CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

Platformă pentru evaluarea cunoștințelor pe domenii

Coordonator științific, Absolvent,

Ș.l. dr. ing. Mihai Vlase Sandu Alexandru

Galați

2018

Cuprins

[Introducere 3](#_Toc518081672)

[Justificarea temei 4](#_Toc518081673)

[Cerințe și specificații 6](#_Toc518081674)

[Analiza problemei 8](#_Toc518081675)

[Tipuri de utilizatori 8](#_Toc518081676)

[Elementele aplicatiei 8](#_Toc518081677)

[Cazuri de utilizare 10](#_Toc518081678)

[Proiectarea bazei de date 13](#_Toc518081679)

[Proiectarea interfeţei 15](#_Toc518081680)

[Tehnologii 21](#_Toc518081681)

[Implementarea aplicaţiei 25](#_Toc518081682)

[Concluzii 44](#_Toc518081683)

[Bibliografie 45](#_Toc518081684)

# Introducere

O platformă de evaluare este o aplicaţie (utilizată de regulă în spaţiul Web) complexă creată cu scopul testării cunoştinţelor acumulate într-un domeniu, abilităţilor motorii, psihomotorii, dar şi evoluţiei copiilor preşcolari conform nivelului specific de dezvoltare intelectuală, testarea capacităţii de efectuare a unor operaţii aritmetico-logice.

Platformele de evaluare prezintă un trend ascendent pe piaţa din România, ele fiind utilizate nu doar în scop educaţional, dar mai ales în domenii precum mediciă, inginerie şi multe altele. Practic, aproape că nu există domeniu al vieţii ştiinţifice ori sociale, în care evaluarea cu ajutorul unei aplicaţii – platformă să nu fie prezentă. Ele pot fi utilizate în cadru organizat, în interiorul unor grupuri de lucru, cu un scop comun sau individual, de curiozitate, de nevoie, ori de plictiseală.

În momentul de față, trendul este către mediul online, cu tot mai multe site-uri și platforme fiind pe internet pentru a ajuta elevii și studenții să se pregătească la diferite materii/pe diferite domenii.

Două astfel de exemple, din două tabere să spunem așa relativ diferite ar fi platformă AEL și platformă ExamenulTau, prima având mai mult rol pedagogic, și fiind folosite de către cadrele didactice, iar a doua fiind folosită mai mult de către elevi pentru verificarea cunoștințelor înainte de examen, spre exemplu.

Ambele au la bază parcurgerea unor cursuri sau lecții preliminare și apoi susținerea unui test pentru verificarea sedimentării cunoștințelor respectivului.

Totodată, cu atâtea surse de informație la îndemână prin intermediul internetului, deja nu mai revine în rolul platformei de evaluare să fie totodată cea care se ocupă și de pare de instruire înainte de evaluarea elevilor/studenților.

Astfel, se poate folosi efortul depus în crearea lecțiilor și tutorialelor pentru îmbunătățirea strategiilor și algoritmilor de evaluare, sau pentru acoperirea a mai multor arii.

## Justificarea temei

Am ales crearea unei astfel de platforme întrucât consider că poate ajută elevii și studenții să își poate îmbunătăți cunoștințele de cultură generală, având în vedere că acestea nu sunt întotdeauna corelate cu studiile și totodată nu întotdeauna cunoștințele și lecțiile din sistemul de învățământ se translatează mai apoi în mentalitatea individului că și noțiuni de cultură generală.

Implementarea aplicației este o recomandare pentru persoanele foarte concentrate pe un singur domeniu, caz în care cunoștințele din alte domenii cumva sunt destul de scăzute, și se vrea schimbarea acestui aspect. Totodată, este util și pentru îmbunătățirea cunoștințelor pe un domeniu selectat, indiferent de nivel, întrucât se poate permite ajustarea nivelului de dificultate.

Ca și aspecte ce ar face diferența de ce este deja pe piață, se poate scoate în evidența în primul rând faptul că, odată cu înregistrarea utilizatorului, când acesta accesează platformă pentru prima oară, are ocazia să susțină o evaluare inițială, în funcție de care i se va stabili nivelul de dificultate de la care începe.

Acesta este un lucru foarte benefic pentru utilizatorii care au deja un nivel în domeniul respectiv, și îi ajută să treacă peste primele capitole, relativ ușoare pentur ei, comparativ cu nivelul estimat, lucru care de altfel i-ar putea plictisi. Un modul nou, menit să confere aceeași experiență inițială utilizatorilor, indiferent de nivelul lor inițial.

Un alt lucru cu care vine că și noutate este sistemul de knowledge safeguard, care nu permite utilizatorului să susțină testul pentru promovare la nivelul următor decât dacă are scoruri bune la utlimele 3 sesiuni de training, Asta previne învățarea mecanică/norocul, întrucât întrebările sunt aleatorii din bază de date, și deci se înlătura astfel probabilitatea că utilizatorul să promoveze la nivelul următor fără să aibă cunoștințele necesare, făcând astfel sistemul de evaluare cât mai stabil, complet și precis.

Totodată, o observație la platformele existente este faptul că încearcă să implice și partea de dinainte de testarea propriu-zisă, care ar trebui să fie scopul principal, însă pe platformele respective accentul este pus pe partea pedagogică și pe tutoriale, evaluarea cunoștințelor la final fiind doar o formalitate.

Fiind astfel concentrate pe partea preparativa, de dinainte de evaluarea propriu-zisă, ele pierd însăși esența, și anume testarea individului, punând accentul pe aspectele de dinaintea evaluării. Acest lucru va fi evitat de aplicația în cauză, întrucât această nu are alt scop decât evaluarea elevilor și studenților, iar aceștia sunt responsabili singuri de pregătirea de dinainte de evaluare.

Un lucru inovativ cu care vine această aplicație este că, prin conceptul de a se concentra doar pe evaluarea studentului, se lasă la latitudinea studentului/elevului că acesta să se pregătească cum consideră, cultivându-se astfel simțul autodidact, de care acesta are nevoie din ce în ce mai mult pe parcursul ciclului didactic cât și după terminarea acestuia, în special în domeniile tehnice, și nu numai.

Totodată, platformă poate fi folosită și în scop recreativ, în sensul că utilizatorii pot concura între ei, cu scopul de a ajunge la un nivel cât mai avansat, oferindu-li-se astfel posibilitatea de a fi într-o competiție recreaționala, unde se stimulează totodată activitatea intelectuală a tinerilor.

Un lucru care are potențialul de a face această aplicație mai ușor de utilizat față de concurență ar putea fi faptul că are un design simplist, orientat pe funcționalitate și rezultate, nu pe aspect. Majoritatea aplicațiilor au mult bloatware (soft și module adiționale), care de cele mai multe ori distrag cursantul de la scopul principal, și anume evaluarea cunoștințelor.

În această aplicație, asemenea distrageri sunt evitate printr-un design minimalist, unde utilizatorul are doar strictul necesar, prezentat într-o metodă cât mai lipsită de distrageri, astfel că acesta se poate concentra pe scopul principal al platformei, și anume, evaluarea cunoștințelor.

# Cerințe și specificații

Scopul general al aplicației este acela de a evalua și stimula prin testare cursanții pe diferite domenii de cultură generală.

Acest lucru se realizează pe bază unor nivele de dificultate, și avansarea cursanților se face prin promovarea la următorul nivel de dificultate, în urmă susținerii unui test, acesta conținând atât întrebări din nivelul curent cât și întrebări din toate nivelele anterioare. În momentul în care utilizatorul ajunge la un punctaj de peste 60%, atunci este promovat la nivelul următor.

Promovarea la un nivel superior nu împiedică un cursant să acceseze nivelele anterioare, să susțină sesiuni de training sau evaluări la nivelele respective. Totodată, se menționează faptul că nici rezultatele la evaluări, nici la sesiunile de training nu se iau în calcul, în special nu se iau în calcul pentru nivelul curent al utilizatorului.

Totodată, platformă are un sistem de asigurarea consistenței rezultatelor, realizat prin introducerea sesiunilor de training. Acestea sunt o modalitate prin care utilizatorul poate susține o versiune simplificată a examenelor, care conține doar întrebări din nivelul curent.

Aplicația menține înregistrate rezultatele de la ultimele trei sesiuni de training, astfel că utilizatorul nu poate promova la nivelul următor, ba chiar nu poate susține examenul către promovare, dacă nu are un punctaj mediu de peste 60% realizat pe bază rezultatelor la ultimele trei sesiuni de training. Întrebările, atât pentru sesiunile de training, cât și pentru cele de examen, sunt alese în mod aleatoriu, la momentul susținerii testelor, asigurându-se astfel o evaluare cât mai imparțială și mai corectă a cursanților.

La primul contact cu platforma, se va prezenta doar un ecran cu posibilitate de logare în aplicație, și o opțiune de înregistrare, în cazul în care utilizatorul este acolo pentru prima oară și nu beneficiază încă de un cont pe platformă.

La înregistrarea cursanților, se va specifica un nume de utilizator specific, care trebuie să nu fie deja luat de către un alt cursant, parola, împreună cu confirmarea acesteia, precum și domeniul de cunoștințe ales de către utilizator.

După logarea utilizatorilor pe platformă, se ajunge la un landing page, cu informații generale atât despre utilizator, cât și despre aplicație. Deasemenea, de pe această pagină de home utilizatorul își poate schimba domeniul, având totuși ca amendament faptul că, odată schimbat domeniul, cursantul își va pierde nivelul de dificultate la care a ajuns, acesta refiind resetat la zero.

În partea din stânga a paginii sunt situate butoanele folosite pentru navigarea pe platformă, și anume în zonă superioara butoanele pentru ajungerea pe pagină de home, butonul folosit pentru delogare, precum și butonul pentru pagină de administrare, în cazul administratorilor.

Mai apoi, în zonă inferioară, se găsesc butoanele pentru navigarea către diferitele nivele de dificultate ale aplicației. Toate butoanele pentru accesul la nivele sunt vizibile încă de la început, însă acestea nu devin funcționale decât în momentul în care cursantul a reușit să promoveze la nivelul respectiv.

Dintre cerințele și specificațiile aplicației, un aspect extraordinar de important, care diferențiază aplicația de restul concurenței, este faptul ca cursanții, la logarea pe platformă, în cazul în care este prima oară când aceștia se conecteza, utilizatorii pot susține o evaluare inițială, prin care li se poate stabili nivelul inițial de cunoștințe în domeniul ales.

Acest aspect este foarte important, întrucât prin stabilirea nivelului inițial, se poate evită parcurgerea primelor nivele de către cursanții cu un nivel ceva mai ridicat de cunoștințe, lucru care ar fi putut să îi plictisească sau să îi facă să își piardă interesul în platformă, întrucât această nu îi stimulează la nivelul adecvat.

Totodată, se pune accentul și pe controlul administratorilor asupra conținutului platformei, astfel că aceștia au o pagină separată pentru activitățile specifice de administrare ale aplicației. Această pagină permite administratorilor să adauge noi întrebări pentru diferite nivele și domenii, totodată editarea celor deja existente precum și ștergerea acestora.

Deasemenea, un administrator poate creă noi domenii pe care să le folosească utilizatorii, și tot acesta poate șterge domenii. Astfel, administratorii au control asupra conținutului bazei de date cu întrebări și domeniile acestora, asigurându-se astfel o experiență cât mai completă și mai bună pentru utilizatori.

# Analiza problemei

## Tipuri de utilizatori

Aplicația are ca target principal utilizatorii tineri, eventual în diferite stagii ale ciclului de învățământ, care vor să își testeze cunoștințele în diferite domenii.

Aceștia de cele mai multe ori aleg aplicația pentru a își îmbunătăți pregătirea pentru un anumit domeniu fie în scop didactic, cum ar fi pregătirea înainte de un examen la o materie pe domeniul respectiv, cât și în scop recreațional, într-o competiție amicală cu colegii spre exemplu, cu scopul de a ajunge la un nivel de dificultate cât mai avansat.

Totodată, aplicația poate fi folosită și de utilizatori ceva mai înaintați în vârstă, datorită mecanismului complet de asigurare a consistenței evaluării cursantului, astfel că firecare examen pentru promovarea la nivelul următor implică atât sesiuni de training, cât și evaluarea propriu-zisă conține întrebări din nivelele anterioare.

Pe lângă utilizatorii cu drepturi normale, mai avem o altă categorie, total diferită, și anume administratorii. Aceștia au rolul de moderatori ai aplicației să spunem așa, în sensul ca aceștia controlează contentul platformei. Ei pot deasemenea acționă că și utilizatorii standard, dar pe lângă asta au și optinea unui panel administrativ de unde pot adăuga, șterge sau edita întrebări, precum și domenii de cunoștințe

## Elementele aplicației

Elementele esențiale ale aplicație se conturează după funcționalitățile platformei și după scopul acesteia. Astfel, se conturează ca și entități ale aplicației însăși întrebările, ele fiind unitatea funcțională, și caracateristici auxiliare ale acestora, cum ar fi domeniile de activitate.

Totodată, acestea sunt structurate și grupate sub formă de diferite tehnici de evaluare, acestea fiind trei la număr, constituite de evaluarea inițială, sesiunea de training și evaluarea pentru promovarea la următorul nivel de dificultate.

Evaluarea inițială are ca scop principal realizarea unui examen preliminar în care se stabilește nivelul cursantului de dinainte de a avea contact cu paltformă. Rolul acestei evaluări este acela de a facilita accesul la platformă al utilizatorului de la un nivel cât mai apropiat al acestuia de cel deja existent, astfel evitându-se plictisirea sau pierderea interesului prin forțarea lui în a trece prin primele nivelel, în cazul în care acesta posedă deja cunoștințe ceva mai avansate în domeniul respectiv. Această evaluare se realizează printr-un sistem care se auto-ajustează în funcție de utilizator, astfel că mereu următoarea întrebare va fi ajustată la un nivel superior sau inferior față de cea precedentă în funcție de răspunsul cursantului.

Sesiunile de training au rolul de a sedimenta cunoștințele cursantului, prin executarea unor evaluări la oscara ceva mai mică, conținând doar întrebări din nivelul curent. Acestea sunt totodată mecanismul prin care platformă asigură un anumit nivel de consistență al cunoștințelor utilizatorului, astfel ca acesta nu poate susține examenul pentru promovare la următorul nivel până nu a are o medie de peste 60% la utlimele trei sesiuni de training la nivelul respectiv.

Testul pentru promovarea la următorul nivel al cursantului este testul prin care utilizatorul are șansa, după ce a dovedit că stăpânește nivelul curent prin acumularea unui punctaj de trecere la sesiunile de training, să susțină examenul și, prin luarea unui puncatj satisfăcător, să promoveze la următorul nivel de dificultate. Totodată, acest examen conține și întrebări aleatorii din nivelele anterioare, astfel că se asigură o evaluare cât mai completă și precisă a cursantului, asigurându-se că acesta stapnaeste în continuare și nivelele anterioare.

Întrebările, că și strucutra, au enunțul, însăși întrebarea în șine, din domeniul din care face parte și nivelul de dificultate căruia ea aparține, și apoi trei variante de răspuns, dintre care una dintre ele este cea corectă. Această organizare este folosit pentru simplificarea ramificării algoritmicii de evaluare, și totodată pentru a familiariza ușor cursantul cu funcționalitatea structurii de evaluare.

Domeniile constituie gruparea după context a întrebărilor, și sunt bază formării diferitelor categorii de întrebări, constituind un criteriu important de diferențiere a acestora, împreună cu nivelele de dificultate.

Nivelele de dificultate ale platformei de evaluare constituie al doilea criteriu principal de diferențiere a întrebărilor, și nu numai, întrucât acesta este mijlocul de departajare a însăși utilizatorul, și anume nivelul de dificultate curent al acestora.

## Cazuri de utilizare

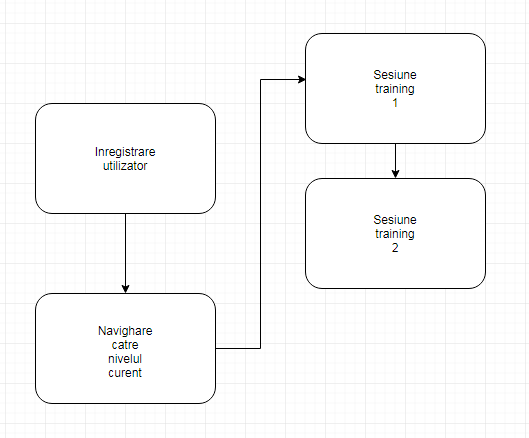
1. Înregistrarea unui utilizator pentru prima oară pe platformă, susținerea examenlui inițial de către acesta și eventual execuția sesiunilor de training și apoi a examenului pentru promovarea la nivelul următor.



Acesta este cazul principal de utilizare al aplicației, în care utlizatorii noi iau contact pentru prima oară cu aplicația, și de cele mai multe ori , ajung să parcurgă mai toate flow-urile majore ale aplicației.

2. Logarea unui utilizator deja existent pentru executarea câtorva sesiuni de training pentru nivelul curent.

Acesta este cazul cel mai des întâlnit pe platformă, în care utilizatorul vizitează zilnic aplicația pentru a se antrena, executând câteva sesiuni de training pentru a se pregăti de un viitor eventual examen de promovare la nivelul următor, în momentul în care acesta se simte pregătit să îl susțină, și desigur, întrunește criteriile necesare, și anume să aibă un punctaj mediu de peste 60% la ultimele trei sesiuni de training.



3. Susţinerea examenului de promovare la nivelul următor de către cursant şi executarea unei sesiuni de training pe noul nivel disponibil.

Acesta este un caz de utilizare destul de comun între cursanți, implicând susținerea examenului de promovare la următorul nivel de dificultate, și odată luat acesta cu success, utilizatorul în cele mai multe cazuri execută o sesiune sau două training pe nould nivel de dificultate pentru a-și face o idee despre nouă dificultate la care acces acum.



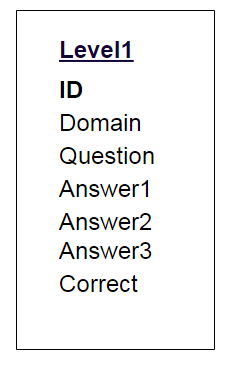
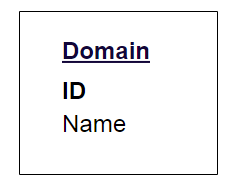
## Proiectarea bazei de date

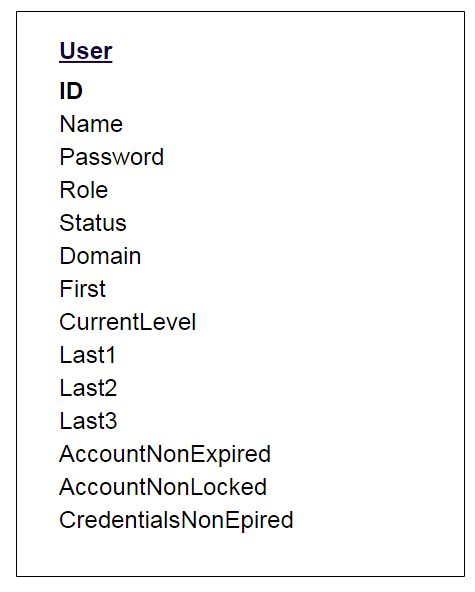
Pentru proiectarea bazei de date s-a folosit ca şi arhitectura un sistem descentralizat pentru stocarea întrebărilor şi datelor utilizatorilor, în sensul că nu s-au folosit chei secundare şi nu s-au legat tabelele la nivel relaţional, pentru evita dependinţele intre date şi blocarea execuţiei unor operaţii tranzacţionale pe baza legăturilor datelor respective cu alte tabele.

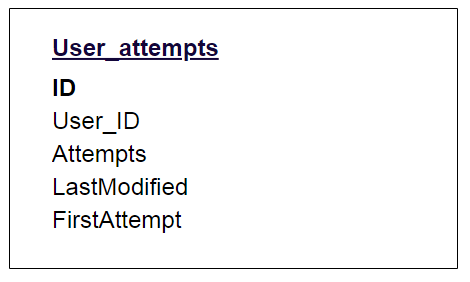
Astfel, s-a folosit un sistem de baze de date relaţional, urmând mai apoi ca tabelele respective să fie mapate în entităţi la nivel logic pentru implementarea algoritmilor prin framework-ul Spring, în limbajul Java.

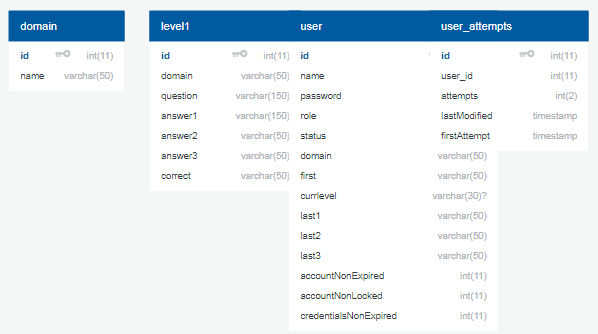
Baza de date a aplicaţiei conţine ca şi entităţi tabele pentru domenii, utilizatori, precum şi tabele pentru întrebări, acestea fiind structurate pe nivele, astfel, exista atâtea tabele de întrebări cât nivele de dificultate.

Ca si entitati esentiale ale schemei bazei de date, avem tabelele create pentru intrebari, tabelul domeniilor, acestea fiind importante pentru functiile si logica efectiva a platformei, si totdata avem tabela pentru utilizatori si o tabela aditionala pentru a verifica incercarile utilizatorului de logare, acestea doua avand rol principal in infrastructura aplicatiei.

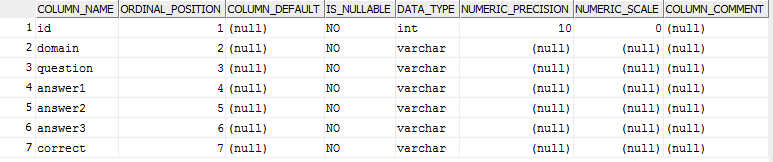




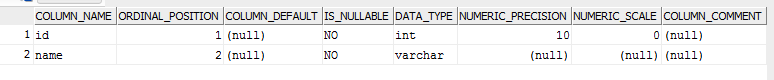




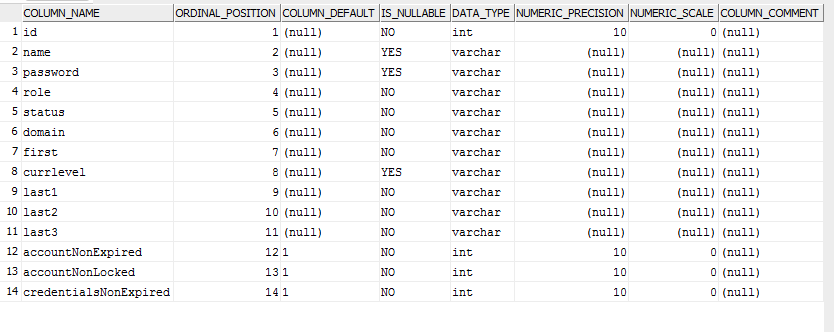
Tabelele unde sunt stocate întrebările sunt organizate pe nivele, şi structura coloanelor acestora consta într-un id unic, care se autoincrementeaza, domeniul întrebării, enunţul acesteia, precum şi cele trei variante de răspuns, şi totodată un câmp adiţional unde avem răspunsul corect, acesta trebuind să coincidă cu una din cele trei variante de răspuns.



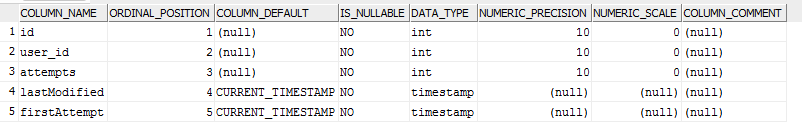
Domeniile sunt într-un tabel separat, ca şi entitate, întrucât au un scop vast atât în administrarea întrebărilor, cât şi conturilor utilizatorilor.



O altă entitate complexă a bazei de date este tabela de utilizatori, unde se ţin datele personale ale cursanţilor şi totodată statistici ale acestora specifice platformei, dintre care amintim că şi exemplu nivelul curent sau ultimele trei rezultate la sesiunile de training, fără de care nu s-ar putea face evaluarea pentru acceptanta susţinerii examenului de promovare la nivelul următor.

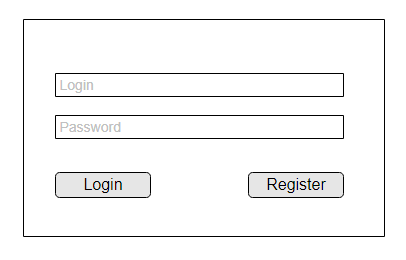


O tabelă adiţionala introdusă în schema bazei de date este tabela folosită pentru securitatea platformei, cu scopul principal de a ţine evidenţa încercărilor utilizatorului de a se loga, folosindu-se mai apoi o implementare în aplicaţia pentru a preveni logarea greşită a utilizatorului pentru un număr prea mare de încercări.

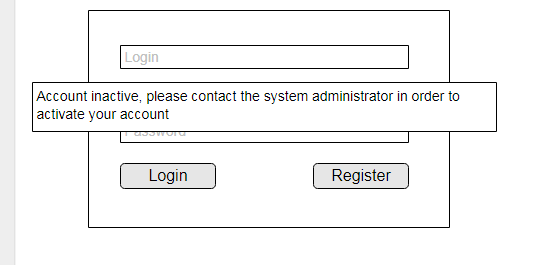


# Proiectarea interfeţei

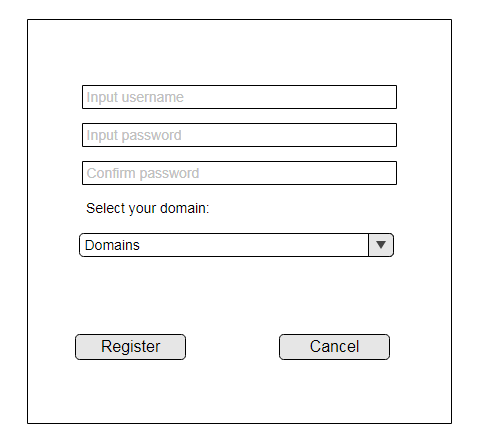
Proiectarea interfeţei începe în primul rând cu interfaţa cu care ia contact pentru prima oară utilizatorul, şi anume interfaţa de logare. Aceasta trebuie să fie simplă, minimalista, şi să conţină câmpurile pentru introducerea numelui de utilizator, parola acestuia şi butonul de login. Adiţional, aceasta conţine şi un buton pentru înregistrare, în cazul în care cursantul nu are încă un cont înregistrat pe această platformă.



Totodată, în cazul în care utilizatorul a depăşit numărul de încercări de logare disponibile, el va fi atenţionat printr-un banner informativ ca acum contul lui este blocat şi şi trebuie să vorbească cu un admin pentru a i se activa din nou.



Pentru interfaţa de înregistrare, o interfaţă prin care vor trece toţi utilizatorii standard, întrucât aceştia trebuie să îşi creeze cont pentru a putea accesa platforma, structura acestei interfeţe consta în aşezarea câmpurilor folosite pentru alegerea numelui de utilizator, parola, confirmarea acesteia , precum şi o listă cu domeniile disponibile pentru întrebări, mai apoi venind butoanele pentru register şi cancel.

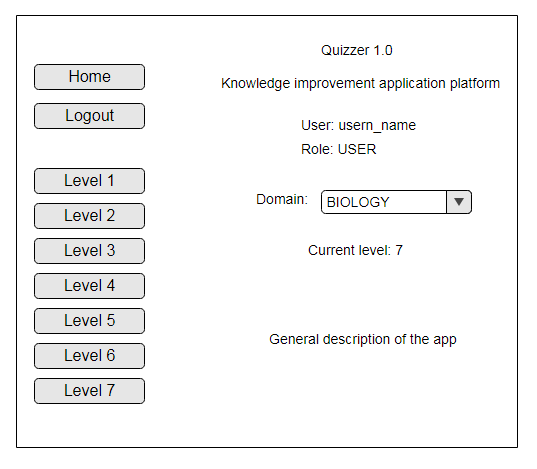


După apăsarea butonului pentru înregistrare, se va verifica dacă valorile introduse pentru câmpurile pentru parola şi respectiv confirmarea ei coincid, şi restul criteriilor sunt îndeplinite, cum ar fi numele de utilizator să nu fie deja existent.

Odată logat, utilizatorului va avea acces la platformă, unde în zona superioară din stânga sunt butoanele administrative, respectiv butonul pentru home, logout, eventual şi butonul pentru administraţie, în cazul în care utilizatorul este administratorul.

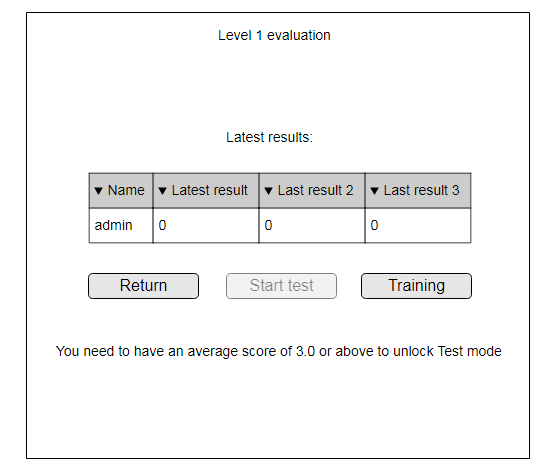
În zona inferioară din stânga avem prezente butoanele pentru navigarea pe diferite nivele de dificultate.

În zona din dreapta a landing-page-ului avem o descriere sumară a aplicaţiei şi totodată un combobox de unde utilizatorul îşi poate schimba domeniul pentru întrebări.



Interfaţa pentru fiecare nivel în partea va avea un tabel ce conţine numele utilizatorului şi rezultatele acestuia la ultimele trei sesiuni de training, precum şi butoanele pentru start test, în cazul în care criteriile sunt îndeplinite, butonul de training, pentru pornirea unei sesiuni de training, şi totodată butonul de return, pentru a întoarce utilizatorul la pagina de home.

În partea de jos a ecranului se va afişa şi o notă informativă în legătură cu nota medie la care trebuie să ajungă studentul pentru a putea susţine examenul pentru promovarea la următorul nivel de dificultate.



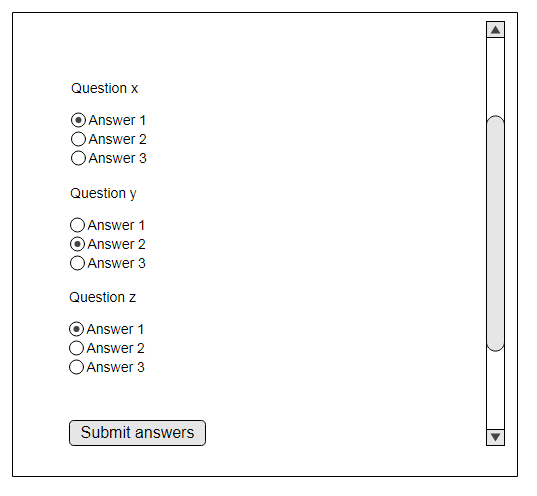
Interfaţa de utilizator responsabilă pentru teste în sne, fie ele cele de promovare la nivelul următor, sesiuni de training sau testarea iniţială a uşerului, trebuie să aibă o strucutra cât mai minimalista şi mai simplă.

Astfel, se asigura o experienţă cât mai lipsită de distrageri pentru cursant, scopul principal fiind acela de a-l ajuta să se concentreze cât mai eficient pe toată perioada susţinerii testului, indiferent de scopul acestuia.

Folosindu-se o astfel de structură, întrebările sunt organizate într-o listă, având enunţul în partea superioară şi mai apoi dedesubt cele trei variante posibile de răspuns, cu o singură variantă posibilă fiind corectă.

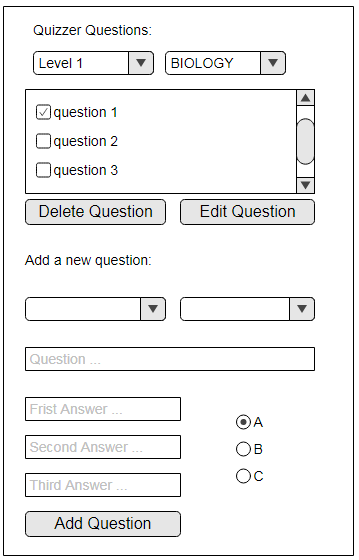
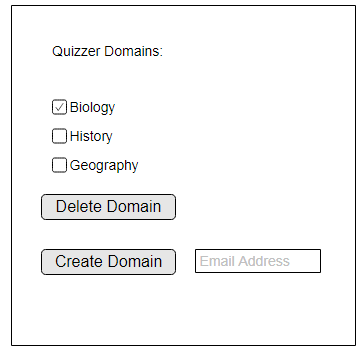
O astfel de aşezare simplificată face mai uşoară familiarizarea cursantului cu platforma şi totodată îl ajuta să se poată concentra cât mai uşor pe verificarea noţiunilor respective.

Deasemenea, după parcurgerea listei de întrebări, cursantul va găsi în partea inferioară a paginii butonul folosit pentru trimiterea răspunsului şi începerea procesului de validare al acestora, astfel ca acesta mai apoi primeşte într-o notă informativă pe pagina viitoare rezultatul în urma susţinerii testului şi eventuale remarci cu privire la performanţă acestuia.



Adiţional, pentru proiectarea interfeţei de utilizator folosită de către administratori, acesta va avea o zonă pentru adăugarea/ştergerea de domenii, precum şi o zonă similară, folosită pentru ştergere sau editare a întrebărilor, cu posibilitatea de a le filtra după domeniul acestora şi nivelul de dificultate.

Totodată, panel-ul administrativ va conţine şi o zonă desemnată pentru adăugarea de noi întrebări, unde adminul va specifică exact enunţul acesteia, precum şi variantele de răspuns, împreună cu cea corectă.



# Tehnologii

În ceea ce priveşte stack-ul de tehnologii folosit la aplicaţie, pentru proiectarea bazei de date s-a folosit MySQL ca şi sistem de gestionare a bazelor de date relaţionale, fiind o alternativă destul de lightweight la Oracle, cu o sintaxă similară.

Pentru proiectarea efectivă a aplicaţiei şi a arhitecturii serviciilor folosite de aceasta s-a folosit pentru programare limbajul Java, datorită scalabiliatatii oferite, suportului foarte bun pentru partea de multi-threading şi totodată datorită portabilităţii datorate JVM-ului, făcând astfel aplicaţia cross-platform compatible.

Ca şi framework-uri adiţionale implementării în Java, pentru partea de back-end a aplicaţiei Spring a fost nucleul să spunem aşa, folosindu-ne de facilităţile acestuia pentru a exercita cât mai clar şi eficient logica aplicaţiei la nivel programatic.

Spring este un framework redus ca dimensiuni, dar în acelaşi timp flexibil şi universal, orientat spre a facilita crearea de aplicații Java SE şi Java EE. Spring Framework urmăreşte să ofere infrastructura necesară pentru realizarea aplicațiilor Java, astfel încât programatorul să se poată concentra asupra dezvoltării efective de cod. Spring utilizează pe scară largă obiecte Java clasice (POJOs – Plain Old Java Objects).

La bază, o aplicație Java constă din obiecte care colaborează între ele pentru soluționarea unei probleme – obiectele dintr-o aplicație având dependențe între ele.

Platforma Java nu dispune de mijloacele necesare pentru a organiza obiectele, această sarcină revenind dezvoltatorilor şi arhitecților care au la dispoziție pattern-urile clasice (design patterns) pentru a organiza clasele şi obiectele. Totuşi, acestea trebuie implementate de către programator.

Facilitatea de inversiune a controlului (Inversion of Control – IoC) pusă la dispoziție de Spring Framework adresează această problemă – organizarea obiectelor – oferind mijloacele necesare pentru a permite componentelor disparate să fie asamblate într-o aplicație funcțională.

S-a folosit Hibernate ORM (Object-Relational Mapping) pentru maparea tabelelor din baza de date la nivel de entităţi în contextul Spring, pentru o translatare cât mai uşoară mai apoi în obiecte Java pentru o prelucrare cât mai uşoară a informaţiilor din baza de date.

Object / Relaţional Mapping (ORM) este o tehnică de programare ce face posibilă accesarea și manipularea obiectelor fără ca programatorii să fie interesați de sursa de date de unde provin aceste obiecte. Această tehnică a apărut din nevoia de a depăși diferențele de paradigmă dintre modelul orientat pe obiecte (susținut de limbajele de programare de nivel înalt actuale) și modelul relațional (utilizat de cele mai populare sisteme de gestiune a bazelor de date).

Limbajele de programare orientate pe obiecte reprezintă datele într-un graf interconectat de obiecte, pe când bazele de date relaționale folosesc un mod tabelar de reprezentare. Efortul de a conecta atributele claselor definite prin intermediul unui limbaj orientat pe obiecte cu câmpurile tabelelor din baza de date nu poate fi ignorat, iar scopul unui ORM este acela de a crea o relație naturală, transparentă, fiabilă și de durată între cele două modele. Această nepotrivire de paradigmă pare să nu își fi găsit încă o soluționare definitivă care să fie aprobată de toți actorii din industria IT, însă opinia generală este aceea că framework-urile ORM reprezintă un important pas înainte. Procesul automat de stocare a obiectelor într-o bază de date relațională folosind un framework ORM, constă în maparea obiectelor la tabelele corespunzătoare, asociereadintre ele fiind descrisă folosind metadata**.**

Hibernate reprezintă o soluție de tip ORM pentru asigurarea persistenței și a interogării eficiente a informațiilor stocate în baze de date relaționale, prin intermediul limbajului de programare Java. Este un produs open-source ce realizează maparea dintre tabelele bazei de date și clase Java, asigurând astfel interacțiunea dintre sistemeul de gestiune pentru baze de date și serverul de aplicații ce implementează logica aplicației.

Mecanismul prin care sunt reprezentate legăturile dintre baza de date și clasele Java este reprezentat de fișiere XML, astfel încât nu este necesară dezvoltarea de cod sursă pentru obținerea unei astfel de funcționalități. Astfel, în situația în care apare o schimbare la nivelul bazei de date sau la nivelul claselor Java, este suficient ca aceasta să fie realizată și la nivelul fișierului XML corespunzător pentru că legătura dintre entități să fie menținută. API-ul pus la dispoziție atât pentru asigurarea persistenței cât și pentru interogarea diferitelor informații din baza de date este flexibil și scalabil, fiind accesibil tuturor programatorilor, indiferent de nivelul de pregătire. Se oferă o abstractizare a operațiilor de la nivelul sistemului de gestiune pentru baze de date, astfel încât detaliile de implementare de la nivelul acestuia nu trebuie să fie cunoscute, interacțiunea făcându-se la nivelul obiectelor Java ce corespund unor concepte specifice afacerii sau culturii organizației respective. Pot fi modelate astfel legături oricât de complexe dintre tabelele entitate de la nivelul bazei de date. Mai mult, accesul la date se realizează optimizat, folosindu-se diferite strategii inteligente pentru interogarea informațiilor, menținând în același timp ușurința în utilizare. De asemenea, poate fi utilizat independent de un server de aplicații.

Pentru operaţiile la nivel de baze de date cum ar fi adăugarea, ştergerea sau modificarea datelor din tabele s-au folosit JPA Repositories ale Spring, astfel că fiecare tabel are corespondent un repository la nivel de Java ce permite efectuarea unor astfel de operaţiuni, şi totodată permite adăugarea unor operaţiuni noi, definite de ingineri, prin adăugarea noii metode folosite şi adnotarea acesteia cu query-ul pe bază de date care se vrea a fi efectuat.

JPA (Java Persistence API) reprezintă o colecție de clase (în limbajul de programare Java), oferind funcționalități legate de interogarea eficientă a informațiilor stocate în baze de date relaționale, mai ales în situația în care volumul acestora este considerabil. Această interfață de programare implementează un nivel de abstractizare, toate informațiile fiind stocate la nivel de obiecte, astfel încât programatorul nu este obligat să cunoască detaliile specifice unui sistem de gestiune al bazelor de date. Prin urmare, JPA realizează o punte între informațiile reprezentate relațional - așa cum sunt reținute într-o bază de date - și sub formă de instanțe ale unor clase, permițând un transfer facil între aceste moduri de organizare, implementând în același timp funcționalități de acces la date prin diferite mecanisme de interogare.

JPA reprezintă un mecanism prin care entitățile care își găsesc un corespondent în logica aplicației sunt convertite în date relaționale și viceversa. Acesta descrie modul în care pot fi descrise obiectele de tip POJO (Plain Oriented Java Object) precum și modul în care pot fi gestionate relațiile dintre acestea. Spre exemplu, un obiect de tip Entity reprezintă o entitate persistentă, fiind utilizat pentru stocarea și încărcarea informațiilor din baze de date relaționale. Acesta deține un singur atribut (opțional) - name, denumirea entității respective. De regulă, se folosește ca adnotare pentru a indica faptul că trebuie asigurată persistența clasei în cauză, la nivelul unei tabele în baza de date.

Pentru partea de front-end a aplicaţiei s-a folosit ca şi framework cu implementare în Java Vaadin, un framework relativ similar cu Swing, având că diferenţa esenţială aplicabilitatea acestuia la aplicaţiile pe o platformă Web, cum este cazul aici, astfel ca Vaadin translatează la runtime codul implementat în Java în cod JavaScript, pentru a putea fi interpretat de browser. Majoritatea componentelor folosite sunt standard, folosindu-se astfel un design simplist. Pentru organizarea efectivă a proiectului, pentru organizarea dependinţelor, plugin-urilor, librăriilor şi tuturor configurărilor necesare, precum şi pentru orchestrarea instalării acestora, a artefactelor şi a deploy-ului aplicaţiei s-a folosit ca şi tool principal Maven, acesta ajutând totodată şi la organizarea structurii directoarelor, având la rădăcina proiectului ca şi fişier esenţial fişierul pom.xml, bazându-se pe conceptul Project-Object-Model, aceasta fiind unitatea principală pe care se bazează Maven când executa goal-uri, acestea fiind acţiuni specifice, cum ar fi clean, install, deploy, etc.

Maven este un sistem de build și management al proiectelor, scris în Java. Face parte din proiectele găzduite de Apache Software Foundation. Funcționalitățile sale principale sunt descrierea procesului de build a softwareului și descrierea dependențelor acestuia. Un fișier XML descrie proiectul care urmează să fie build-uit, dependențele acestuia sau ale module și componente de care depinde, ordinea în care se execută build-ul, directoarele și plug-in-urile necesare. Maven descarcă dinamic bibliotecile Java şi plug-in-uri necesare, din unul sau mai multe repository-uri.

Pentru deploy-ul aplicaţiei s-a folosit ca şi webserver Tomcat, întrucât are o compatibilitate excelentă cu tehnologiile bazate pe JVM şi este foarte eficient pentru orchestrarea deploy-urilor.Apache Tomcat, de altfel cunoscut sub numele de „Tomcat” este un server web open-source, fiind dezvoltat și menținut tot de Apache Software Foundation. Este o implementare software pentru tehnologiile Java şi rulează pe Java Servlets și pagini JavaServer, cunoscute sub numele de JSP-uri.

Ca şi tool de versionare, folosit pentru a avea în primul rând siguranţa unui back-up al aplicaţiei, şi totodată posibilitatea de a face rollback la o versiune anterioară, în cazul în care au apărut probleme în cea curentă, s-a folosit Gât, întrucât este un tool la îndemână, open-source, cu multe resurse de informare asupra lui pe internet, care este totodată şi foarte versatil.

Un sistem de versionare este un mod de management al fișierelor care permite păstrarea istoricului tuturor modificărilor aduse fișierelor urmărite.

GitHub, serviciul folosit în acest caz, este un serviciu de hosting al proiectelor gât (un proiect gât este numit repository). Acesta permite păstrarea unei copii a proiectului online și vizualizarea tuturor fișierelor și a modificărilor aduse acestora.

# Implementarea aplicaţiei

Primul stagiu al implementării aplicaţiei a fost alegerea tehnologiilor folosite pentru dezvoltare, detaliate în capitolul Tehnologii, a tool-urilor adiţionale implementării, cum ar fi IDE-ul şi workbench-ul pentru baza de date.

După alegerea tehnologiilor şi instalarea/configurarea acestora, următorul pas a fost configurarea IDE-ului, în cazul acesta fiind folosit IntelliJ IDEA, un IDE bazat pe Java, pentru dezvoltare de aplicaţii complexe folosind tehnologiile Java. Pentru accesul şi modificările adiţionale asupra bazei de date, ca şi tool a fost folosit în faza incipientă mysqld, tool-ul default cu care vine baza de date MySQL, ulterior, pentru opratii mai complexe fiind ales SQL Developer, un tool dezvoltat de Oracle, cu un plugin pentru MySQL.

Ca un prim pas în începerea dezvoltării aplicaţiei avem proiectarea bazei de date, întrucât aceasta constituia fundaţia fără de care nu s-ar fi putut continua dezvoltarea tuturor celorlalte module ale aplicaţiei.

După hotărârea asupra arhitecturii bazei de date, asupra tabelelor care vor fi folosite, s-a continuat cu adauagrea efectivă a acestora în baza de date. Detalii adiţionale cu privire la structura tabelelor şi detalii specifice se pot găsi la capitolul pentru Proiectarea bazei de date.

Următorul pas, după stabilirea bazei de date, a fost reprezentat de construirea scheletului aplicaţiei. Acesta lucru a fost facilitat de folosirea framework-ului SpringBoot, astfel ca folosind tehnologiile specificea acestuia, am adăugat în fişierul specific Maven, şi anume pom.xml dependinţele necesare pentru rularea aplicaţiei, diferite librarii şi plugin-uri necesare pentru dezvoltare, şi eventuale guidelines cu privire la artefactele rezultatele în urma compilării şi aşa mai departe. Totodată, în acest stagiu s-au adăugat configurările necesare pentru framework-ul de Spring, prin fişierul de application.properties .

După stabilirea unei structuri incipiente a aplicaţiei, următorul pas a fost construirea fundaţiilor aplicaţiei pentru parte de back-end.

Acestea au fost reprezentate în prima fază de maparea tabelelor din baza de date ca şi entităţi în contextul Spring, create la paşii anteriori la nivel de obiecte Java, pentru a putea exercita la nivel programatic operaţiuni şi business logic asupra lor cu uşurinţă.

import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.EnumType;  
import javax.persistence.Enumerated;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
  
*/\*\*  
 \* Created by alexandru.sandu  
 \*/*@Entity // This tells Hibernate to make a table out of this class  
public class User {  
 public enum Role {  
 *ADMIN*("ADMIN"), *USER*("USER");  
 private final String value;  
 private Role(String value) {   
 this.value = value;   
 }  
 public String toString() {  
 return this.value;  
 }  
 }  
  
 @Id  
 @GeneratedValue(strategy=GenerationType.*AUTO*)  
 private Long id;  
  
 private String name;  
 private String password;  
   
 @Column(name = "role")  
 @Enumerated(EnumType.*STRING*)  
 private Role role;

Totodată, acesta este pasul în care au fost create şi repositoarele bazate pe JPA, facilitându-se astfel executarea operaţiunilor asupra bazei de date. Felul în care acestea au fost folosite în arhitectura aplicaţiei a constat în adăugarea unor metode şi funcţii la nivelul lor, acestea fiind programate să corespundă unor query-uri parametrizate, astfel că de fiecare dată când o metodă era chemată din limbajul Java, aceasta de fapt în spate apela aceste repository-uri pentru execuţia unor query-uri stabilite de dezvoltatori asupra bazei de date.

import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Repository;  
  
  
@Repository  
public interface UserRepository extends JpaRepository<User, Long> {  
 @Query(value = "select u from User u where u.name = :name and u.password = :password ")  
 User findByNameAndPassword(@Param("name") String name, @Param("password")String password);  
  
 User findByName(String userName);  
}

Astfel, am putea spune că maparea tabelelor la nivel de entităţi este o mapare informaţională, iar maparea query-urilor pe bază de date în metode ale repositoarelor este o mapare funcţională.

Tot în acest stagiu au fost create şi serviciile aplicaţiei, aceasta acţionând în arhitectura aplicaţiei ca nişte providere de informaţii, ele reprezentând de fapt o grupare asupra unor funcţii adiţionale, un layer adiţional de abstractizare am putea spune asupra repositoarelor, astfel ca, spre exemplu, toate întrebările, indiferent de nivelul de dificultate al acestora sau de domeniul de care aparţin, pot fi accesate sau modificate drect dintr-un sngur nivel, în cazul acesta, level service, creat special pentru acest scop.

@Service  
public class UserService {  
   
 private static final Logger *logger* = Logger.*getLogger*(UserService.class.getName());  
   
 @Autowired  
 private UserRepository userRepository;

public User findByName(String userName){  
 *logger*.info("Entered method findByName(name= [" + userName + "])");  
 Assert.*hasText*(userName, "The name must must not be empty and must contain at least one non-whitespace character!");  
 return userRepository.findByName(userName);  
}  
  
public User findByNameAndPassword(String name, String password){  
 *logger*.info("Entered method findByNameAndPassword(name= [" + name + "], password= [" + password + "])");  
 Assert.*hasText*(name, "The name must must not be empty and must contain at least one non-whitespace character!");  
 Assert.*hasText*(password, "The password must must not be empty and must contain at least one non-whitespace character!");  
 return userRepository.findByNameAndPassword(name, password);  
}  
  
public User saveUser(User user) {  
 *logger*.info("Entered method save(User user)");  
 Assert.*notNull*(user, "The user must not be null");  
 *logger*.info("Saving user: [" + user.toString() + "])");  
  
 User savedUser = userRepository.save(user);  
 *logger*.info("Saved user is " + savedUser.toString());  
 return savedUser;  
}

Designul aplicaţiei, bazat pe acest tip de back-end implica un EntiyManager care va crea nişte bean-uri (beans - unitatea structurală a contextului Spring) pe baza entităţilor definite programatic de către dezvoltator, acesta putând fi apoi iniţializate şi totodată accesate de oriunde din contextul aplicaţiei, folosindu-se conceptul injecţiei de dependinţe (dependency injection). Astfel, printr-o adnotare de tipul @Autowired, un repository declarat în contextul aplicaţiei este legat (wired) automat de entitatea corespondenţa, fără a mai fi nevoie de adăugarea unor configuraţii adiţionale.

Folosindu-se o arhitectură a aplicaţiei bazate pe obiecte, multi-threading, operaţiuni asincrone, şi context, şi în primul rând o aplicaţie bazate pe interfaţă cu utilizatorul, s-a folosit ca şi concept principal conceptul de inversare a controlul (IOC- Inversion of Control).

Printr-un astfel de concept, spre deosebire de o aplicaţie dezvoltată în C spre exemplu, unde dezvoltatorul foloseşte ca şi tehinca de programare programarea imperativă, spre deosebire de cea orientata pe obiecte, deci aplicaţia în sine controlează flow-ul aplicaţiei, aici flow-ul aplicaţiei este dictat de utilizator, în funcţie de ce doreşte acesta să activeze. Acestuia nu i se poate dicta ce comportament să aibă, ce flow-uri să activeze, ci decide singur asuprea aceastor lucruri, aplicaţia doar punându-i la dispoziţie diferitele alegeri posibile pe care acesta le poate face.

Odată creat back-endul aplicaţiei, următorul pas a fost implemetarea modului pentru front-end. Aici, folosind ca şi framework Vaadin, bazat şi acesta tot pe Java, s-a putut realiza o integrare foarte facila cu celelalte componente, bazându-se toată implementarea aplicaţiei pe tehnocologii cu aceeaşi bază, în cazul de faţă, limbajul de programare Java.

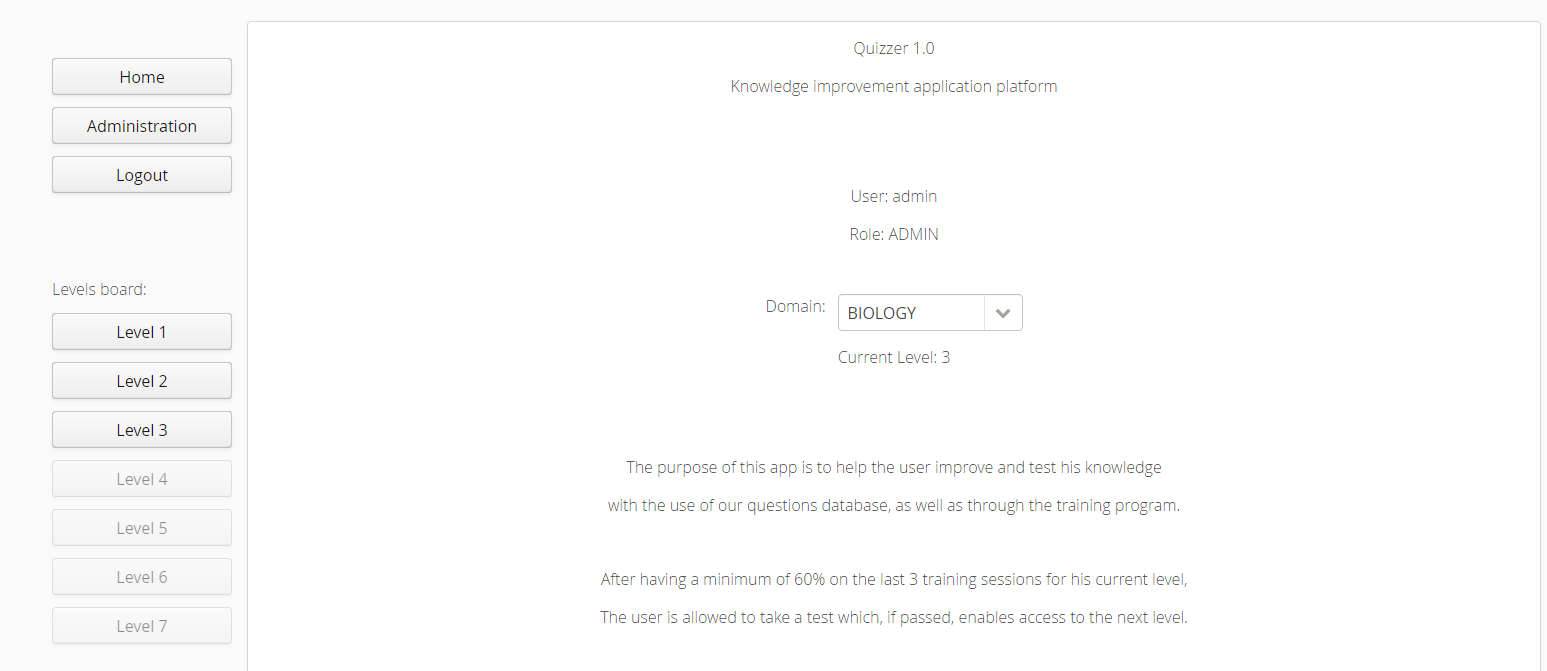
Fiind un framework realtiv similar cu Swing, totuşi optimizat pentru web development, astfel ca acesta transforma codul Java în cod Javascript pentru a putea fi interpretată de browser, nemaifiind astfel nevoie de vrei un limbaj adițional pentru a putea exercita business logic-ul aplicației.

La baza interfeței de utilizator stă un element de tipul UI, astfel ca toate elementele interfeței de utilizator sunt coordonate de către acesta, așadar acesta acționează ca un middle tier între front-end și back-end, putând unei astfel exercita logica prin intermediul acestei componente.

import com.vaadin.navigator.Navigator;  
import com.vaadin.navigator.ViewChangeListener;  
import com.vaadin.server.VaadinRequest;  
import com.vaadin.server.VaadinSession;   
import com.vaadin.shared.ui.ui.Transport;  
import com.vaadin.spring.annotation.SpringUI;  
import com.vaadin.spring.navigator.SpringViewProvider;  
  
*/\*\*  
 \* Created by alexandru.sandu  
 \*/*@SpringUI  
@Theme("valo")  
@Push(transport = Transport.*WEBSOCKET*, value = PushMode.*AUTOMATIC*)  
@PreserveOnRefresh  
@Widgetset("com.training.quizzer.ui.quizzerWidgetSet")  
public class quizzerUI extends UI {  
 private static final Logger *logger* = Logger.*getLogger*(quizzerUI.class.getName());

Aceasta componentă sta la baza arhitecturii fron-end-ului, întrucât ea este responsabilă de routarea să spunem aşa view-urilor, de implementarea unui navigator, care face posibilă mutarea între diferite views ale aplicaţiei.

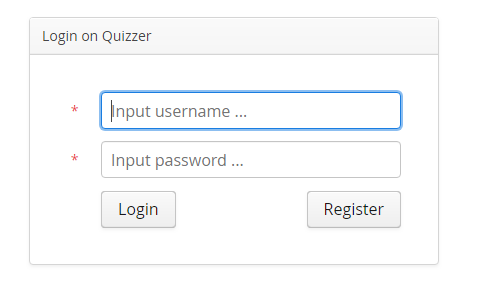
Structura de bază a interfeței de utilizator, odată ce acesta este logat pe platformă, este constituită din panel în zona din dreapta a paginii, Unde este conținutul efectiv al meniurilor și al informațiilor aferente platformei, iar în partea din stânga a paginii, în partea superioară, avem prezente butoanele  Administrative ale sesiunii, în speță butonul pentru home și butonul pentru delogare, tot aici fiind prezent și butonul pentru  deschiderea panelului de administrare, în cazul în care utilizatorul are rolul de administrator al platformei. în partea inferioară avem butoanele folosite pentru navigarea între diferitele nivele de dificultate ale aplicației,  Deci navigarea către sesiunile și examenele specifice fiecărui nivel de dificultate.



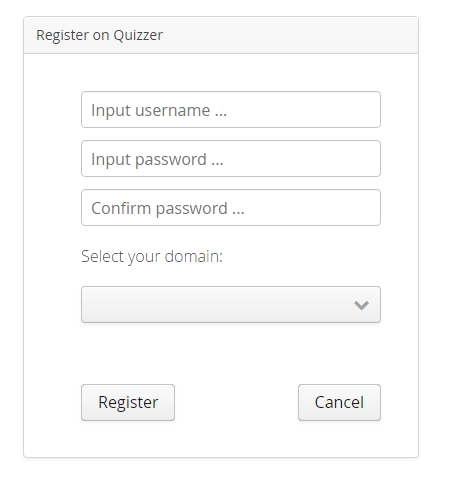
Implementarea paginilor consta de fapt în implementarea unor view-uri, având la baza Vaadin, care sunt mai apoi introduse şi ele în contextul Spring, ca şi SpringViews, având un nume distinct fiecare dintre ele. În baza acestui nume distinct se face mai apoi redirecţionarea către ele prin intermediul navigatorului, care este responsabil de modificarea contentului din panel-ul principal, din partea dreaptă a interfeţei, butoanele de navigare din partea stângă a interfeţei ramând în aceeaşi stare, totodată din ele apărând posibilitatea de trigger a evenimentelor de navigare.

@SpringView(name = RegisterView.*VIEW\_NAME*)  
public class RegisterView extends CustomComponent implements View {  
 public static final String *VIEW\_NAME* = "registerView";  
 public User user;  
 @Autowired  
 UserService userService;

Pentru prima fază a proiectării frontend-ului, eforturile s-au concentrat asupra creării interfeței pentru logare. Astfel, o dată cu generarea interfeței care conține un câmp pentru introducerea numelui de utilizator, un câmp pentru introducerea parolei, apoi butonul pentru logare și butonul pentru înregistrare, în momentul logării, se folosește un Binder, astfel că contul utilizatorului este legat de sesiunea aplicației pe care acesta tocmai a deschis-o. Astfel, se asigură o unitate a experienței pentru fiecare utilizator și totodată o măsură de siguranță pentru conectarea utilizatorului la secțiunea corespunzătoare.

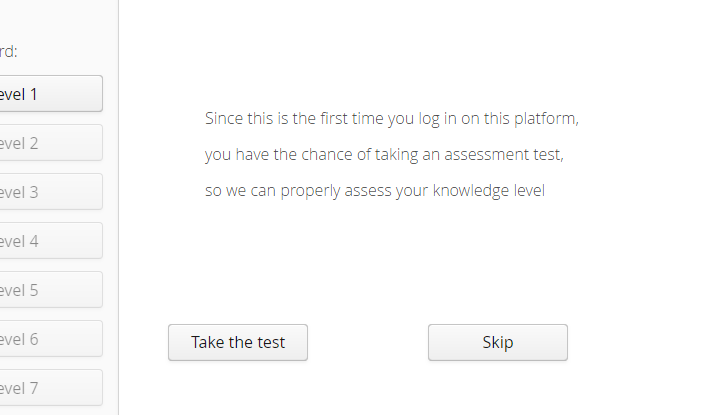


Mai apoi, odată ce aceasta a fost creată, următorul pas a fost reprezentat de crearea interfeței pentru înregistrarea utilizatorului nou în cazul în care utilizatorul contact cu platforma pentru prima oară. Astfel, interfața de utilizator a fost concepută ca având un câmp pentru introducerea numelui de utilizator, totodată verificarea acestuia pentru unicitate la nivel de bază de date, mai apoi câmpurile pentru introducerea parolei și totodată pentru confirmarea acesteia,urmând ca mai apoi următorul aliment să fie o listă derulantă cu domeniile disponibile, aşa că fiecare utilizator trebuie să își aleagă un domeniu de activitate în care acesta va fi testat.  mai apoi în zona inferioară a paginii sunt prezente butoanele pentru confirmarea înregistrării, și butonul pentru cancel, în cazul în care utilizatorul  nu vrea să salveze modificările.



După apăsarea butonului pentru înregistrare, aplicația face o interogare la baza de date pentru a verifica unicitatea a numelui de utilizator. În cazul în care acest criteriu este îndeplinit, aplicația verifică mai apoi dacă Câmpurile pentru introducerea parolei și respectiv verificarea acesteia coincid, urmând ca mai apoi, ultima verificare înaintea salvării înregistrări pentru utilizatorul respectiv  să fie selecția corectă a unui domeniu.

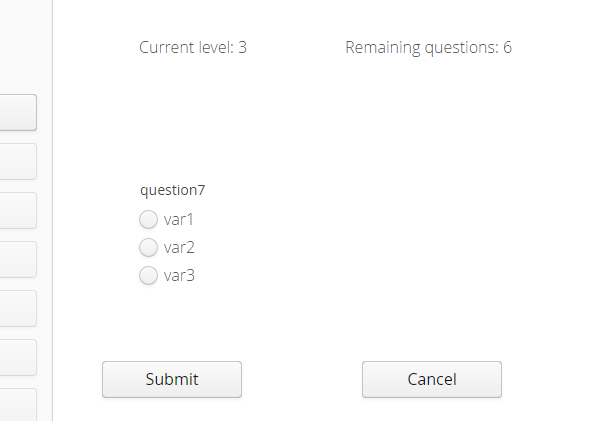
O dată înregistrat, utilizatorul se poate loga pe platformă. în cazul în care utilizatorul este pentru prima oară logat, acesta va fi informat că are posibilitatea de a susține un examen pentru o evaluare inițială, acesta având posibilitatea atât de a susține examenul, cât și de a nu face acest lucru, cazul în care acesta va începe automat cu nivelul 1 de dificultate.



Această evaluare inițială are ca scop principal stabilirea nivelului utilizatorului de dinainte ca acesta să fi folosit platforma putîndu-i-se  astfel atribui un nivel mai apropiat de cel deja existent.Motivul pentru care este recomandat o astfel de evaluare este acela de a evita într-o oarecare măsură plictisirea utilizatorului sau forțarea acestuia în a parcurge nivele sub nivelul lui curent de dificultate. Astfel, Se asigură o experiență cât se poate de unitară asupra platformei, indiferent de gradul de cunoștințe pe care le posedă cursantul.

Algoritmul din spatele acestei testări inițiale constituie în primul rând plecarea de la nivelul 1 de dificultate ca și nivel inițial, atenția fiind îndreptată asupra răspunsurilor pe care le dă cursantul în urma întrebărilor primite.

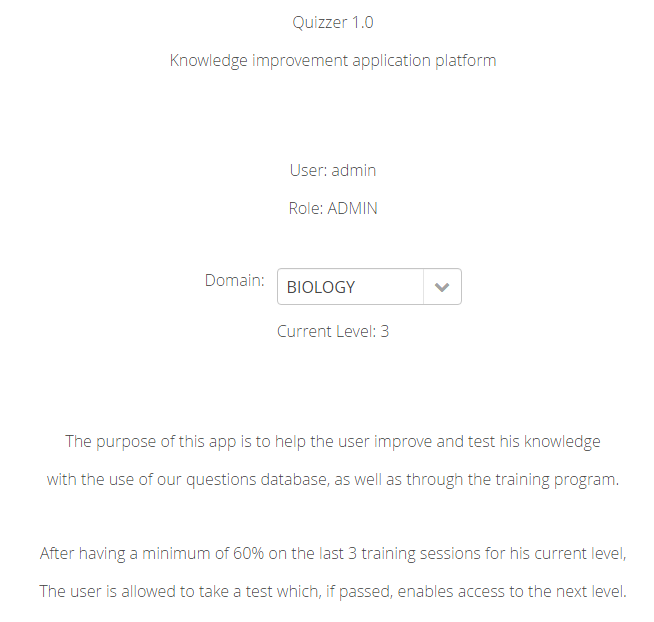
Acesta nu are vizibilitate, și nu poate răspunde decât la o singură întrebare pentru fiecare pas, astfel că următoarea întrebare pe care o va primi va fi cu un nivel superior sau inferior de dificultate față de cea curentă, în funcție de răspunsul acestuia. Cu alte cuvinte, dacă cursantul răspunde corect la întrebarea curentă, următoarea întrebare vă fi cu un nivel de dificultate incrementat, iar în cazul în care cursantul răspunde cu un răspuns greșit, atunci următoarea întrebare vă veni cu un nivel de dificultate decrementat.



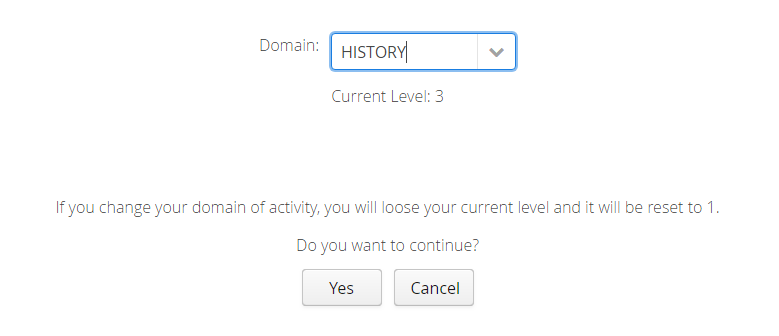
Pe toată durata testului, se ține minte nivelul curent al cursantului, în funcție de nivelul de dificultate al întrebării curente. Astfel, la finalul evaluării inițiale, platforma are un nivel mediu al răspunsurilor cursantului în funcție de rezultatele acestuia și de nivelul curent al ultimei întrebări, care este dependent de  de răspunsurile utilizatorului pe toată durata execuției testului, astfel că are locul evaluare cât mai completă a cunoștințelor cursantului și o evaluare a nivelului acestuia cât se poate de precisă putând fii categorisit într-unul din nivelele de dificultate ale platformei.

Odată cu finalizarea evaluării inițiale, utilizatorul are deblocate toate nivelele de dificultate inferioare nivelului curent. Astfel, Spre exemplu, dacă utilizatorul are evaluat nivelul de dificultate ca fiind corespunzător nivelului de dificultate 3 al platformei, atunci acestuia i se deblochează atât nivelul curent în cazul acesta 3, cât și nivelele inferioare, în speță nivelele 2 și 1. Folosind o astfel de logică, i se permite utilizatorului să poată susține examene și sesiuni de training și de la nivelele inferioare, chiar dacă acesta nu a avut contact cu ele.

Odată logat pe platformă, utilizatorului îi este prezentată pagina de start a aplicației, care conține informații generale despre platformă, despre contul utilizatorului și rolul acestuia, informații generale despre regulile de utilizare a acesteia, precum și o listă derulantă din care utilizatorul își poate schimba domeniul de activitate în care acesta dorește să fie testat.

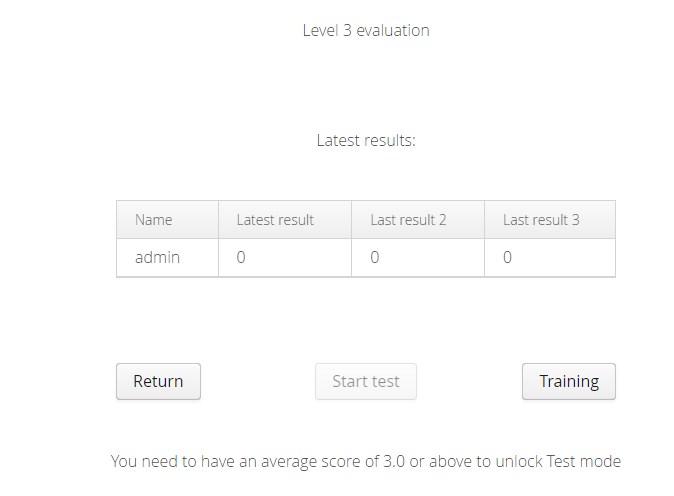


în cazul în care utilizatorul dorește să schimbe domeniul întrebărilor, cursantul este avertizat că în momentul în care acesta dorește să schimbe domeniul, el își va pierde nivelul curent la care acesta a ajuns în momentul de față, și va începe evaluarea pe domeniul selectatîncepând de la nivelul de dificultate 1.



Odată navigat pe unul dintre nivelele de dificultate deblocate pentru utilizator, ne este prezentată interfața specifică nivelelor de dificultate care conține un notă informativă cu nivelul curent selectat, un tabel cu numele utilizatorului și rezultatele acestuia la ultimele trei sesiuni de antrenament, întrucât acestea sunt esențiale pentru a debloca oportunitatea de a susține examenul de promovare către nivelul următor.

Algoritmul aplicației presupune că un utilizator nu poate susține examenul pentru promovare la următorul nivel până când nu a îndeplinit un criteriu de bază, și anume, realizarea unui punctaj de peste 60% la ultimele trei sesiuni de training.Această metodă, acest mecanism de asigurare a consistenței cunoștințelor utilizatorului asigura o pregătire cât mai completă și mai complexă, în urma căreia prin confirmare, se poate susține examenul pentru promovare la următorul nivel de dificultate.

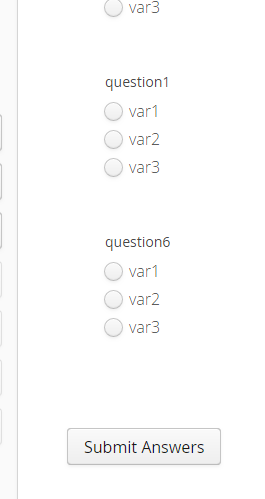


Sesiunile de training sunt una dintre bazele platformei pentru evaluarea de cunoștințe, întrucât utilizatorul pe trece o foarte mare parte din timpul alocat pe platformă, în aceste stadii.în urma execuției unei sesiuni de antrenament, rezultatul acesteia, indiferent de cuantum, este salvat în înregistrările bazei de date referitoare la utilizatorul respectiv, făcând dus felul lor în prospectare a statisticilor pe care acesta le vede la nivelul de interfață de utilizator.

Examenele pentru promovarea la următorul nivel de dificultate sunt parte esențială a platformei în sine, ele fiind de fapt modalitatea prin care cursantul poate verifica și cuantifica progresul pe care acesta l-a făcut până în momentul de față, și totodată este mijlocul principal de evaluare pe care această platformă de evaluare a cunoștințelor îl oferă.

Aceste teste, spre deosebire de sesiunile de antrenament, conțin atât întrebări din nivelul de dificultate curent, cât și o serie de întrebări aleatoare specifice nivelelor de dificultate anterioare. Astfel, prin acest mecanism de asigurare a  persistenței cunoștințelor, se asigură o evoluție cât mai precisă, și o evaluare cât mai completă și corectă a cursantului, întrucât acesta este nevoit să dovedească că posedă un continuare cunoștințele de la nivelele inferioare, criteriu esențial în promovarea la următorul nivel de dificultate.

 Interfața de utilizator responsabilă pentru susținerea efectivă a examenelor a fost gândită să fie cât mai simplistă și cât mai minimalistă scopul unui astfel de design fiind acela de a permite utilizatorului un mediu cât mai lipsit de distrageri, acesta putându-se astfel concentrară cât se poate de eficient și de complet asupra întrebărilor din textul pe care acesta îl efectuează.



Algoritmul aplicației din spatele execuției testelor implică utilizarea serviciilor atât celui pentru utilizator cât și celui pentru nivele. Pe baza Serviciul de utilizator se fac legăturile cu sesiunea curentă și cu rezultatele din baza de date asupra utilizatorului, în speță nivelul curent, etc. Iar pe baza serviciului de nivele se face providing-ul cu întrebări, în funcție de nivelele necesare, mai precis nivelul curent și nivelele inferioare acestuia.

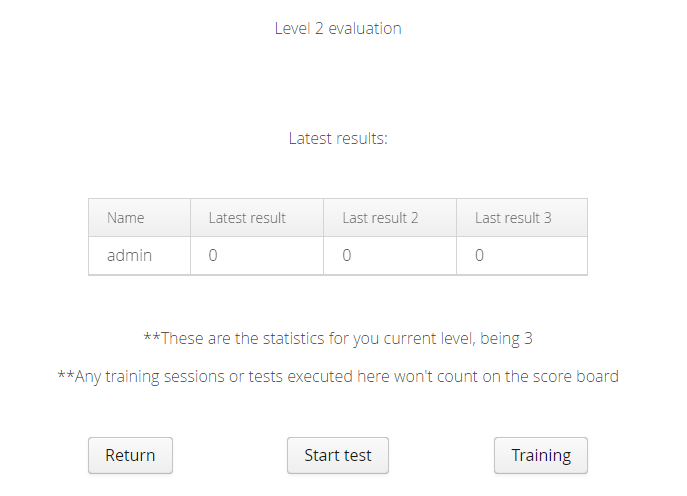
Totodată o altă parte foarte importantă algoritmice este reprezentată de mape, o structură de date de tipul cheie evaluare, care ne permiteți să mapăm întrebările și selecția utilizatorului ca fiind răspunsul corect. în baza acestei selecții, și a întrebării corespunzătoare acesteia, se face compararea între răspunsul utilizatorului și răspunsul știind că ei fiind cel corect pentru întrebarea respectivă, pentru validarea punctajului la întrebarea în cauză.

@Autowired  
private LevelService levelService;  
  
private Map<Question, String> answersMap;  
private VerticalLayout questionLayout;  
private Button submitButton;  
private double userScore;  
private VerticalLayout resultLayout;  
private Button returnButton;  
private VerticalLayout mainLayout;

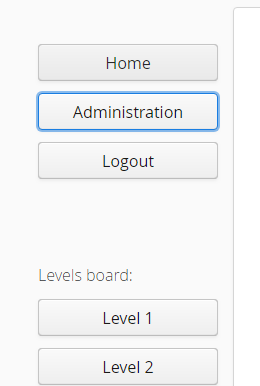
Validările au loc în momentul în care utilizatorul apasă pe butonul pentru trimiterea rezultatelor, implicit astfel confirmare a finalizării testului.Fiecare nivel are un așa a zis desired score, bazat pe numărul de întrebări pe care acesta îl conține, astfel ca punctajul necesar promovării în urma execuţiei unui test diferă de la nivel la nivel, în funcție de numărul de întrebări. Acest desired score este calculat în momentul afișării întrebărilor pentru testul respectiv, și este luat în calcul pentru Evaluarea rezultatului la final, după ce utilizatorul apasă pe butonul pentru trimiterea răspunsurilor.

private void addSubmitButtonListener() {  
 submitButton.addClickListener(clickEvent -> {  
 userScore = 0;  
 double desiredScore = getDesiredScore();  
 answersMap.keySet().forEach(key -> {  
 if(answersMap.get(key).equals(key.getCorrect())) userScore++;  
 });  
  
 if (userScore >= desiredScore){  
 showResult("success");  
 returnButton.addClickListener(event -> getUI().getNavigator().navigateTo(Level2View.*VIEW\_NAME*));  
 } else {  
 showResult("fail");  
 returnButton.addClickListener(event -> getUI().getNavigator().navigateTo(Level1View.*VIEW\_NAME*));  
 }  
 });  
}

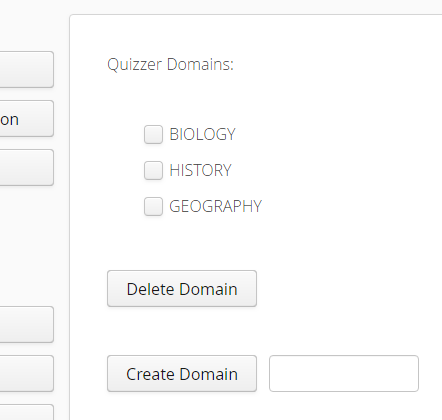
În cazul în care utilizatorul nu se află pe nivelul curent și este în panelul specificul unui alt nivel, inferior, acesta va fi atenționat că rezultatele în urmă a execuției sesiunilor atât de antrenament cât și cele de promovare din nivelul respectiv nu se vor lua în calcul, și mai ales nu se vor lua în calcul pentru nivelul curent la care acesta activează. Aceste nivele sunt în momentul de față pentru cursant acolo doar cu caracter informativ.



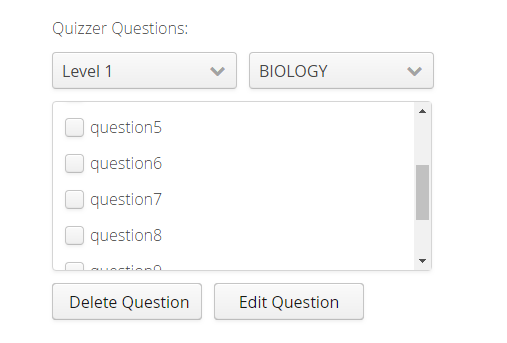
Pe lângă interfeței de utilizator specifice utilizatorilor standard, în speță cursanților, mai avem o interfață de utilizator specifică administratorilor, și anume panelul pentru administrare, prin care se poate ajunge printr un buton amplasat în zona butoanelor pentru home și pentru delogare, buton care nu este vizibil utilizatorilor standard ci doar administratorilor.



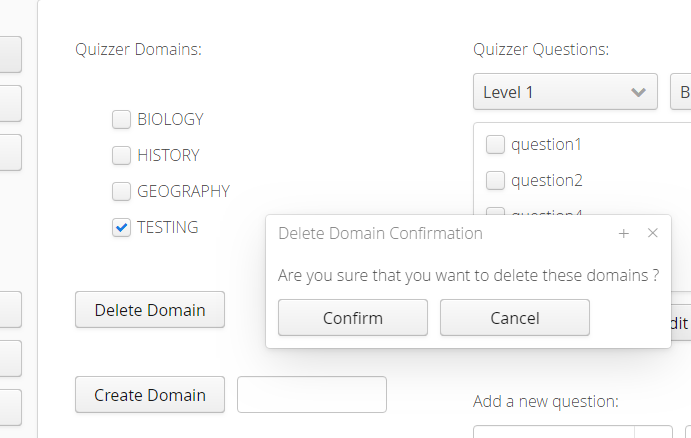
Acest panou administrativ conține trei zone importante, unde dintre ele fiind zona pentru administrarea domeniilor platformei, unde se pot vedea domeniile existente la momentul respectiv pe platformă, și totodată administratorii au abilitatea de a adăuga sau de a șterge domenii de pe platformă.



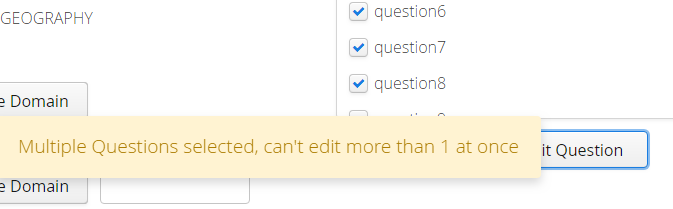
O altă zonă importantă a panelului administrativ este cea corespunzătoare administrării întrebărilor existente pe platformă. Acestea se pot vedea printr o modalitate de filtrare bazată pe nivelul acestora și pe domeniile corespunzătoare lor, cu posibilitatea de a fi șterse, respectiv editate.



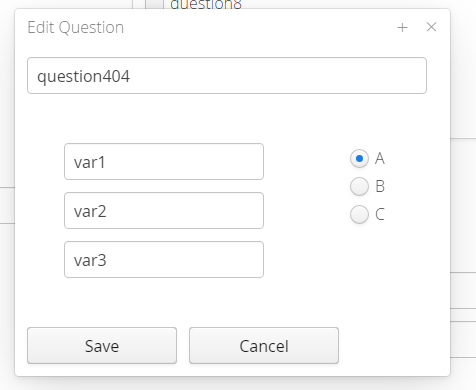
Atât în cazul ștergerii unui domeniu cât și a întrebărilor utilizatorul este informat să confirme acțiunea pe care urmează să o desfășoare, pentru a preveni ștergerea accidentală a datelor.



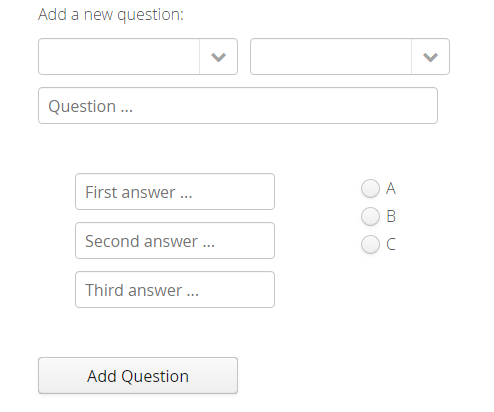
Pentru editarea întrebărilor administratorul nu poate selecta decât o singură întrebare. În cazul în care acesta a selectat multiple întrebări, sau nu a selectat nici una, fiind informat de utilizare a defectuoasă a editării întrebărilor.



După selectarea unei singure întrebări și apăsarea butonului pentru editare, administratorului i se va deschide o fereastră adițională, de unde acesta poate modifica oricare din câmpurile întrebării, în speță enunțul acesteia sau vreuna dintre cele trei variante de răspuns.



Cea de a treia zonă importantă din panelul administrativ este reprezentată de zona de unde administratorul poate adăuga noi întrebări pe platformă această zonă conține toate câmpurile specifice unei întrebări, enunțul acesteia, cele trei variante posibile de răspuns, precum și o zonă de selecție, de unde administratorul poate specifica care din cele trei răspunsuri este cel corect, și de asemenea administratorul este prezentat cu două liste derulante de unde acesta poate selecta domeniul, respectiv nivelul de dificultate ale întrebării nou create.



Prin panelul administrativ, administratorii sunt diferențiați de utilizatorii standard, în speță cursanții,  aceștia vând în plus abilitatea de a controla conținutul efectiv al platformei, putând adăuga, șterge atât domenii de activitate cât și întrebările aferente acestora.

# Concluzii

Ca și concluzii, putem observa că și lucru adus nou în acest domeniu, față de restul aplicațiilor, testarea inițială, având ca scop principal garanția unei experiențe similare pentru toți cursanții, indiferent de nivelul acestora de pregătire inițială, înainte de luarea contactului cu platforma.

Un alt lucru esențial pe care îl aduci această platformă este simplitatea și minimalismul arhitecturii interfeței grafice, astfel că utilizatorul se poate concentra eficient și complet asupra examenului pe care acesta îl susține în momentul respectiv, fără distrageri.

Un alt mecanism eficient al platformei pentru asigurarea consistenței și a persistenței cunoștințelor cursanților este reprezentat de sesiunile de antrenament, prin care utilizatorul trebuie să treacă înainte de a putea susține examenul pentru promovarea la următorul nivel de dificultate. Tot aici intră și cuprinsul examenului pentru promovare, și anume, prezența în conținut a întrebărilor de la nivelele inferioare nivelului curent, astfel asigurând-se o plajă de cunoștințe cât mai completă a cursanților.

O remarcă a abordării diferite față de majoritatea platformelor similare este constituită de concentrarea asupra testelor și evaluărilor susținute, și totodată de lipsa materialelor ajutătoare pentru pregătirea cursanților virgulă întrucât trăim într-un mediu în care informația este accesibilă oricui, astfel nemaifiind necesară implicarea platformei în pregătirea utilizatorilor.

în concluzie, această platformă destinată evaluării cunoștințelor iese în evidență prin concentrarea asupra evaluărilor utilizatorilor a și totodată prin algoritmică folosită pentru a asigura are a nivelului de cunoștințe satisfăcător al cursanților.

# Bibliografie

http://master.cedcsv.ro/cedc/infos/informatica/platforme\_de\_evaluare.pdf

http://spring.io/projects/spring-framework

https://spring.io/projects/spring-boot

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

https://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial

https://vaadin.com/docs

https://maven.apache.org/

http://blog.infoeducaţie.ro/tutorial/2015/04/14/github-101.html

http://aipi2015.andreirosucojocaru.ro/laboratoare/laborator02

https://moqups.com/