structurate în controllere, unde fiecare controller are mai multe acțiuni posibile. Controllerele sunt obiectele care tratează requesturile HTTP, mapând un request la acțiune disponibilă din controller (Get, Post, Put, Delete).

Acest nivel este nivelul responsabil cu tratarea cererilor recepționate de la aplicația client. Business tier este structurat pe trei layere independente: Request Handling Layer, Business Logic Layer și Data Access Layer. Structurarea pe layere ajută la distingerea și la distribuirea responsabilitațiilor pe care acest nivel al aplicației le are. Fiecare layer fiind independent, poate fi extins sau înlocuit fără a fi nevoie că celelalte layere să fie alterate în urmă modificărilor aduse.
În figura de mai jos, este prezentată schema arhitecturală a nivelului de business:

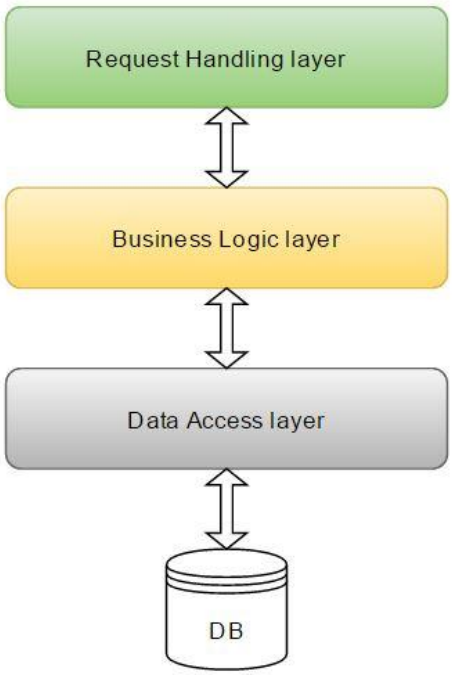

Fig. 5.5 Arhitectura Business Tier

5.1.2.1 Request Handling Layer

Acest layer este responsabil cu interceptarea request-urilor și cu construirea mesajelor de răspuns. Pentru a facilita ușoara implementare a unor dezvoltări viitoare, s-a optat pentru construirea câte unui controller pentru fiecare tip de date disponibil în aplicație. Pentru a accesa o acțiune disponibilă într-un controller, aplicația client va face un request la adresă hostname:port/api/{controller}/{id} unde {controller} reprezintă numele controllerului (ex. Questions), iar parametrul id este opțional, el fiind folosit doar când se dorește o acțiune pe o resursa individuală.

Înaintea definitei fiecărui controller, este prezent atributul [Authorize] care specifică faptul că resursele vor fi accesibile doar dacă userul este autorizat în aplicație. Autorizarea în aplicație după modelul de autentificare OAuth Owner Resource Flow[3].

Principalii actori ai fluxului de autentificare al acestui sistem sunt:

- **Resource Owner** – detinatorul resursei, în cazul sistemului prezentat fiind userul
- **Client** – aplicația client a acestui sistem
- **Authorization Server** – serverul care acceptă credentialele userului, și returnează un token de acces criptat. În cazul sistemului prezentat, serverul de autorizare este încorporat

în serverul web, iar logica de validare a credentialelor este realizată în componenta AuthorizationServerProvider

- **Resource Server** – serverul web al aplicației

Userul denumit și resource owner, prezintă setul de credentiale aplicației client. Clientul transmite credentialele componenței AuthorizationServerProvider, iar dacă credentialele sunt corecte, este returnat un token criptat aplicației client, care folosește mai departe token-ul primit când face requesturi la serverul web al aplicației, care a fost configurat să accepte Access Tokens, și să le decripteze pentru a confirma identitatea userului. Această configurare a fost făcută în clasă de start a Web API:

```
public void Configuration(IAppBuilder app)
{
	//.. cod de configurare
	ConfigureOAuth(app);
	//.. cod de configurare
}
private void ConfigureOAuth(IAppBuilder app)
{
	var OAuthServerOptions = new OAuthAuthorizationServerOptions
	{
		AllowInsecureHttp = false,
		TokenEndpointPath = new PathString("/token"),
		AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
		Provider = new AuthorizationServerProvider()
	};
	// Token Generation
	app.UseOAuthAuthorizationServer(OAuthServerOptions);
	app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());
	app.UseOAuthBearerTokens(OAuthServerOptions);
}
```

Se poate observa din codul prezentat configurarea autenficarii cu token-uri de tip Bearer, și configurarea componentei AuthorizationServerProvider ca și provider pentru autentificarea în aplicație. Autentificarea se face folosind URL-ul localhost:port\token după cum a fost configurat, iar token-ul este configurat să expire după o zi de la generarea acestuia. În figura de mai jos, Fig. 5.6, este prezentat fluxul de autentificare în sistemul prezentat:

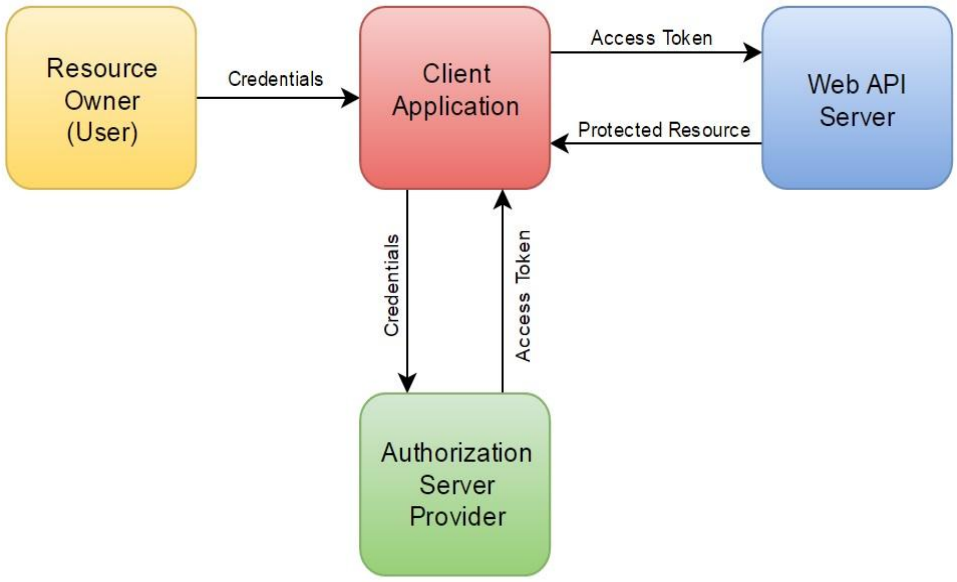

Fig. 5.6 Fluxul de autentificare OAuth

5.1.2.2 Business Logic Layer

Acest layer face legătura între layerul anterior descris și layerul de acces la date. În acest nivel se regăsește logica de business a aplicației care va fi aplicată asupra datelor. Tot acest layer este responsabil cu conversia obiectelor de tip DTO în entități POCO, sau invers, pentru a face posibilă comunicarea pentru layerele adiacente.

Pentru fiecare tip de entitate din aplicație, există câte o clasă de management, care implementează una din interfețele existențe `IStandardManagementService<DTO,POCO>` sau `IExtendedManagementService<DTO,POCO>`. Interfața standard pentru management conține o definiție pentru operațiile CRUD, iar interfața extinsă pentru management conține și alte metode pe lângă cele pentru realizare operațiilor CRUD, cum ar fi metoda care stabilește dacă o entitate poate fi ștearsă. Pentru a putea facilita fluxul de date, pentru fiecare tip de entitate este nevoie de un serviciu de conversie a acesteia. Serviciile de conversie a entitațiilor vor implementa interfața `IConverterService<D,P>`, unde D reprezintă clasa DTO iar P reprezinta clasa POCO. Această interfața conține două metode care trebuie implementate: ConvertToPoco(D entity) și ConvertToDto(P entity).

Pe lângă facilitarea fluxului de date între layere, acest nivel este responsabil efectuarea unor anumite decizii sau calcule care țin de logica aplicației. Un exemplu de decizie de business făcut la acest nivel este aceea dacă o anumită entitate, cum ar fi o întrebare sau o secțiune, poate fi ștearsă din sistem. Astfel la citirea acesteia din bază de date, se stabilesc dependințele acesteia, și se ia o decizie dacă această poate fi ștearsă sau nu. Un alt exemplu de calcul care ține de logica aplicației, este calcularea notei obținută pe un test. Astfel, sunt aduse din baza de date toate

informațiile necesare pentru calcularea acesteia (întrebările testului, nota pe fiecare răspuns în parte, ponderea secțiunii din care o întrebare face parte), sunt calculate notele în parte pentru fiecare secțiune precum și o notă finală, după care toate aceste informații sunt transmise mai departe spre client, unde vor fi afișate.

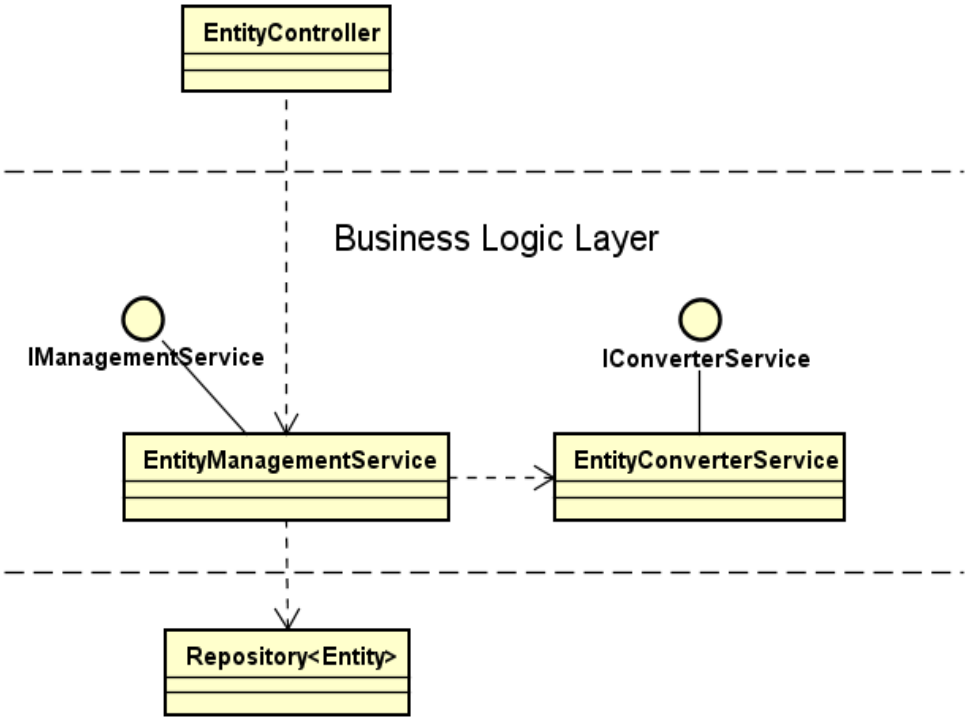

Fig. 5.7 Diagrama de clase a nivelului de business logic din perspectiva unei entități

În figura de mai sus, Fig. 5.7, este prezentată diagrama de clase a nivelului de business, din perspectiva unei entități. Pentru fiecare tip de entitate din aplicație scenariul este același.

5.1.2.3 Data Access Layer

Scopul acestui nivel este de a asigura și a implementa conexiunea cu bază de date. Pentru construirea eficientă a nivelului, se consideră evitarea codului duplicat, care crește șansa de a introduce erori de programare. În vederea atingerii acestui lucru, s-a folosit un repository generic care funcționează cu orice tip de entitate, separând logica de business de logica de acces la date. Acest repository are rolul de a lega nivelul de business cu nivelul de de acces din aplicație. Separarea adusă are următoarele beneficii:

- Centralizează logica
- Furnizează o arhitectură flexibilă care poate fi adaptată atunci când aplicația evoluează
- Imbunătățește mentenabilitatea coduluui

Acest nivel a fost implementat folosind framwork-ul Entity Framework, folosind abordarea Code-First care mapeaza entitățile și relațiile dintre acestea la bază de date. În acest fel sunt facilitate următoarele:

- Materializarea datelor returnate de baza de date ca și entități ale aplicației
- Urmarirea schimbărilor care au loc pe obiecte
- Gestionarea accesului concurent la baza de date
- Propagarea schimbărilor care au avut loc pe obiecte înapoi în baza de date

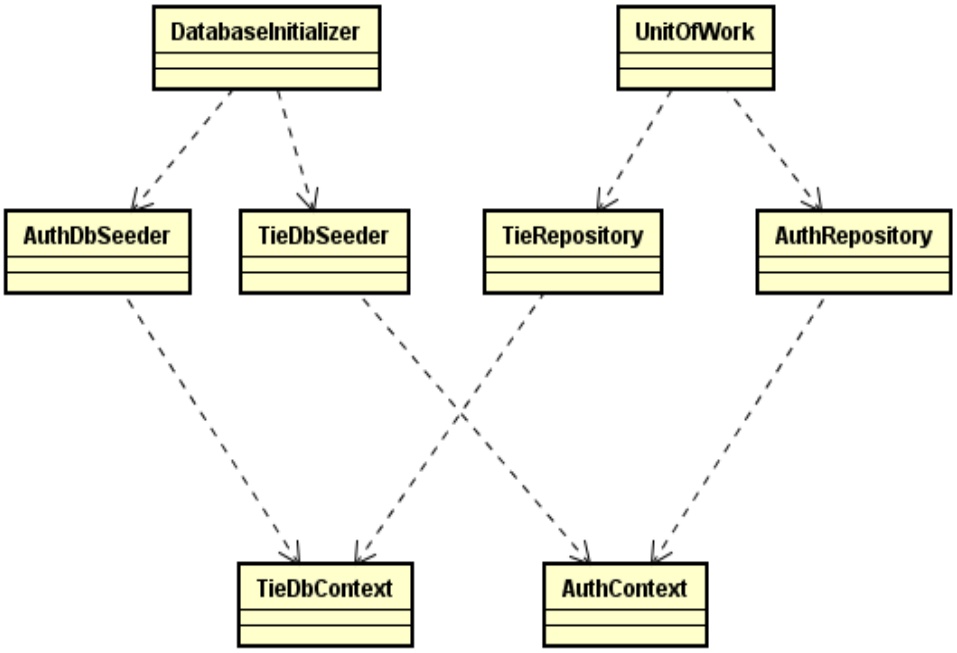

Fig. 5.8 Diagrama de clase a nivelului de acces la date

Clasele primare care sunt responsabile cu interacțiunea datelor sub formă de obiecte sunt clasele TieDbContext și AuthContext care sunt derivate din clasa DbContext furnizată de EF. Clasele context administrează obiectele entitate la run time, incluzând popularea obiectelor cu date din baza de date, urmărirea schimbărilor care au fost efectuate asupra obiectelor și persistența datelor la baza de date. Clasele context expun proprietăți de tipul DbSet<entity> care reprezintă colecții de entități specifice, care în cazul aplicației sunt întrebările, răspunsurile, etc. TieDbContext înglobează toate entitățile de business ale aplicației mai puțin entitățile care țin de autentificare, care vor fi gestionate separat de clasa AuthContext. Accesând o proprietate de tip

DbSet reprezintă executarea unei interogări care aduce toate datele de tipul respectiv. Accesul la DbSet-uri se face prin intermediul repositoryului generic, astfel încât nu este nevoie de cod specific pentru fiecare tip de entitate, codul fiind centralizat într-un singur loc.

În dezvoltare aplicației s-a folosit pattern-ul Unit of Work, care combină un set de interacțiuni și le comite împreună folosind o tranzacție. În clasa UnitOfWork pot fi accesate clasele repository, pentru a efectua operații de inserare, modificare și ștergere asupra datelor. La apelarea metodei Save() din această clasa, toate modificările aduse până în punctul de față sunt înglobate într-o tranzacție, care este transmisă către baza de date. Punerea în aplicare a acestui pattern ajută la izolarea aplicației de modificările care au loc în depozitul de date și avantajează automatizarea testării.

Acest nivel mai dispune de o componentă responsabilă cu inițializarea bazei de date. În momentul creării bazei de date, această este inițializată de către clasa DatabaseInitializer cu un set de date specificat în clasele TieDbSeeder și AuthDbSeeder.

5.2 Deployment

În diagrama de deployment sunt prezentate componentele hardware sau nodurile pe care rulează artefactele (componentele software) ale sistemului prezentat, dar și modul în care aceste componente sunt interconectate. Scopul acestei diagrame este de a prezenta structura hardware necesară rulării sistemului.

Sistemul conține urmatoarele noduri:

- Client – PC - pe acest nod este situată aplicația client. Aplicația web poate fi accesată și de pe orice dispozitiv mobil, însă dezvoltarea interfeței grafice a utilizatorilor a fost concepută în principal pentru ecrane de dimensiuni ridicate.
- Web Server – pe acest nod este situată aplicația web, care are trei componente de bază: Request Handling, Business Logic și Data Access.
- Sql Database Server – pe acest server este situată baza de date a aplicației

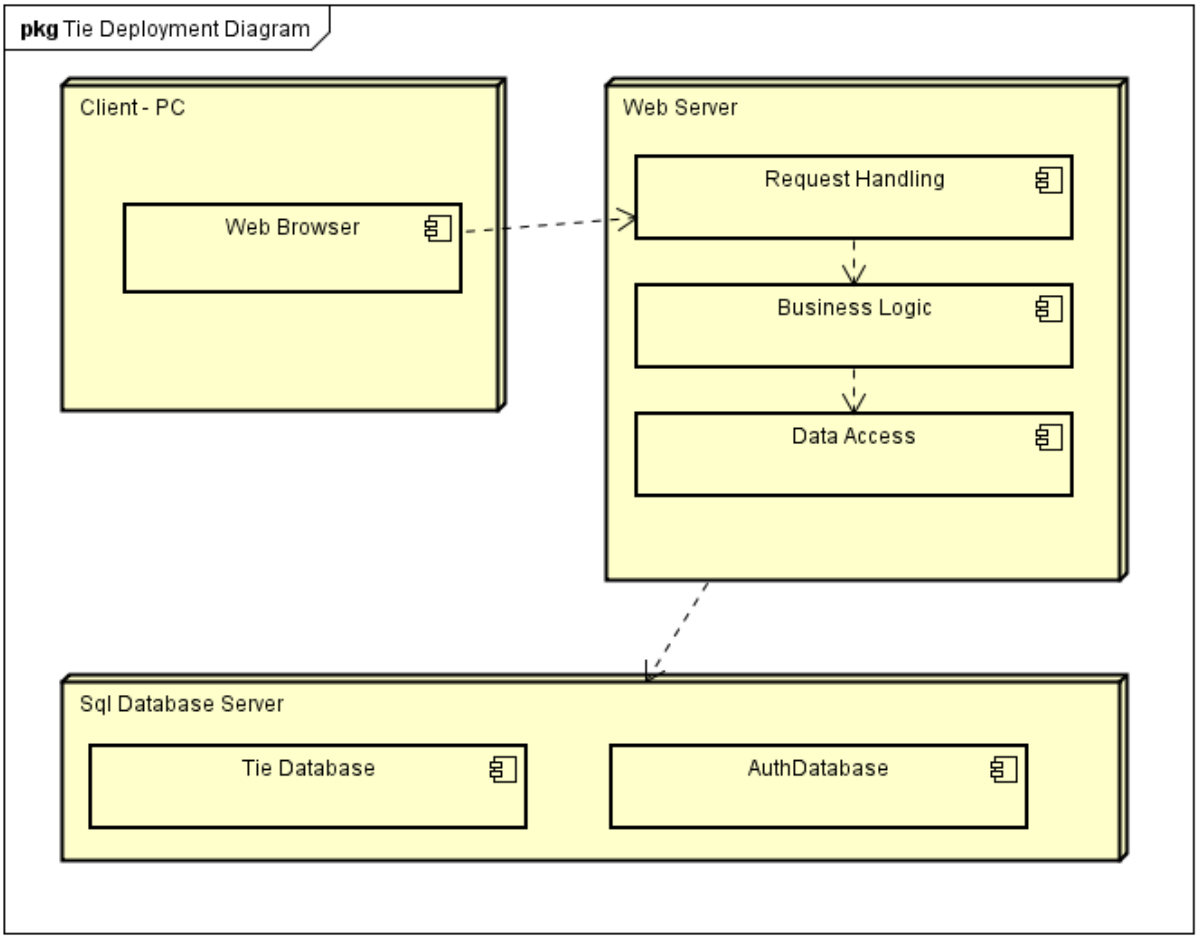

Fig. 5.9 Diagrama de deployment a sistemului

5.3 Proiectarea bazei de date

Aplicația utilizează o bază de date care conține 15 tabele, tabele structurate în două cataloage diferite, Auth și Tie. Tabelele din catalogul Auth sunt responsabile cu stocarea userilor,a rolurilor și a claim-urilor acestora, iar tabelele din catalogul Tie sunt responsabile cu stocarea datelor de business, cum ar fi întrebările, testele, răspunsurile. Baza de date a fost construită pe platforma oferită de Microsoft SQL Server.

În proiectarea bazei de date s-a dorit reprezentarea cât mai corectă a informațiilor și diminuarea posibilității de a ajunge la informații eronate. În vederea atingerii acestui scop, s-a folosit tehnica de normalizare a bazei de date. Mai jos sunt descrise formele normale pe care baza de date le respectă.

Forma normală 1, FN1, presupune că la intersecția unei linii cu oricare coloană, să se găsească un câmp care conține exact o valoare. O tabelă nu poate avea grupuri repetitive iar fiecare atribut poate să primească doar o valoare atomică. Baza de date construită satisface aceste constrângeri, deoarece fiecare atribut al unui tabel primește doar o valoare atomică.

Forma normală 2, FN2, este îndeplinită dacă FN1 este îndeplinită și fiecare atribut care nu aparține cheii primare, este total dependent funcțional de cheia primară. Forma normală 2 trebuie verificată la relațiile care au cheie compusă din mai multe atribute pe post de cheie primară. Deoarece pentru toate relațiile din baza de date cheia este compusă dintr-un singur atribut, baza de date a sistemului se află în forma normală 2.

Forma normală 3, FN3, deste îndeplinită dacă FN1 și FN2 sunt îndeplinite, și dacă nu există niciun atribut care să nu aparțină cheii principale și care să fie tranzitiv dependent de cheia principală, (Dependentă tranzitivă: Dacă atributele A, B, C sunt în relațiile A→B și B→C, atunci spunem că atributul C este dependent tranzitiv de atributul A, prin B). În general, astfel de dependințe există doar dacă tabelul stochează date din mai multe domenii, de exemplu dacă în tabela Test am reține template-ul folosit pentru acest test și sesiunea de testare pentru care a fost definită template-ul.

Forma normală Boyce-Codd (BCNF): O relație este în formă normală Boyce-Codd dacă și numai dacă orice determinant din relație este cheie candidat.

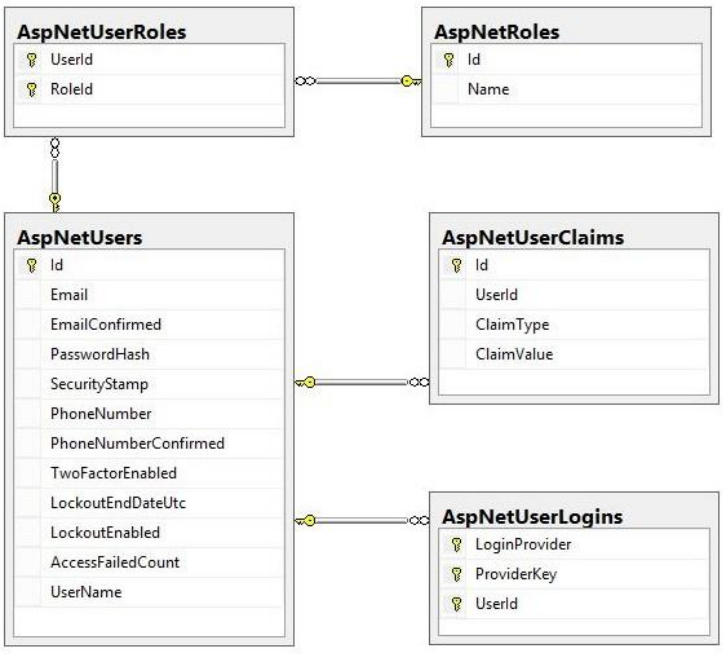

Fig. 5.10 Diagrama bazei de date (catalogul Auth)

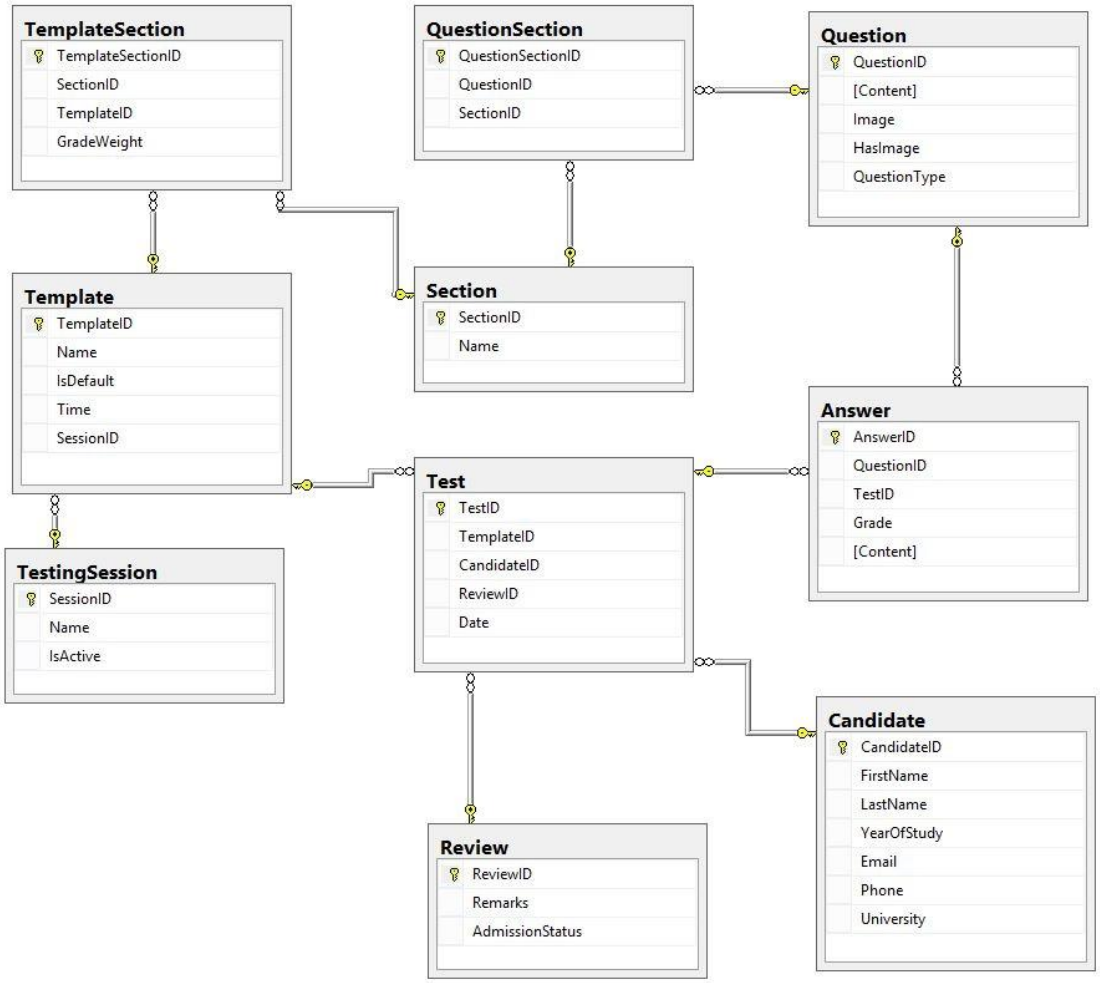

Fig. 5.11 Diagrama bazei de date (catalogul Tie)

Capitolul 6. Testare, Validare si Evaluare

În acest capitol vor fi prezentate principalele procese de testare care au fost realizate asupra sistemului propus. Pentru acest proiect, a fost construită o suită de teste pentru funcționalitățile importante ale aplicației,suită de teste compusă din mai multe cazuri de testare. Un caz de testare este un set de condiții cu ajutorul cărora care un tester va determina dacă aplicația, sau o caracteristică a acesteia, funcționează așa cum a fost stabilit inițial să funcționeze.

Un caz de test oferă cunoștințele funcționale pentru o aplicație și conține condițiile prealabile ale cazului de testare, adică în ce stare ar trebui sistemul să fie astfel încât funcționalitatea dorită este disponibilă, toate măsurile necesare pentru a fi efectuate pentru ca această funcționalitate să ajungă la scopul său, un rezultat așteptat pentru fiecare etapă a cazului de testare, și în cele din urmă, rezultatul așteptat al cazului de testare, și anume obiectivul cazului de testare. În cele ce urmează vor fi prezentate cele mai semnificative cazuri de testare folosite pentru testarea aplicației.

Caz de testare: Testarea creării unui test

Precondiții:

- Un administrator este logat in sistem

Acțiune	Rezultat așteptat
1. Navigare la submeniul Questions din meniul Test management	O pagină cu întrebările disponibile în sistem este afișată
2. Apasarea butonului “Create new”	Este afișată pagina de creare a unei noi întrebări
3. Completarea corectă câmpurilor și apăsarea butonului “Add question”	Întrebarea este adăugată în baza de date, userul este redirecționat la pagina care conține întrebările disponibile, nouă întrebare regăsindu-se printre ele
4. Navigare la submeniul Sections din meniul Test management	O pagină cu secțiunile disponibile în sistem este afișată
5. Apasarea butonului “Create new”	Este afișată pagina de creare a unei noi secțiuni
6. Introducerea numelui secțiunii și apăsarea butonului “Add section”	Nouă secțiune este adăugată, și este vizibilă în pagina care afișează secțiunile
7. Identificarea secțiunii nou adăugate, click pe butonul “Edit”	Userul este redirecționat la pagina de management a secțiunii respective
8. Selectarea întrebării anetrior adăugate în sistem și apăsarea butonului “Add question”	Întrebarea este adăugată la secțiune
9. Apăsarea butonului “Update question”	Secțiunea este salvată cu succes
10. Navigare la submeniul Templates din meniul Test management	O pagină cu templateurile disponibile în sistem este afișată
11. Apăsarea butonului “Create new”	Este afișată pagina de creare a unei noi secțiuni
12. Completarea numelui, selectarea sesiunii de testare “.Net” și a timpului de testare și apăsarea	Templateul este adăugat cu succes și este vizibil în lista templateurilor disponibile în aplicație

butonului “Add template”	
13. Identificare noului template și apăsarea butonului “Edit”	Userul este redirecționat la pagina de management a tempalteului respectiv
14. Selectarea templateului ca default și apăsarea butonului “Update”	Este afișat un mesaj de eroare în care userul este informat ca deja există un template default pentru sesiunea “.Net”, indicând numele acestuia
15. Dezactivarea templateului indicat ca template default	Setarea este salvată
16. Accesarea din nou a meniului de management a templateului nou adăugat, și adăugarea secțiunii create anterior cu grade weight setat la valoarea 100	Secțiunea este adăugată cu succes la template
17. Setarea templateului ca default și apăsarea butonului “Update template”	Datele sunt salvate cu succes și userul este redirecționat la pagina de vizualizare a templateurilor

Caz de testare: Testarea susținerii unui test

Precondiții:

- Serviciile expuse de aplicație sunt disponibile
- Cazul de testare denumit “Testarea creării unui test” a fost deja rulat

Acțiune	Rezultat așteptat
1. Pornire aplicație	Pagina principală este afișată, este vizibilă bara de navigare afișând doar controalele necesare pentru login
2. Introducerea unor credențiale greșite	Câmpul care conține parola este mascat
3.Apăsarea butonului “Log in”	Este afișat un mesaj de eroare care avertizează introducerea unor credențiale greșite
4. Introducerea unor credențiale corecte pentru un cont de tip candidat	Logarea este efectuată cu succes, userul este redirecționat la pagina de începere a testului, unde îi este prezentat un formular
5. Completarea numelui,a numărului de telefon, a universității, și a anului de studiu	Butonul “Take test” este dezactivat, deoarece nu au fost acceptați termenii și condițiile testului
6. Completarea emailului cu date invalide, selectarea checkboxului “I agree” și apăsarea butonului “Take test”	Userul este informat de faptul că emailul este invalid iar testul nu începe
7. Completarea unui email valid, apăsarea butonului “Take test”	Testul începe, fiind prezentat testul definit anterior în cazul de testare “Testarea creării unui test”
8. Completarea unui răspuns la întrebare (de exemplu “răspuns”) și apăsarea butonului Submit	Apare un mesaj în care utilizator este întrebat dacă vrea să încheie testul
9. Apăsarea butonului “No”	Testul nu este închis, iar răspunsul introdus anterior poate fi alterat
10. Apăsarea butonului “Submit” și apoi “Yes”	Testarea ia sfârșit, răspunsul este salvat în baza de date iar userul primește un mesaj de informare

În prezent, sistemul este utilizat în cadrul companiei Accesa din Cluj-Napoca, pentru a eficientiza procesul de intrenship. Pentru validarea gradului de satisfacție adus de interfața cu utilizatorul, beneficiarii aplicației, atât candidații cât și utilizatorii administrativi (Administrator și Hr) au fost să accorde rugați un calificativ pe o scară de la 1 la 5 interfeței, din punctul de vedere a aspectului și a intuitivității acesteia. Rezultatele vor fi prezentate prin diagramele de mai jos.

Din evaluarea interfeței de către 10 utilizatori administrativi, a rezultat un scor de 4.0 pentru intuitivitate și 4.3 pentru aspect:

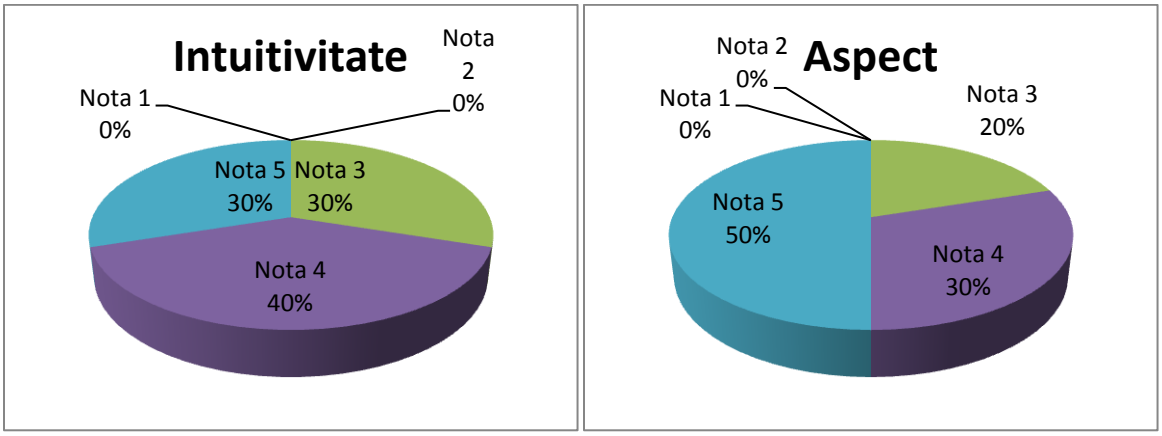

Fig. 6.1 Notarea interfeței de către utilizatorii administrativi

Din evaluarea interfeteii de către 22 utilizatori candidați, a rezultat un scor de 4.27 pentru intuitivitate și 4.36 pentru aspect.

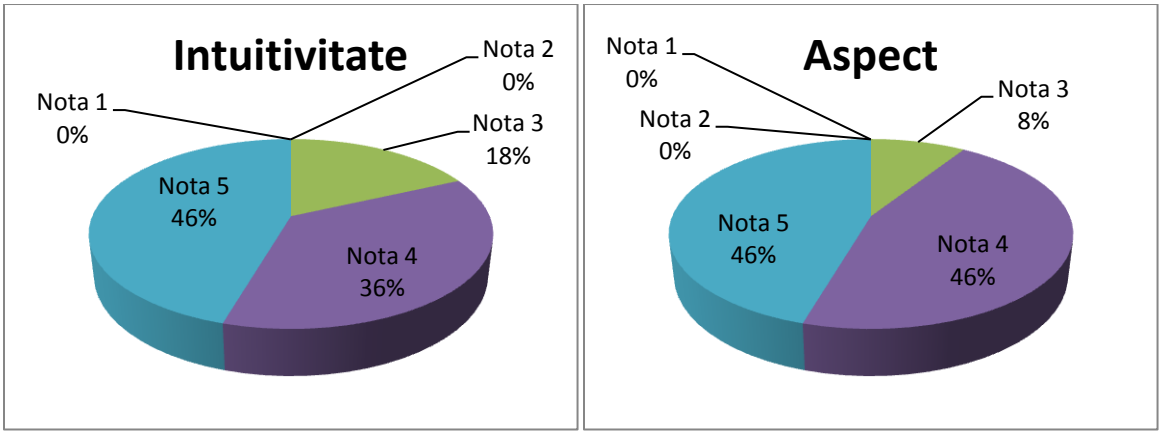

Fig. 6.2 Notarea interfeței de către utilizatorii candidat

Capitolul 7. Manual de Instalare și Utilizare

În acest capitol sunt prezentați pașii care trebuie urmați pentru instalarea cu succes a componentelor sistemului pe o mașină locală sau pe un server într-o rețea locală. Tot în acest capitol vor fi prezentate resursle software și hardware necesare rulării, precum și un manual de utilizare a aplicației.

7.1 Instalarea aplicației

Deploymentul aplicației se va face în rețeaua companiei, astfel orice calculator conectat la rețeaua locală va acces la aplicație. Utilizatorii aplicației nu trebuie să dispună de resurse hardware sau software specifice, ci doar de accesul la un browser web. Pentru aplicația care rulează pe server, sunt identificate următoarele dependințe:

- Sistem de operare din gama Windows, care dispune de componenta IIS (Internet Information Services), cu ajutorul careia va fi lansata aplicatia
- Microsoft SQL Server
- Microsoft .Net Framework 4.0 sau o versiune mai noua

Pentru instalarea aplicației, următorii pași trebuie urmați. Componentele cât și baza de date pot fi găzduite pe aceiași mașină sau pe mașini diferite.

7.1.1 Găzduirea componentelor aplicației

Se vor crea două foldere separate, care vor conține fișierele executabile ale aplicației client și respectiv aplicației care rulează pe server (de ex. D:Tie Client și D:Tie Server), care vor fi utilizate pentru gazuirea componentelor. Pentru găzduirea unei componente se executa următorii pași (acești pași vor fi executați atât pentru aplicația client cât și pentru aplicația server):

1. Logarea in sistem cu un cont de aministrator
2. Se accesează Internet Services Manager din Control Panel → Administrative Tools
3. Dupa deschiderea IIS Manager, se expandează folderul web server, iar dupa folderul Sites. Click dreapta pe folderul sites si se selectează optiunea “Add New Website”

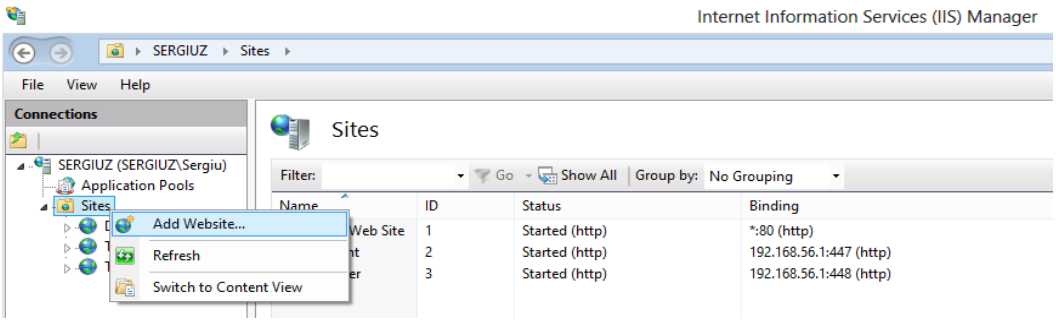

Fig. 7.1 IIS - Adăugarea unui website

4. In fereastra “Add New Website” vor fi completate urmatoarele informatii:
   4.1. Site Name – numele site-ului
   4.2. Physical Path – locatia pe server-ul local unde sunt ținute fișierele website-ului (în cazul nostru D:\Tie Client sau D:\Tie Server)
   4.3. Type – se va selecta HTTP sau HTTPS în cazul in care site-ul foloseste un certificat SSL
   4.4. IP Address – Ip-ul la care site-ul va răspunde
   4.5. Host name – un nume de domeniu (ex. Internship.companie.ro)
5. Se selectează noul site adăugat, iar în fereastra Features View se va selecta “Default document”, unde vom specifica fișierul care va fi inițial executat la accesarea site-ului. În cazul aplicației client se va selecta “index.html” iar în cazul aplicației server se va selecta “Tie.API.Services.dll”.

7.1.2 Instalarea bazei de date

Va fi necesară instalarea Microsoft SQL Server Express[16], care de asemenea implica instalarea Microsoft .Net Framework 4.0 sau o versiune mai nouă. Dupa instalarea Sql Server Express[17], vor fi executați următori pași:

1. Se va deschide Command Prompt cu drepturi de administrator
2. Se va crea instanța care contine baza de date responsabilă cu autentificarea folosind comanda *SqlLocalDb create Auth*
3. Se va crea instanța care conține baza de date cu tabelele aplicației folosind comanda *SqlLocalDb create Tie*
4. Se vor porni instanțele ulterior create folosind comenzile *SqlLocalDb start “Auth”* și *SqlLocalDb start “Tie”*

7.1.3 Configurări

După găzduirea componentelor și crearea instanțelor bazei de date, va fi necesară efectuarea unor configurări. În folderul unde se regăsesc fișierele aplicației server (D:Tie Server), se localizează și se deschide fișierul “Tie.API.Services.dll.config”:

1. Sub nodul “appSettings”, se va identifica setarea cu cheia “origins”. Se va seta valoarea acesteia la IP-ul care a fost asignat aplicației client, astfel va fi permis accesul acesteia la server. Dacă se dorește permiterea accesului unui client indiferent de locația la care acesta se află, se va seta cu valoarea “*”.

[16] Tutorial de instalare Sql Server Express
http://www.sherweb.com/blog/how-to-install-sql-server-2012-express-edition/
[17].Net Framework 4.0 – link pentru download
https://www.microsoft.com/en-us/download/details.aspx?id=17851

2. Sub nodul “connectionStrings” se găsesc două elemente care definesc connectionString pentru cele două cataloage. Se vor seta calea corectă către acestea sub forma host:port\instanta

In folderul unde se regasesc fișierele aplicației client (D:\Tie Client), se va accesa folderul “app” după care se localizează si se deschide fisierul “app.common.js”.

3. Se va seta constanta “serverPath” la adresa ip la care se regasește aplicația server.
4. Se va seta constanta “secureServerPath” la adresa ip la care este găzduită aplicația server care folosește certificate SSL (dacă pentru aplicația server se va folosi protocolul https)

La prima accesare a aplicației, vor fi populate tabelele și datele folosite în aplicație în mod automat, aceasta fiind gata de utilizare.

7.2 Manual de utilizare

Pentru utilizarea sistemului, utilizatorii trebuie mai întâi să se autentifice îs sistem, introducând usernamul și parola în câmpurile dedicate din bara de navigație, după care vor apăsa butonul “Log in”.

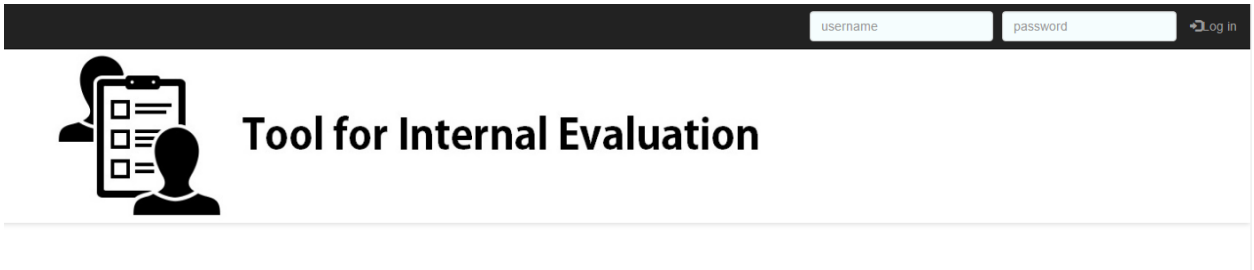

Fig. 7.2 Autentificarea în sistem

Pentru administrarea întrebărilor, secțiunilor sau a template-urilor, administratorii vor folosi meniul “Test Management”, care apare după autentificarea în sistem. Pentru corectarea unui test se va folosii meniul “Reviews” iar pentru vizualizarea testelor se va folosii meniul “Reports”.

Candidații care doresc să susțină un test vor completa mai întâi formularul care le va fi prezentat în momentul autentificării în aplicație. Pentru începerea unui test, mai întâi utilizatorii vor accepta termenii și condițiile testului. Fiecare test are o constrângere de timp, dupe expirarea căreia testul va fi finalizat fără accordul candidatului.

Capitolul 8. Concluzii

În acest capitol vor fi aduse concluzii asupra sistemului dezvoltat. Se vor trecere în revistă realizările și obiectivele care au fost atinse prin acest proiect, urmată de o descriere a posibilităților de dezvoltare ulterioară.

8.1 Realizările sistemului

Sistemul realizat reușește să își atingă scopul, eficientizând cu succes procesele de recrutare sau internship. Obiectivele propuse pentru acest proiect au fost realizate, astfel sistemul oferă capabilitatea de creare și management a întrebărilor, secțiunilor și a templateurilor care pot fi asociate în orice combinație dorită. Posibilitatea de centralizare a testelor, asocierea acestora la sesiuni de testare definite de utilizator în funcție de care se pot efectua filtrări, corectarea rapidă și acordarea de calificative, precum și vizualizarea eficientă a rezultatelor pe care se pot efectua căutări și filtrări ajută semnificativ la îmbunătățirea și eficientizarea proceselor de internship din cadrul unei companii. Utilizatorii sunt clasificați pe roluri, fiecarea având meniuri și resurse specifice pe care le pot accesa. A fost asigurată și securitatea aplicației care folosește autentificarea pe bază de tokenuri, iar parolele reținute în baza de date sunt fost criptate în cazul unui acces neautorizat la aceasta.

Sistemul reușește să livreze o interfață grafica intuitivă și ușor de învățat, lucru susținut rezultatele evaluării realizate utilizatorii aplicației din cadrul companiei, cât și de candidații care au susținut testele de internship pe această aplicație.

8.2 Dezvoltări ulterioare

Șansele ca un sistem informatic să fie supus unor schimbări sau dezvoltări ulterioare sunt mari, motiv pentru care sistemului prezentat a fost proiectat să fie extensibil. În cele ce urmează va prezentată o listă cu îmbunătățiri care pot fi aduse acestui sistem sau posibilități de dezvoltare ulterioară.

- Dezvoltarea unei componente online care să se ocupe cu automatizarea programărilor reprezintă o posibilitate de dezvoltare ulterioară pentru sistem. Astfel un candidat își va putea completa online datele personale, iar sistemul îi va rezerva automat o zi și o oră din programul disponibil pentru evaluare, scutiind organizatorii de necesitatea de a stabilii o întâlnire cu fiecare candidat în parte.
- O altă posibilitate de dezvoltare ulterioară este adăugarea de noi clienți, atât desktop cât și mobile, oferind o flexibilitate mai mare în organizarea proceselor de testare
- O îmbunătățire posibila care ar putea fi adusă sistemului este posibilitatea de configurare a mai multor tipuri de întrebări pentru un test. La momentul actual, aplicația suporta doar întrebari simple, a căror răspuns va fi completat într-un camp aflat sub acestea. Astfel pentru un test vor putea fi folosite întrebări de tip grilă, posibilitatea de completare a spațiilor goale

sau crearea de asocieri, sau chiar şi desenarea unor diagrame ca răspuns la o întrebare, făcând procesul de testare mult mai interactiv pentru candidați

- O altă îmbunătățire posibilă care ar putea fi adusă acestui sistem este dezvoltarea unui tool de print/export a testelor și a rezultatelor acestora, în funcție de o anumită perioadă de timp, sesiune de testare, etc.
- O altă dezvoltare ulterioară ar putea fi constituită de implementarea unei componente contactare automată a candidatiilor prin e-mail sau SMS în momentul când testul lor a fost corectat. Aceasta componentă ar scuti organizatorii evenimelor de internship de contactarea individuală a fiecărui candidat, astfel eficientizănd si mai mult procesul.

Bibliografie

[1] Robert C. Martin, "*Clean Code: A Handbook of Agile Software Craftsmanship*", Prentice Hal, 1st edition, 2008.

[2] Cody Lindley, "*JavaScript Succinctly*", Syncfusion, 2012.

[3] *"The OAuth 2.0 Authorization Framework"*, Microsoft, Octombrie 2012, ISSN: 2070-1721, Sectiunea 4.3,
Disponibila la: https://tools.ietf.org/html/rfc6749#section-4.3

[4] "*Three Tier Client/Server Architecture: Achieving Scalability, Performance"*, and Efficiency in Client Server Applications." Open Information Systems 10, Ianuarie 1995

[5] Chen, Lianping; Ali Babar, Muhammad; Nuseibeh, Bashar, "Characterizing Architecturally Significant Requirements",Aprilie 2013, *IEEE Software* **30** pag 38 – 45 ,Disponibilă la: doi:10.1109/MS.2012.174

[6] Documentația oficială Web API : http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api

[7] Documenația oficială Owin and Katana, http://www.asp.net/aspnet/overview/owin-and-katana

[8] Documentația oficială pentru AngularJS API, https://docs.angularjs.org/api

[9] Documentația oficială Entity Framework, https://msdn.microsoft.com/en-us/data/ee712907.aspx

[10] HTML & CSS tutorials, http://learn.shayhowe.com/

[11] MySql Reference Manual, https://dev.mysql.com/doc/refman/5.7/en/c-api.html

[12] C# Coding Conventions, https://msdn.microsoft.com/en-us/library/ff926074.aspx

[13] Bootstrap Components, http://getbootstrap.com/components/

Anexa 1 – Lista figurilor și a tabelelor din lucrare

Lista figurilor din lucrare

Lista tabelelor din lucrare

Anexa 2 – Glosar de termeni

API	Application Programming Interface
SPA	Single-Page Application
HTTP	HyperText Transfer Protocol
CSS	Cascading Style Sheet
W3C	World Wide Web Consortium
REST	Representational State Transfer
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
LINQ	Language-Integrated Query
DOM	Document Object Model
AJAX	Asynchronous JavaScript
CRUD	Create, Read, Update, Delete
SQL	Structured Query Language
UML	Unified Modeling Language
IIS	Internet Information Services
DTO	Data Transfer Object
CLR	Common Language Runtime
POCO	Plain-Old CLR Object