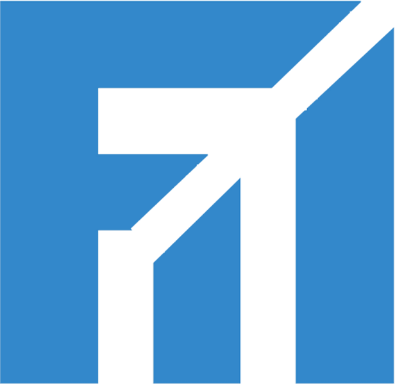
**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Knowledge Base**

**propusă de**

***Cezar-Iustinian Iacob***

**Sesiunea**: *iulie, 2021*

**Coordonator științific**

**Lect. Dr. Cristian Frasinaru**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

**Knowledge Base**

**Cezar-Iustinian Iacob**

**Sesiunea**: *iulie, 2021*

**Coordonator științific**

***Lect. Dr. Cristian Frasinaru***

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele Lect. Dr. FRĂSINARU CRISTIAN

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a)  **IACOB CEZAR-IUSTINIAN** domiciliul în România, **Jud. Iași Municipiu PAȘCANI** născut(ă) la data de **11 septembrie 1999**, identificat prin CNP **1990911225912**, absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, **Facultatea de INFORMATICĂ** specializarea **INFORMATICĂ**, promoția 2021, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: **Knowledge Base** elaborată sub îndrumarea dl. / d-na **Lect. Dr. CRISTIAN FRĂSINARU**, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime. De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop. Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

**DECLARAȚIE DE CONSIMȚĂMÂNT**

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul **Knowledge Base**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, data

Absolvent **CEZAR-IUSTINIAN IACOB**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

# Cuprins

[Introducere 4](#_Toc71556741)

[I. Specificații funcționale 9](#_Toc71556742)

[II. Arhitectura aplicației 12](#_Toc71556743)

[Baza de date 12](#_Toc71556744)

[Server 15](#_Toc71556745)

[III. Detalii de implementare 18](#_Toc71556746)

[Crearea unui cont și logarea 19](#_Toc71556747)

[Adăugarea unui concept și editarea sa 21](#_Toc71556748)

[Filtrarea conceptelor după tag-uri și editarea tag-urilor 27](#_Toc71556749)

[Postarea comentariilor 30](#_Toc71556750)

[Tehnologii folosite 33](#_Toc71556751)

[IV. Manual de utilizare 42](#_Toc71556752)

[Crearea unui cont 42](#_Toc71556753)

[Vizualizarea conceptelor asociate unei materii 45](#_Toc71556754)

[Adăugarea unui nou concept 49](#_Toc71556755)

[Filtrarea conceptelor după tag-uri 50](#_Toc71556756)

[Vizualizarea comentariilor asociate unui concept și postarea de noi comentarii 53](#_Toc71556757)

[Aflarea materiei corespunzătoare unui concept anume 54](#_Toc71556758)

[Vizualizarea grafului 55](#_Toc71556759)

[Vizualizarea statisticilor 56](#_Toc71556760)

[V. Concluzii și direcții viitoare 58](#_Toc71556761)

[VI. Bibliografie 59](#_Toc71556762)

# Introducere

Un concept poate fi reprezentat de către o noțiune teoretica, o formula, un algoritm, o teorema , o euristica ,etc.

Aplicația este menita să servească pe post de instrument didactic, ajutând atât studenții cat și profesorii. Studenții vor fi ajutați să se pregătească mai bine din punct de vedere teoretic pentru susținerea examenului de licență . Profesorii vor putea să structureze mai ușor materia ce trebuie învățată de către studenți pentru examenul de licență .

Aceasta le va oferi studenților posibilitatea să se pregătească mai bine și mai rapid pentru examenul de licență, acumulând intr-un singur loc toate cunoștințele considerate importante din cadrul tuturor materiilor obligatorii la licență cat și a interdependentelor existente intre acestea .

Totodată, ea va oferi o unealta folositoare profesorilor în stabilirea cunoștințelor/teoremelor/algoritmilor etc. necesare susținerii examenului de licență în cadrul materiilor pe care aceștia le predau. În acest sens, profesorii vor putea adaugă cunoștințe/concepte noi necesare studiului materiilor pe care ei le predau dar vor putea de asemenea să stabilească conexiuni intre aceste concepte și alte concepte de la alte materii, ori intre aceste concepte și alte materii, etc .

Acest lucru este unul important, deoarece facilitează procesul de învățare pentru studenți oferindu-le acestora toata informația și toate conceptele ce trebuie asimilate în vederea susținerii examenului de licență intr-un format schematic, bine structurat. Astfel, studenții vor putea realiza ușor conexiuni intre concepte studiate în cadrul unei anumite materii/mai multor materii învățând mai logic .

De asemenea, profesorii vor putea să asigneze oricâte taguri doresc sau considera necesare fiecărui concept în parte. Ei vor putea să șteargă din tagurile existente ale unui concept sau să ii adauge alte taguri noi chiar și după ce conceptul a fost adăugat în baza de date.

Tag-urile dintre care profesorii vor putea alege în vederea asignării lor diferitelor concepte existente în aplicație sunt cele definite în cadrul “ACM Computing Classification System” în anul 2012 și pot fi vizualizate la următoarea [adresa web](https://dl.acm.org/ccs).

Aplicația va oferi și opțiunea filtrării conceptelor existente în baza de date după unul sau după mai multe taguri atât profesorilor cat și studenților.

In acest sens, dacă studentul dorește să afle ce lucruri predate în facultate fac parte din cadrul unei anumite arii de cunoaștere desemnata de anumite tag-uri el poate filtra toate conceptele existente și stocate în aplicație după respectivele tag-uri. Studentul va obține astfel o lista cu denumirile tuturor conceptelor predate în facultate care se înscriu în respectiva arie de cunoaștere și va putea astfel să își reîmprospăteze memoria în ceea ce privește fiecare dintre ele.

Una dintre cele mai importante caracteristici a gestiunii cunoașterii este reprezentata de către interacțiunea intre utilizatori în timp real. În acest sens, aplicația va avea implementat un sistem de comentarii ce va permite studenților să interacționeze intre ei sau chiar să interacționeze cu profesori în vederea obținerii unei înțelegeri cat mai bune a cunoștințelor predate de către aceștia.

Fiecare concept în parte va avea propria sa secțiune de comentarii unde atât profesorii cat și studenții vor putea posta. Profesorii vor putea să și șteargă din comentariile postate spre deosebire de studenți.

Un avantaj al acestui lucru poate fi reprezentat de faptul ca uneori studenții pot descoperi anumite resurse suplimentare ce ar putea fi folosite pentru înțelegerea mai temeinica a unui anumit concept anume, iar prin intermediul comentariilor aceștia ar putea efectua un schimb de cunoaștere intre ei. Totodată, în cazul anumitor setup-uri ce trebuie efectuate de exemplu s-ar putea ca unii studenți să aibă o configurație diferita de cea uzuala, folosită de majoritatea studenților, iar acesta secțiune de comentarii ar putea fi utilizata de studenți pentru a încerca să rezolve intr-un mod rapid și eficient problemele întâmpinate.

In același timp, este probabil ca pentru studierea anumitor concepte din cadrul anumitor cursuri să fie necesara cunoașterea altor concepte din cadrul altor cursuri sau a altor concepte din cadrul aceluiași curs . În același timp este posibil ca un același concept să fie necesar studiului mai multor materii exact în aceeași forma . Exemplu : Algoritmi precum BFS, DFS sunt folosiți în cadrul mai multor cursuri și stau la baza a multor altor concepte . Acesta este motivul pentru care stabilirea și reprezentarea acestor conexiuni/legături intre diferite concepte este deosebit de importanta .

Fiecare materie în parte va avea propria ei pagina unde toate detaliile și proprietățile conceptelor ce ii aparțin vor putea fi vizualizate dar și modificate de către profesori . Adâncimea la care se face aceasta căutare va putea fi setata în mod interactiv .

Conceptele mai strâns legate de o materie anume vor fi mai apropiate de aceasta decât cele mai specifice, mai slab legate. Altfel spus, conceptele de baza, foarte generale vor trebui legate direct de materie în timp ce conceptele mai specifice decât ele vor fi legate de acestea din urma . Pe măsura ce ne îndepărtam de o materie, conceptele ce se afla în componenta conexa aparținând materiei respective vor fi mai specifice, mai slab legate de aceasta.

Tocmai din aceasta cauza oferirea unei opțiuni în vederea setării adâncimii la care se va face căutarea conceptelor aparținând unei materii este foarte importanta întrucât un student poate decide singur ce și cat anume dorește să învețe în funcție de cunoștințele sale precedente sau de interesul manifestat fata de un domeniu anume. În acest sens, dacă el dorește să învețe doar anumite noțiuni de baza, generale, aparținând unei materii anume el va asigna acestei adâncimi un număr mic . Dar dacă studentul dorește să aprofundeze cunoștințele sale deja existente sau să învețe lucruri specifice sau chiar foarte specifice atunci el are mereu opțiunea de a seta un număr mare sau foarte mare ca adâncime pana la care să se efectueze căutarea de concepte legate de materia respectiva, selectata în prealabil de către student.

Pentru stocarea și reprezentarea conceptelor cat și a legăturilor dintre acestea va fi folosită o baza de date de tip graf și anume Neo4J.

Vor exista doua label-uri diferite, Concept respectiv Materie. Conceptele vor putea fi legate intre ele prin intermediul diverselor relații de către profesori sau ele vor putea fi legate direct de materii.

Graful construit de aplicație pe partea de back-end va putea fi accesat atât din cadrul interfeței web a Neo4J cat și prin intermediul aplicației, acest lucru putând fi foarte util pentru acei studenți ce au o memorie vizuala foarte buna sau pentru studenții care vor să aibă o vedere de ansamblu a tuturor lucrurilor predate în facultate și a interdependentelor dintre ele.

Aplicația va pune la dispoziția studenților și o pagina de statistici unde aceștia vor putea consulta diverse statistici precum: cele mai folosite taguri, conceptele care au cel mai mare număr de relații din aplicație ( acele noduri care au gradul cel mai mare în cadrul bazei de date de tip graf stocata cu ajutorul Neo4J), etc.

Aceste lucruri se pot dovedi foarte folositoare pentru un student deoarece un concept care are multe relații asociate lui va reprezenta probabil un element foarte important de înțeles în vederea înțelegerii și învățării multor alte concepte.

In mod similar, un tag folosit des poate fi de interes pentru un student întrucât acesta va putea apoi filtra conceptele după respectivul tag în vederea aflării mai multor informații specifice domeniului desemnat de către acel tag.

In caz ca studentul cunoaște denumirea unui concept dar nu își mai amintește exact numele materiei în cadrul căreia respectivul concept a fost predat, aplicația va oferi o funcționalitate de determinare a materiei corespunzătoare acelui concept dat de către student.

Fiecare concept în parte va avea obligatoriu un nume . Numele ales va trebui să nu mai fi fost folosit pentru desemnarea altui concept în mod anterior deoarece conceptele vor fi identificate în mod unic prin intermediul numelor lor . Acesta poate avea și o descriere ce va putea conține diverse informații utile studenților precum link-uri către materiale bibliografice, diverse tutoriale, explicații suplimentare, etc. dar și mai multe relații către alte concepte/materii.

Fiecare relație va avea un nume iar o relație anume va putea exista doar o data (nu va fi permisa existența relațiilor duplicate). Relațiile vor fi identificate unic după numele lor și după numele nodurilor intre care se afla.

Numele , descrierea, relațiile existente ale unui concept vor putea fi modificate de către profesori în orice moment, chiar dacă conceptul exista deja, atât timp cat vor fi respectate anumite restricții precum cea legata de unicitatea numelui conceptului.

Conceptele asociate diferitelor materii cat și materiile în sine vor reprezenta noduri în cadrul bazei de date de tip graf, iar interdependentele existente intre acestea vor fi reprezentate sub forma de arce.

Ca aplicații similare putem aminti aplicația dezvoltata ca rezultat al unei colaborări dintre Pennsylvania State University și Microsoft Corporation în anul 2017 și care este pe larg descrisa în cadrul articolului științific numit “Recovering Concept Prerequisite Relations from University Course Dependencies” care poate fi pe larg consultat accesând următoarea pagina [web](http://clgiles.ist.psu.edu/pubs/AAAI2017-course.pdf).

Aceasta aplicație își propune să primească ca date de intrare conceptele necesare învățării unei materii cat și a relațiilor dintre ele în cadrul aceleași materii pentru mai multe materii. Scopul ei este să găsească automat legăturile/conexiunile dintre concepte asociate materiilor diferite. Datele experimentale furnizate în cadrul articolului arata ca metoda dezvoltata de către respectiva echipa de cercetători este mai buna decât metodele cunoscute anterior indiferent de tipul setului de date care este considerat (real sau sintetic) .

# Specificații funcționale

Aplicația va oferi un mediu user-friendly profesorilor prin intermediul căruia aceștia vor putea reprezenta schematic cunoștințele și conceptele ce trebuie studiate și asimilate de către studenți în cadrul materiilor pe care aceștia le predau în vederea susținerii examenului de licență dar și a legăturilor existente intre acestea .

Studenții vor putea accesa și vizualiza aceasta reprezentare schematica din cadrul fiecărei materii în parte. Totodată ei vor putea vedea și tine cont de dependentele existente intre anumite concepte sau intre anumite concepte și anumite materii. Aceștia vor putea de asemenea să consulte indicațiile oferite de profesori pentru fiecare concept în parte, sau să consulte resursele externe oferite de către aceștia .

La baza acestei aplicații va sta o baza de date de tip graf care va avea ca rol reprezentarea relațiilor existente dintre noduri .

Fiecare nod va avea mai multe proprietăți asociate lui deoarece de exemplu profesorii vor putea specifica resurse externe ce ar trebui consultate de către studenți în vederea asimilării conceptului desemnat de nodul amintit mai sus . De asemenea profesorii ar putea oferi o descriere generala a respectivului concept, etc.

In același timp, dintr-un nod pot pleca mai multe arce către mai multe alte noduri, acest lucru semnificând faptul ca intre nodul inițial și nodurile destinație exista o legătura, o relație. În cazul acestei aplicații, acest lucru înseamnă ca intre conceptul desemnat de nodul inițial și conceptul desemnat de nodul destinație exista o legătura . Ex: dacă exista arc de la nodul A la nodul B, acest lucru înseamnă ca cele 2 concepte desemnate de nodurile A și B sunt legate intre ele, existând o relație intre ele . De exemplu, conceptul desemnat de nodul A poate fi necesar a fi cunoscut înaintea studierii conceptului B .

In acest sens, aplicația va avea 2 tipuri de utilizatori: profesor și student.

Informațiile corespunzătoare fiecărui cont creat vor fi stocate în interiorul unei baze de date.

Profesorii vor trebui să își creeze un cont în timp ce studenții pot opta pentru crearea unui cont sau nu .

Aplicația va oferi și cate o secțiune de comentarii dedicata fiecărui concept în parte pentru a facilita împărtășirea de cunoaștere intre studenți sau intre studenți și profesori, însă doar utilizatorii logați vor posta comentarii. Totuși, atât utilizatorii logați cat și cei nelogați vor putea vizualiza comentariile deja postate.

Fiecare concept în parte va putea avea tag-uri asociate lui. Aceste tag-uri vor fi extrase efectuând o operațiune de scraping pe acest [site](https://dl.acm.org/ccs), aparținând celor de la ACM(Association for Computing Machinery).

Atât utilizatorii ce poseda privilegii de profesor cat și cei ce sunt simpli studenți vor putea filtra conceptele stocate în baza de date de tip graf după unul sau mai multe tag-uri.Astfel, dacă se va efectua o filtrare după mai multe tag-uri, aplicația va returna o lista a conceptelor care au măcar respectivele tag-uri asociate lor.

Motivul pentru care profesorii trebuie să aibă un cont este acela ca ei vor putea modifica baza de date de tip graf aflata la baza aplicației, în timp ce studenții vor putea doar vizualiza informațiile introduse de către profesori în interiorul acestei baze de date, ei nu vor putea efectua modificări asupra acesteia . Indiferent de modul în care studenții vor alege să folosească aplicația ( fie ca utilizatori autentificați ceea ce presupune crearea unui cont, fie ca simpli vizitatori ) aceștia nu vor putea modifica informațiile introduse de către profesori , nu vor putea efectua schimbări asupra bazei de date de tip graf creata de către profesori .

In acest sens profesorii vor putea de exemplu să adauge concepte noi ce trebuie studiate și învățate în cadrul materiilor predate , acestea fiind reprezentate sub forma de noduri dar totodată ei vor putea stabili legături intre aceste concepte și altele deja existente, sau intre aceste concepte și diferite alte materii. Legăturile amintite anterior vor fi reprezentate sub forma de arce intre nodurile specificate de către profesori . Desigur, profesorii vor putea și să vizualizeze atât informațiile introduse de către ei înșiși cat și cele introduse de către alți profesori .

Profesorii vor putea totodată să asigneze oricâte tag-uri conceptelor dar și să efectueze orice tip de modificare asupra conceptelor deja create. Conceptelor deja existente li se va putea schimba numele, descrierea, șterge anumite relații, adaugă noi relații, adaugă noi tag-uri, șterge anumite tag-uri deja asociate, șterge din comentariile asociate, etc. de către utilizatori cu privilegii de profesor.

In același timp, conceptele vor putea fi șterse în întregime de către profesori.

Din punct de vedere arhitectural, se va recurge la folosirea design pattern-ului MVC (Model View Controller) .

Aplicația va expune un API REST și va fi responsive ( putând fi astfel folosită cu ușurință și pe dispozitive mobile ) .

Cu ajutorul API-ului vor fi implementate toate operațiunile precum adăugarea unei noi relații unui concept, adăugarea unui comentariu, ștergerea unui comentariu, modificarea numelui, a descrierii unui concept, ștergerea unui concept în sine împreună cu toate informațiile și relațiile asociate lui.

# Arhitectura aplicației

În acest capitol voi evidenția arhitectura aplicației Knowledge Base.

În ceea ce privește arhitectura aplicației, aceasta este structurata conform design pattern-ului MVC(Model View Controller). Pentru stocarea datelor am utilizat o baza de date MySQL dar și o baza de date de tip graf implementata cu ajutorul Neo4j, conferind astfel aplicației o eficiență sporita.

Partea de back-end se constituie dintr-un server Apache conectat la o baza de date MySQL și la o baza de date de tip graf stocata în Neo4j.

## Baza de date

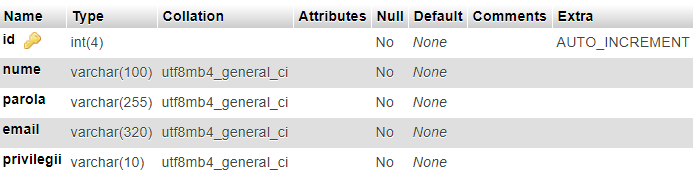
Pentru a spori gradul de securitate al site-ului au fost folosite statement-uri prepared de fiecare data când a fost interogata baza de date MySQL.

Baza de date MySQL este formata din următoarele tabele:

* utilizatori
* taguri
* comentarii

În tabelul “utilizatori” vor fi stocate toate informațiile corespunzătoare conturilor utilizatorilor și anume numele, parola, email-ul(acest câmp poate lipsi) cat și privilegiile lor(profesor sau student). După crearea contului, aceste informații vor fi folosite pentru operațiunea de logare.

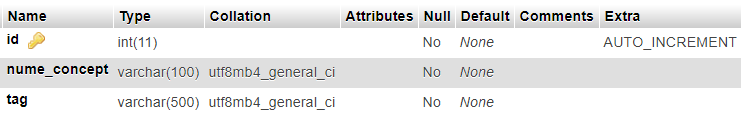
Numele fiecărui utilizator va fi unic.



*Figură 1 Structura tabelului utilizatori*

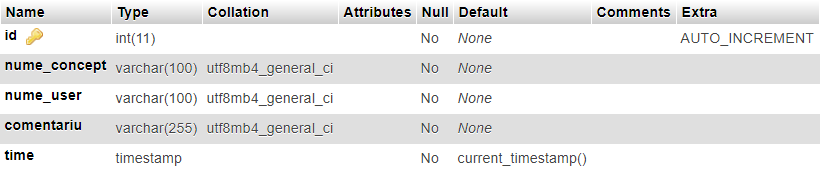
In tabelul “taguri” vor fi stocate perechi de forma (nume\_concept,tag). Acest tabel va fi responsabil de stocarea tag-urilor ACM corespunzătoare unui concept în parte.

O astfel de pereche (nume\_concept,tag) va fi unica în cadrul acestui tabel.



*Figură 2 Structura tabelului taguri*

In tabelul “comentarii” vor fi stocate comentariile lăsate de utilizatori anumitor concepte. Astfel un rând din acest tabel va conține numele conceptului căruia comentariul ii aparține, numele utilizatorului care a postat respectivul comentariu, data și ora postării sale cat și conținutul comentariului în sine.



*Figură 3 Structura tabelului comentarii*

Materiile vor avea concepte asociate lor, conceptele vor intra în diferite relații cu alte concepte sau cu materii. Un concept va putea avea asociate lui însuși mai multe tag-uri și mai multe comentarii. Un utilizator va putea posta mai multe comentarii.

In cadrul bazei de date de tip graf stocata cu ajutorul Neo4j vor exista 2 etichete(label-uri):

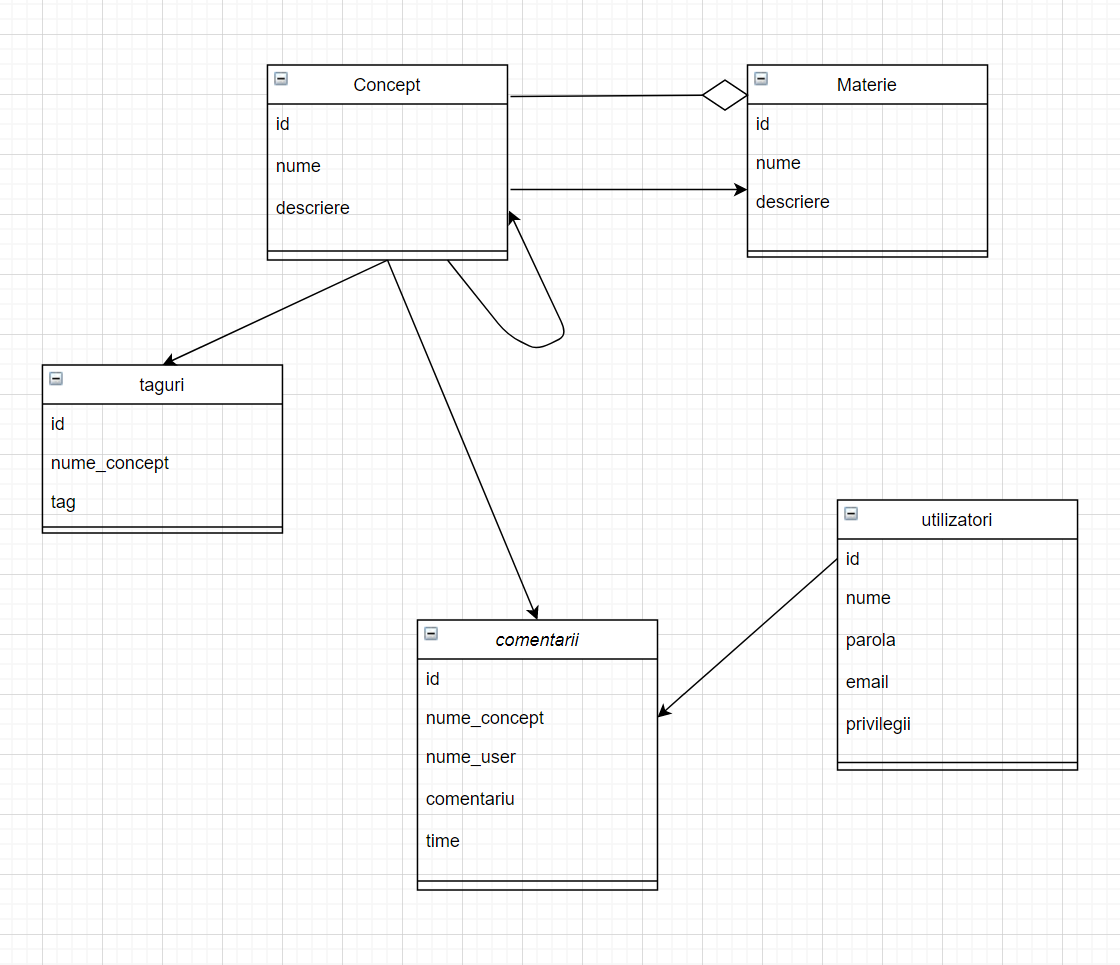
* Concept
* Materie

Fiecare nod va avea 2 proprietăți și anume nume și descriere.

Nodurile vor putea fi legate intre ele prin oricât de multe relații, aceste relații având la rândul lor nume specifice.

Numele fiecărui nod va fi unic în cazul bazei de date.

Nu vor putea exista relații duplicate.



*Figură 4 Privire de ansamblu*

## Server

După cum am amintit, partea de server respecta design pattern-ul MVC(Model View Controller).

Pentru parsarea URL-ului catre API a fost folosit cod din cadrul laboratoarelor de Tehnologii Web din anul II.

Exista 3 modele:

* Concept
* Relație
* User

In cadrul modelului Concept sunt implementate toate funcțiile necesare lucrului cu concepte, precum adăugarea, redenumirea, modificarea descrierii,etc a unui concept anume.

In cadrul modelului Relație vor fi implementate funcțiile responsabile de adăugarea respectiv ștergerea unei relații intre un concept și un altul sau intre un concept și o materie.

In cadrul modelului User vor exista funcții ce se vor ocupa cu anumite operațiuni precum crearea unui cont, logarea, deconectarea, etc.

In cadrul aplicației exista mai multe View-uri, fiecare dintre ele controlând o pagina anume a site-ului și anume:

* index
* indexLogat
* signup
* adaugaConcept
* materiiProfesor
* concepteleUneiMaterii
* concepteleUneiMateriiStudent
* filtrareProfesor
* filtrareStudent
* comentarii
* comentariiStudent
* aflaMateria
* graf
* statistici

In cazurile în care o pagina arata suficient de diferit în versiunile ei destinate profesorilor respectiv studenților, am optat în a avea doua view-uri diferite corespunzătoare ei, unul pentru versiunea ce va fi afișată profesorilor iar unul pentru versiunea ce va fi disponibila studenților.

View-urile “index” și “indexLogat” vor reprezenta cele doua versiuni ale paginii de start:atunci când utilizatorul nu este logat respectiv atunci când este conectat în aplicație.

View-ul “signup” va reprezenta pagina prin intermediul căreia un utilizator va putea să își creeze un nou cont în cadrul aplicației.

View-ul “adaugaConcept” corespunde paginii prin intermediul căreia pot fi adăugate noi concepte în aplicație, pagina ce va putea fi accesata doar de către profesori.

View-ul “materiiProfesor” va desemna pagina ce va conține o lista cu toate materiile obligatorii disponibile pentru a fi alese în cadrul părții teoretice a examenului de licență. Aceasta pagina va putea fi accesata atât de studenți cat și de profesori.

View-urile “concepteleUneiMaterii” respectiv “concepteleUneiMateriiStudent” reprezintă cele doua versiuni a unei aceleași pagini și anume pagina prin intermediul căreia pot fi vizualizate conceptele aparținând unei materii. Profesorii vor putea să adauge relații, să șteargă relații, să schimbe numele diverselor concepte, să le modifice descrierea, etc prin intermediul acestei pagini în timp ce studenții vor putea doar vizualiza informațiile introduse de către profesori.

View-urile “filtrareProfesor” respectiv “filtareStudent” corespund paginii ce oferă utilizatorilor opțiunea de a filtra conceptele existente după unul sau mai multe tag-uri. Profesorii vor putea de asemenea să adauge tag-uri noi sau să șteargă din cele deja existente în timp ce studenții vor putea doar vizualiza informațiile corespunzătoare fiecărui concept în parte.

View-urile “comentarii” și “comentariiStudent” aparțin paginii destinate postării și vizualizării de comentarii. Profesorii vor avea opțiunea de a șterge comentarii.

View-ul “aflaMateria” conduce către o pagina ce va oferi posibilitatea utilizatorilor de a determina care este materia asociata unui concept anume.

View-ul “graf” reprezintă o pagina ce va afișa baza de date de tip graf stocata în Neo4j în cadrul aplicației web.

View-ul “statistici” desemnează pagina ce va afișa diverse statistici precum:conceptele cu cel mai mare număr de tag-uri, cel mai des folosite tag-uri,conceptele cu cele mai multe relații etc.

Aplicația are 3 controllere și anume:

* Home
* Materii
* Signup

Controllerele vor determina care din view-uri va trebui afișat utilizatorului când acesta efectuează anumite acțiuni în cadrul aplicației web, cum ar fi apăsarea unui buton anume.Totodata controllerele decid care funcții din cadrul modelelor vor trebui apelate în funcție de circumstanțe.

Aceste decizii vor fi luate în funcție de anumiți parametri precum privilegiile utilizatorului, statutul utilizatorului(logat sau nelogat în aplicație) etc.

Controller-ul Home va gestiona tot ceea ce tine de pagina de start a aplicației și de operațiunile de logare, delogare și de creare a unui nou cont.

Controller-ul Materii va fi responsabil de toate operațiunile în care sunt implicate concepte respectiv materii, noduri ale grafului stocat în Neo4j sau relații ale acestora.

Controller-ul Signup va afișa view-ul “signup”.

# Detalii de implementare

În continuare voi prezenta detaliile de implementare ale proiectului, și anume modalitățile în care acțiunile principale ale utilizatorului sunt implementate.

## Crearea unui cont și logarea

Toate informațiile legate de aceasta etapa vor fi stocata într-o baza de date MySQL în cadrul tabelului numit “utilizatori”.

Aplicația va avea mai multe variabile de tip sesiune care vor stoca anumite informații legate de utilizatorul ce folosește site-ul.

Aceste variabile sunt:

* $\_SESSION[‘logat’] va stoca dacă utilizatorul este sau nu logat în aplicație.
* $\_SESSION[‘userID’] va stoca id-ul corespunzător utilizatorului
* $\_SESSION[‘privileges’] va stoca privilegiile utilizatorului(student sau profesor)
* $\_SESSION[‘numeUtilizator’] va stoca numele utilizatorului

La crearea unui cont datele introduse vor fi validate iar apoi se va verifica dacă numele introdus este unic în cadrul aplicației. În caz contrar, o eroare va fi returnata iar utilizatorul va fi îndrumat să aleagă alt nume.

La logare, în mod similar, se va verifica dacă datele introduse sunt valide conform formatului specificat și dacă contul cu respectivele credențiale exista.

|  |
| --- |
| $anUser=$this->model('User');  if(isset($\_POST['LoginButton']))  {  $anUser->password=$\_POST['Parola'];  $anUser->username=$\_POST['Nume'];  if($anUser->validareInfoLogin())  {  if($anUser->loginCorect())  {  $\_SESSION['logat']=1;  $this->view('indexLogat');  }  else //parola sau username gresite  {  $this->view('index',['message'=>"Parola sau username gresit!"]);  }  }  else //datele introduse nu respecta formatul  {  $this->view('index',['message'=>"Format invalid al datelor!"]);  }  }  else if (isset($\_POST['SignupSubmitButton']))  {  $anUser->username=$\_POST['Nume'];  $anUser->password=$\_POST['Parola'];  $anUser->email=$\_POST['Email'];  $anUser->privilegii=$\_POST['StudentSAUProfesor'];  if ($anUser->validareDateSignup())  {  if($anUser->numeDisponibil())  {  $anUser->stocheazaNoulContDB();  $\_SESSION['logat']=-1;  $\_SESSION['userID']=-1;  $\_SESSION['privileges']="student";  $this->view('index',['message'=>"Cont creat cu succes!"]);  } |

*Figură 1*

Validarea datelor va fi făcute după lungimea/dimensiunea lor dar și după formatul lor folosind expresii regulate.

Vor fi folosite statement-uri prepared la interacțiunea cu baza de date MySQL.

|  |
| --- |
| public function validareDateSignup()  {  if(strlen($this->username)>100)  return false;  if(strlen($this->password)>50)  return false;  if(strlen($this->email)>70)  return false;  if($this->email!='')  if (!filter\_var($this->email, FILTER\_VALIDATE\_EMAIL)) //validam emailul  return false;  if(!preg\_match("/^[a-zA-Z\dăâîșțĂÂÎȘȚ]+$/",$this->username))  return false;    if(!preg\_match("/^[a-zA-Z\dăâîșțĂÂÎȘȚ]+$/",$this->password))  return false;    return true;  } |

*Figură 2*

## Adăugarea unui concept și editarea sa

Editarea informațiilor aparținând unui concept va fi făcută utilizând rute către un API REST.

Afișarea informațiilor legate de concepte existente va fi făcută cu ajutorul unor apeluri de tip GET către același API menționat anterior.

La adăugarea unui nou concept, datele introduse vor fi validate iar apoi se va verifica dacă numele conceptului pe care utilizatorul dorește să-l creeze a mai fost folosit în mod anterior sau nu.

Dacă datele introduse sunt valide va fi creat un nou nod în cadrul bazei de date de tip graf stocata în Neo4J cu proprietățile(nume și descriere) specificate. Acestui nod ii vor fi adăugate relațiile specificate de către utilizator.

In același timp, va fi creata cate o noua înregistrare în cadrul tabelului taguri din MySQL pentru fiecare tag valid asociat respectivului concept. O înregistrare va fi de forma (nume\_concept,tag).

|  |
| --- |
| public function AdaugareConcept()  {  $unConcept=$this->model('Concept');  if (isset($\_POST['ButonAdaugareConcept']))  {  #$this->view('materiiProfesor');  #var\_dump($\_POST);  $unConcept->denumire=$\_POST['NumeConcept'];  $unConcept->descriere=$\_POST['DescriereaConceptului'];  if(isset($\_POST['numeConceptMaterie']))  {  for($i = 0; $i < count($\_POST['numeConceptMaterie']); $i++)  {  $unConcept->concepteCuCareAreRelații[$i]=$\_POST['numeConceptMaterie'][$i];  $unConcept->relații[$i]=$\_POST['relatieCuEa'][$i];  }  if(count($\_POST['numeConceptMaterie'])!=count($\_POST['relatieCuEa'])|| count($unConcept->concepteCuCareAreRelații)!=count($unConcept->relații))  {  $this->view('adaugaConcept',['message'=>"Relații date incorect!"]);  }  }  if(isset($\_POST['unTag']))  {  for($i = 0; $i < count($\_POST['unTag']); $i++)  {  $unConcept->taguri[$i]=$\_POST['unTag'][$i];  }  }  if($unConcept->validareInfoAdaugareConcept())  {  if($unConcept->NUexistaConceptulDeja())  {  $unConcept->adaugaConceptBD();  $this->view('adaugaConcept',['message'=>"Concept adaugat!"]);  }  else  {  $this->view('adaugaConcept',['message'=>"Concept existent deja!"]);  }  }  else  {  $this->view('adaugaConcept',['message'=>"Formatul datelor este invalid!"]);  }  }  } |

*Figură 3*

Totodată se va verifica dacă tag-urile introduse pentru acel concept sunt toate valide(dacă aparțin celor circa 2100 tag-uri ACM).Dacă sunt găsite tag-uri ce nu aparțin acelei liste(ele au fost introduse în pagina prin modificarea codului sursa a paginii), ele nu vor fi asociate conceptului respectiv.

|  |
| --- |
| public function adaugaConceptBD()  {$client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();    $search = array('@<script[^>]\*?>.\*?</script>@si');  $output = preg\_replace($search, '', $this->descriere);  $this->descriere=$output;  $client->run('CREATE (n:Concept{nume:"' . $this->denumire . '",descriere:"' . addslashes($this->descriere) .'"})');  for($i = 0; $i < count($this->concepteCuCareAreRelații); $i++)  {  $query= "MATCH (a:Concept), (b:Concept)  WHERE a.nume = '" . $this->denumire . "' AND b.nume = '" . $this->concepteCuCareAreRelații[$i] . "'  MERGE (a)-[r:" . $this->relații[$i] . "]->(b)  RETURN type(r)";  $client->run($query); //executam query-ul  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  for($i=0;$i<count($this->taguri); $i++)  {  if(in\_array($this->taguri[$i], $arrayToateTagurile) ) //tagul e valid  {  $query = $con->prepare("INSERT INTO taguri(nume\_concept,tag) values(?,?)");  $query->bind\_param("ss",$this->denumire,$this->taguri[$i  $query->execute(); //executam query-ul  }  }  $con->close();} |

*Figură 4*

In ceea ce privește editarea unui concept la o data ulterioara celei adăugării sale aceasta cuprinde mai multe operațiuni posibile: editarea relațiilor sale, editarea tag-urilor asociate, schimbarea numelui sau, modificarea descrierii sale sau chiar ștergerea sa în întregime.Aceste operațiuni vor putea fi făcute doar de către profesori.

La ștergerea unui concept vor fi șterse toate relațiile asociate lui dar și toate tag-urile și comentariile asociate din tabelul taguri și comentarii stocate cu ajutorul bazei de date MySQL.

|  |
| --- |
| public function stergeNod($numeConcept , $underscore='')  {if($underscore!='')  $numeConcept=str\_replace("\_"," ",$numeConcept);  $client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $numeConcept . '" RETURN nod';  $result = $client->run($query);  $rezultatFinal=array();  foreach ($result->records() as $record)  if( $record->get('nod')->get('nume') != null )  {  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $numeConcept . '" DETACH DELETE nod';  $client->run($query);  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("DELETE FROM comentarii where nume\_concept=? "); //facem prepare la query  $query->bind\_param("s", $numeConcept ); //bind-uim parametrii  $query->execute(); //executam query-ul  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("DELETE FROM taguri where nume\_concept=? ");  $query->bind\_param("s", $numeConcept ); //bind-uim parametrii  $query->execute(); //executam query-ul  return true;  }  else  {  return false;  }  } |

*Figură 5*

La adăugarea unei relații unui concept deja existent, pe lângă validarea denumirii relației în sine se va verifica dacă nodurile intre care va fi construita relația există (nodul de start și nodul de final) dar și dacă nu cumva relația respectiva exista deja.

Relațiile duplicate nu vor fi permise iar o eroare va fi returnata utilizatorului în acest caz.

|  |
| --- |
| public function adaugaRelatie()  {  $this->nod1Nume=str\_replace("%20"," ",$this->nod1Nume  $this->nod2Nume=str\_replace("%20"," ",$this->nod2Nume);  $this->nod1Nume=str\_replace("\_"," ",$this->nod1Nume); //in caz ca avem \_  $this->nod2Nume=str\_replace("\_"," ",$this->nod2Nume);  $client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $this->nod1Nume . '" RETURN nod';  $result = $client->run($query);  $rezultatFinal=array();  foreach ($result->records() as $record)  $nodul1Gasit=$record->get('nod')->get('nume');  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $this->nod2Nume . '" RETURN nod'; //ca sa vedem dacă exista  $result = $client->run($query);  $rezultatFinal=array();  foreach ($result->records() as $record)  $nodul2Gasit=$record->get('nod')->get('nume');    if(!isset($nodul1Gasit))  {  return "Nodul 1 nu exista";  }  if(!isset($nodul2Gasit))  {  return "Nodul 2 nu exista!";  }  $query ='RETURN EXISTS( (:Concept {nume:"' . $this->nod1Nume . '"})-[:' . $this->nume . ']->(:Concept {nume: "' . $this->nod2Nume . '" }) ) as mesaj';  $result = $client->run($query);  foreach ($result->records() as $record)  {  if ($record->get('mesaj')==1) //exista relația, deci nu o mai adaugam  {  return "Relatia exista deja!";  }  else //nu exista deci o adaugam  {  $query='  MATCH (nod1:Concept) WHERE nod1.nume="'. $this->nod1Nume . '"  MATCH (nod2:Concept) WHERE nod2.nume="'. $this->nod2Nume . '"  MERGE (nod1)-[:' . $this->nume . ']->(nod2)'; //cream relatia  $result = $client->run($query);  return "Relație adaugata intre cele 2 noduri specificate!";  }  }  } |

*Figură 6*

Modificarea descrierii va putea fi efectuată prin intermediul unui editor text de tip WYSIWYG(what you see is what you get).Acesta va suporta o varietate de formate/stiluri de scris ce vor putea fi folosite pentru a evidenția conținutul.

Se va verifica dacă respectivul concept exista iar conținutul descrierii va fi sanitizat cu ajutorul bibliotecii HTMLPurifier.

|  |
| --- |
| public function modificaDescrierea($continutDescriereNoua)  {  if($continutDescriereNoua!='')  {  if(strlen($continutDescriereNoua)>1500)  return "Descrierea are peste 1500 caractere";  }  $client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();  $search = array('@<script[^>]\*?>.\*?</script>@si');  $output = preg\_replace($search, '', $continutDescriereNoua);  $continutDescriereNoua=$output;  $purifier = new HTMLPurifier();  $continutDescriereNoua = $purifier->purify($continutDescriereNoua);  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $this->denumire . '" RETURN nod';  $result = $client->run($query);  foreach ($result->records() as $record)  if( $record->get('nod')->get('nume') != null )  {  $query ="MATCH (nod:Concept) WHERE nod.nume='". $this->denumire . "' SET nod.descriere='" . addslashes($continutDescriereNoua) . "'";    $result = $client->run($query);  return "Descrierea modificata cu succes!";  }  return "Conceptul cu numele dat nu exista!"; |

*Figură 7*

## Filtrarea conceptelor după tag-uri și editarea tag-urilor

Conceptele existente în aplicație vor putea fi filtrate după tag-urile asociate lor.Astfel, ele vor putea fi filtrate după unul sau mai multe tag-uri(minim un tag).

Pentru a obține lista conceptelor ce au o mulțime anume de tag-uri asociata se va interoga tabelul “taguri”.

Fiecare element prezent în mulțimea de tag-uri după care se va face filtrarea va fi validat, el trebuind să aparțină tag-urilor ACM existente. În acest mod, aplicația nu va permite filtrarea după tag-uri ce nu pot fi asociate unui concept, chiar dacă codul sursa a paginii este modificat.

|  |
| --- |
| public function filtreazaDupaTagurileDate($arrayTaguri)  {  $arrayTaguri=array\_unique($arrayTaguri); //in caz ca s-a dat același tag de 2 ori  $sir='';  foreach($arrayTaguri as $tag)  {  $sir=$sir . ",'" . $tag . "'";  }  $sir=substr($sir, 1); //stergem primul ,  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = "select taguri.nume\_concept " .  "from taguri " .  "where taguri.tag in (" . $sir . ") " .  "group by taguri.nume\_concept " .  "having count(distinct taguri.tag) = " . count($arrayTaguri) ;    $result = mysqli\_query($con, $query); //executam query-ul  $rezultat=array();  if (mysqli\_num\_rows($result) > 0)  {  while($row = mysqli\_fetch\_assoc($result))  {  $rezultat[]=$row;  }  $con->close();  return $rezultat;  }  else  {  $con->close();  return "Nu exista concepte cu toate aceste taguri!";  }  } |

*Figură 8*

Tag-urile existente asociate unui concept vor putea fi șterse. De asemenea, vor putea fi asociate tag-uri noi unui concept. Ambele operațiuni vor putea fi efectuate doar de către utilizatorii care au privilegii de profesor.

La ștergerea unui tag va fi returnata o eroare dacă perechea (nume\_concept,tag) nu există în tabelul taguri și deci nu a fost ștearsa.

|  |
| --- |
| public function stergeUnTag($tagDeSters)  {  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("SELECT \* from taguri where nume\_concept=? and tag=?");  $tagDeSters = str\_replace('%20', ' ', $tagDeSters);  $query->bind\_param("ss",$this->denumire, $tagDeSters);  $query->execute(); //executam query-ul  $result=$query->get\_result();  if($result->num\_rows>=1) //am gasit perechea  {  $query = $con->prepare("DELETE from taguri where nume\_concept=? and tag=?");  $query->bind\_param("ss",$this->denumire, $tagDeSters);    $query->execute(); //executam query-ul    if(mysqli\_affected\_rows($con)==1)  {  $con->close();  return "Tag sters!";  }  else  {  $con->close();  return "Eroare!";  }  }  else  {  $con->close();  return "Perechea concept-tag nu exista!";  }  } |

*Figură 9*

La adăugarea unui tag, se va verifica dacă perechea (nume\_concept,tag) exista deja în tabelul taguri. Dacă aceasta pereche exista deja, atunci aceasta nu va mai fi adaugata întrucât tag-ul respectiv ar fi asociat de doua ori aceluiași concept în cazul respectiv. Se va returna o eroare în acest caz.

Totodată tag-ul va fi validat, el trebuind să aparțină listei tag-urilor ACM.

|  |
| --- |
| public function adaugaUnTag($tagDeAdaugat)  {  $client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $this->denumire . '" RETURN nod';  $result = $client->run($query);  $tagDeAdaugat = str\_replace('%20', ' ', $tagDeAdaugat);    if(!in\_array($tagDeAdaugat, $arrayToateTagurile) ) //tagul nu e valid  {  return "Tagul nu este valid conform ACM CCS!";  }  foreach ($result->records() as $record)  if( $record->get('nod')->get('nume') != null ) //exista  {  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("SELECT \* from taguri where nume\_concept=? and tag=?");  $query->bind\_param("ss",$this->denumire, $tagDeAdaugat  $query->execute(); //executam query-ul  $result=$query->get\_result();  if($result->num\_rows>=1) //am gasit perechea  {  $con->close();  return "Exista deja tagul specificat pentru conceptul dat!";  }  $query = $con->prepare("insert into taguri(nume\_concept,tag) values(?,?)");  $query->bind\_param("ss",$this->denumire, $tagDeAdaugat);  $query->execute(); //executam query-ul    if(mysqli\_affected\_rows($con)==1)  {  $con->close();  return "Tag adaugat!";  }  else  {  $con->close();  return "Eroare!";  }  }  return "Conceptul cu numele dat nu exista!";  } |

*Figură 10*

## Postarea comentariilor

Orice utilizator conectat în cadrul aplicației va putea posta comentarii asociate conceptelor existente, indiferent de privilegiile sale(simplu student sau profesor).

Conținutul comentariilor va fi scris intr-un editor WYSIWYG(what you see is what you get).

Doar utilizatorii logați în aplicație vor putea posta comentarii. Dacă un utilizator nelogat va încerca să posteze un comentariu pe ecran va fi afisata o eroare. Pentru aceasta funcționalitate va fi folosită variabila de tip sesiune $\_SESSION[‘logat’].

Comentariile vor fi stocate în tabelul “comentarii”, fiecare înregistrare aparținând unui comentariu unic și având următoarea forma (id,nume\_concept,nume\_user,comentariu,time).Pentru identificarea numelui utilizatorului care a postat comentariul va fi folosită variabila de tip sesiune $\_SESSION[‘numeUtilizator’].

Conținutul comentariilor va fi validat cu ajutorul bibliotecii PHP HTMlPurifier.

|  |
| --- |
| public function adaugaComentariu($continutComentariu,$numeUtilizator)  {  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("SELECT \* from utilizatori where nume=?");  $query->bind\_param("s",$numeUtilizator); //bind-uim parametrii  $query->execute(); //executam query-ul  $result=$query->get\_result();  $row =$result->fetch\_assoc();  if ($result->num\_rows!=1)  {  return "Numele de utilizator nu exista!";  }    if(strlen($continutComentariu)>1500)  return "Comentariul are peste 1500 caractere!";  $client = ClientBuilder::create()  ->addConnection('default', 'http://neo4j:parola@localhost:7474')  ->build();  $query ='MATCH (nod:Concept) WHERE nod.nume="'. $this->denumire . '" RETURN nod';  $result = $client->run($query);  $search = array('@<script[^>]\*?>.\*?</script>@si');  $output = preg\_replace($search, '', $continutComentariu);  $continutComentariu=$output;  $purifier = new HTMLPurifier();  $continutComentariu = $purifier->purify($continutComentariu);  foreach ($result->records() as $record)  if( $record->get('nod')->get('nume') != null )  {    $query=$con->prepare("INSERT INTO comentarii(nume\_concept,nume\_user,comentariu) values(?,?,?)");  $continutComentariu=addslashes($continutComentariu);  $query->bind\_param("sss",$this->denumire , $numeUtilizator , $continutComentariu );  $query->execute(); //executam query-ul  return "Comentariu adaugat! ID=" . mysqli\_insert\_id($con);  $con->close();  }  return "Conceptul căruia se dorește a fi adaugat comentariul nu exista!";  } |

*Figură 11*

Doar profesorii vor putea șterge dintre comentariile asociate unui concept.Pentru verificarea privilegiilor unui utilizator va fi folosită variabila de tip sesiune $\_SESSION[‘privileges’].

|  |
| --- |
| public function stergeUnComentariu($id)  {  $con=mysqli\_connect("Localhost", "root" ,"", "licenta");  $query = $con->prepare("SELECT \* from comentarii where id=?  $query->bind\_param("i",$id); //bind-uim parametrii  $query->execute(); //executam query-ul  $result=$query->get\_result();  if($result->num\_rows==0)  {  $con->close();  return "Nu exista un comentariu cu acest ID, nu s-a efectuat o stergere!";  }  else  {  $query = $con->prepare("DELETE from comentarii where id=?  $query->bind\_param("i",$id); //bind-uim parametrii  $query->execute(); //executam query-ul  if(mysqli\_affected\_rows($con)==1)  {  return "Comentariu sters!";  }  else  {  return "Comentariul exista dar a aparut o eroare la stergere!";  }  }  } |

*Figură 12*

## Tehnologii folosite

Pentru dezvoltarea aplicației “Knowledge Base” au fost folosite următoarele tehnologii:

* Neo4J
* Server XAMPP
* MySQL

**Neo4J**

Neo4j (Network Exploration and Optimization 4 Java) este un sistem de gestionare a bazelor de date de tip graf dezvoltat de Neo4j, Inc. Descris de dezvoltatorii săi ca fiind o bază de date tranzacțională ACID-compliant, cu stocare și procesare nativă a grafurilor, Neo4j este disponibil într-o "ediție comunitară" open-source licențiată GPL3, cu backup online și extensii de înaltă disponibilitate licențiate sub o licență comercială.

Neo4j este implementat în Java și poate fi accesat prin intermediul software-ului scris în alte limbaje de programare folosind limbajul de interogare Cypher printr-un endpoint HTTP tranzacțional sau prin protocolul binar "bolt".

În Neo4j, totul este stocat sub forma unei muchii, a unui nod sau a unui atribut. Fiecare nod și fiecare muchie în parte pot avea un număr infinit de atribute. Atât nodurile, cât și muchiile pot fi etichetate (le pot fi atribuite label-uri). Etichetele(label-urile) pot fi utilizate pentru a restrânge căutările efectuate. Începând cu versiunea 2.0, indexarea a fost adăugată limbajului de interogare Cypher. Anterior, indexurile erau suportate separat de Cypher.

Unele dintre următoarele caracteristici speciale fac Neo4j foarte popular în rândul dezvoltatorilor și inginerilor software:

* Cypher, un limbaj declarativ de interogare similar cu limbajul SQL, dar optimizat pentru grafuri. Acum, utilizat și de alte baze de date, ar fi “SAP HANA Graph” și “Redis Graph” prin intermediul proiectului openCypher.
* Parcurgeri în timp constant ale grafurilor mari atât în adâncime, cât și în lățime, datorită reprezentării eficiente a nodurilor și relațiilor. Permite scalarea până la miliarde de noduri folosind hardware de capabilități medii.
* Schemă flexibilă de tip “property graph” care se poate adapta ușor în timp, făcând posibilă materializarea și adăugarea de noi relații mai târziu pentru a scurta și accelera lucrurile atunci când situația o cere.
* Drivere pentru limbaje de programare populare, pentru a permite integrarea Neo4J într-o mulțime de medii de dezvoltare diferite. Exista drivere suportate oficial pentru limbaje de programare precum Java, JavaScript, .NET, Python și multe altele.

Versiuni Neo4J



Figură 1

Baze de date de tip graf

Foarte simplu, o bază de date de tip graf este o bază de date concepută pentru a trata relațiile dintre date ca fiind la fel de importante pentru datele în sine. Aceasta baza de date este menita să rețină date fără a le constrânge la un model predefinit. În schimb, datele sunt stocate exact așa cum le extragem mai întâi - arătând modul în care fiecare entitate individuală se conectează cu sau este legată de altele.

Nu există informații izolate, ci doar domenii bogate, conectate intre ele, peste tot în jurul nostru. Numai o bază de date care îmbrățișează nativ relațiile este capabilă să stocheze, să proceseze și să interogheze astfel de conexiuni în mod eficient. În timp ce alte baze de date calculează relațiile la momentul efectuării interogării prin realizarea de operații de tip JOIN costisitoare din punct de vedere computațional, o bază de date de tip graf stochează conexiunile existente împreună cu datele din model.

Accesarea nodurilor și a relațiilor într-o bază de date de tip graf nativă este o operațiune eficientă, realizata în timp constant și permite traversarea rapida a milioane de conexiuni pe secundă pe nucleu de procesor (CPU).

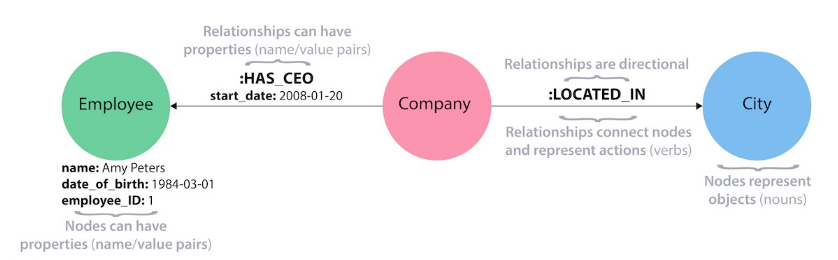
Independente de dimensiunea totală a setului de date existent, bazele de date de tip graf excelează în gestionarea datelor foarte conectate și a interogărilor complexe. Cu doar un model și un set de puncte inițiale, bazele de date de tip graf explorează datele învecinate din jurul acelor puncte de plecare inițiale - colectând și agregând informații din milioane de noduri și relații - și în același timp lăsând neatinse orice alte date din afara perimetrului de căutare.

Modelul Property Graph

Ca și în cazul majorității tehnologiilor, există mai multe abordări diferite pentru a defini ceea ce alcătuiește componentele cheie ale unei baze de date de tip graf. O astfel de abordare este modelul “property graph”, unde datele sunt organizate ca noduri, relații și proprietăți (date stocate pe nodurile sau pe arcele/relațiile grafului).

Nodurile sunt entitățile din graf. Ele pot reține orice număr de atribute (perechi cheie-valoare) numite proprietăți. Nodurilor le pot fi asignate etichete(label-uri), reprezentând rolurile lor diferite în domeniu. Etichetele(label-urile) nodurilor pot servi, de asemenea, la atașarea metadatelor (cum ar fi indexuri sau informații legate de constrângeri) la anumite noduri.

Relațiile oferă conexiuni orientate, denumite, relevante din punct de vedere semantic între două entități de tip nod (de exemplu: Angajat LUCREAZA\_PENTRU Companie). O relație are întotdeauna o direcție, un tip, un nod de start și un nod final. La fel ca nodurile, relațiile pot avea, de asemenea,la rândul lor, proprietăți. În majoritatea cazurilor, relațiile au proprietăți cantitative, precum ar fi weight-uri, costuri, distanțe, evaluări(rating-uri), intervale de timp sau strength-uri. Datorită modului eficient în care sunt stocate relațiile, două noduri pot avea orice număr sau tip de relații comune fără a se sacrifica performanța. Deși sunt stocate într-o direcție specifică, relațiile pot fi întotdeauna navigate eficient în orice direcție.



*Figură 2*

Limbajul de interogare Cypher

Cypher este limbajul de interogare conceput pentru grafuri a Neo4j care permite utilizatorilor să stocheze și să extragă date din baza de date de tip graf. Unul dintre scopurile principale a Neo4j a fost reprezentat de dorința de a face interogarea datelor stocate într-o baza de date de tip graf ușor de învățat, înțeles și utilizat pentru toată lumea, dar și să încorporeze puterea și funcționalitatea altor limbaje standard folosite pentru a accesa date. Acesta este lucrul pe care Cypher își propune să-l realizeze.

Sintaxa Cypher oferă o modalitate vizuală și logică de a face match pe pattern-uri de noduri și de relații dintr-un graf. Este un limbaj declarativ, inspirat de SQL, pentru descrierea pattern-urilor vizuale din grafuri folosind sintaxa ASCII-Art. Ne permite să decidem ce dorim să selectăm, să inserăm, să actualizăm sau să ștergem din baza de date de tip graf fără a oferi o descriere exactă a modului prin care acest lucru ar trebui făcut. Prin Cypher, utilizatorii pot construi interogări expresive și eficiente pentru a gestiona funcționalitățile necesare pentru a crea, citi, actualiza și șterge date.

Cypher este un limbaj de interogare open source. Proiectul openCypher oferă un kit de compatibilitate tehnică și o implementare de referință a parserului, planificatorului și runtime-ului pentru Cypher. Acesta este susținut de mai multe companii din industria bazelor de date și permite implementatorilor de baze de date și clienților să beneficieze în mod liber, să utilizeze și să contribuie la dezvoltarea limbajului de interogare openCypher.

O interogare poate fi scrisa intr-un mod în care să caute valori specifice sau poate fi scrisa cu ajutorul clauzei WHERE în același mod. Ambele interogări vor fi la fel de rapide astfel ca poate fi aleasa oricare dintre ele în funcție de preferințele utilizatorului.

|  |
| --- |
| *//query using equality check in the MATCH clause*  MATCH (j:Person {name: 'Jennifer'})  RETURN j  *//query using equality check in the WHERE clause*  MATCH (j:Person)  WHERE j.name = 'Jennifer'  RETURN j |

*Neo4j 1 Doua modalități de a scrie interogări ce produc aceleași rezultate*

Uneori, poate doriți să returnați rezultate care au valoarea unei anumite proprietăți diferita de o valoare data. În acest caz, trebuie folosită clauza WHERE NOT.

Există câteva tipuri de comparații pe care le puteți rula în Cypher cu ajutorul operatorilor boolean standard: AND, OR, XOR, și NOT. Pentru a afișa o comparație folosind operatorul NOT, vom scrie opusul exemplului de interogare de mai sus.

|  |
| --- |
| *//query using inequality check in the WHERE clause*  MATCH (j:Person)  WHERE NOT j.name = 'Jennifer'  RETURN j |

*Neo4j 2 Folosirea operatorului NOT în cadrul unei interogări*

Există frecvent interogări în care doriți să căutați și să returnați date aflate într-un anumit interval. Intervalele de tip data calendaristica sau numeric pot fi utilizate pentru a căuta evenimente ce au avut loc într-o anumită cronologie,persoane care au o anumita vârstă etc.

Sintaxa pentru acest integrarea acestui criteriu într-o interogare este foarte similară cu cea din SQL și cu cea a altor structuri logice ale diferitelor limbaje de programare pentru verificarea intervalelor de valori.

|  |
| --- |
| MATCH (p:Person)  WHERE 3 <= p.yearsExp <= 7  RETURN p |

*Neo4j 3 Folosirea unui interval numeric*

Este posibil să vă intereseze numai dacă o anumita proprietate există în cadrul unui nod sau în cadrul unei relații. De exemplu, poate doriți să verificați care sunt clienții din sistem care au conturi de Twitter, astfel încât să puteți afișa conținut relevant pentru aceștia.

In Neo4j, o proprietate există (este stocată) numai dacă are o valoare. O proprietate nulă nu va fi stocată. Acest lucru asigură că numai informațiile valoroase, necesare, sunt păstrate în cadrul nodurilor și a relațiilor aflate în baza de date.

Pentru a scrie o astfel de interogare, trebuie doar să utilizați clauza WHERE și metoda exists() pentru proprietatea respectivă.

|  |
| --- |
| *//Query1: find all users who have a birthdate property*  MATCH (p:Person)  WHERE exists(p.birthdate)  RETURN p.name  *//Query2: find all WORKS\_FOR relationships that have a startYear property*  MATCH (p:Person)-[rel:WORKS\_FOR]->(c:Company)  WHERE exists(rel.startYear)  RETURN p, rel, c |

*Neo4j 4 Folosirea metodei exists()*

Unele scenarii de utilizare necesită o sintaxă de interogare care să găsească nodurile a căror proprietăți au anumite valori parțiale. Pentru a face acest tip de interogare, este nevoie de flexibilitate și opțiuni pentru efectuarea căutărilor intr-un sir de caractere sau pentru a face “string-matching”. Indiferent dacă sunteți în căutarea unui șir de caractere care începe cu, se termină cu sau include o anumită valoare, Cypher oferă posibilitatea de a gestiona situația în mod ușor.

Există câteva cuvinte cheie în Cypher utilizate împreună cu clauza WHERE pentru a testa valorile proprietăților de tip sir de caractere. Cuvântul cheie STARTS WITH vă permite să verificați dacă valoarea unei proprietăți începe cu șirul de caractere specificat. Cu ajutorul cuvântului cheie CONTAINS, puteți verifica dacă un șir de caractere specificat face parte din valoarea unei proprietăți. Cuvântul cheie ENDS\_WITH verifică dacă sfârșitul șirului de caractere care determina valoarea proprietății este același cu șirul de caractere specificat.

|  |
| --- |
| *//check if a property starts with 'M'*  MATCH (p:Person)  WHERE p.name STARTS WITH 'M'  RETURN p.name  *//check if a property contains 'a'*  MATCH (p:Person)  WHERE p.name CONTAINS 'a'  RETURN p.name  *//check if a property ends with 'n'*  MATCH (p:Person)  WHERE p.name ENDS WITH 'n' |

*Neo4j 5 Folosirea cuvintelor cheie STARTS WITH, ENDS WITH și CONTAINS*

De asemenea, Cypher suporta folosirea expresiilor regulate în astfel de cazuri.

|  |
| --- |
| MATCH (p:Person)  WHERE p.name =~ 'Jo.\*'  RETURN p.name |

*Neo4j 6 Folosirea expresiilor regulate*

La fel ca în SQL și în alte limbaje de interogare, se poate verifica dacă valoarea unei proprietăți se afla într-o listă. Cuvântul cheie IN vă permite să specificați o serie de valori și să validați conținutul unei proprietăți în raport cu fiecare valoarea din listă.

|  |
| --- |
| MATCH (p:Person)  WHERE p.yearsExp IN [1, 5, 6]  RETURN p.name, p.yearsExp |

*Neo4j 7*

Un lucru care face o baza de date de tip graf unica este concentrarea ei asupra importantei relațiilor. La fel cum exista posibilitatea să filtrați interogările pe baza etichetelor(label-urilor) sau a proprietăților nodului, aveți posibilitatea să filtrați rezultatele și pe baza relațiilor sau pattern-urilor. Acest lucru vă va permite să testați dacă un pattern are, de asemenea, o anumită relație în cadrul său sau nu sau dacă există un alt pattern care să îndeplinească aceasta condiție.

|  |
| --- |
| *//Query1: find which people are friends of someone who works for Neo4j*  MATCH (p:Person)-[r:IS\_FRIENDS\_WITH]->(friend:Person)  WHERE exists((p)-[:WORKS\_FOR]->(:Company {name: 'Neo4j'}))  RETURN p, r, friend  *//Query2: find Jennifer's friends who do not work for a company*  MATCH (p:Person)-[r:IS\_FRIENDS\_WITH]->(friend:Person)  WHERE p.name = 'Jennifer'  AND NOT exists((friend)-[:WORKS\_FOR]->(:Company))  RETURN friend.name |

*Neo4j 8*

Există cazuri în care este posibil să doriți să preluați anumite rezultate din cadrul pattern-urilor, chiar dacă acestea nu fac match pe întregul pattern sau pe toate criteriile specificate. Acesta este modul în care un outer join funcționează în SQL. În Cypher, puteți utiliza un pattern MATCH opțional pentru a încerca să faceți match, dar dacă nu se găsesc rezultate, aceste rânduri vor returna null pentru acele valori.

|  |
| --- |
| *//find all people whose name starts with J and who may work for a company.*  MATCH (p:Person)  WHERE p.name STARTS WITH 'J'  OPTIONAL MATCH (p)-[:WORKS\_FOR]-(other:Company)  RETURN p.name, other.name |

*Neo4j 9*

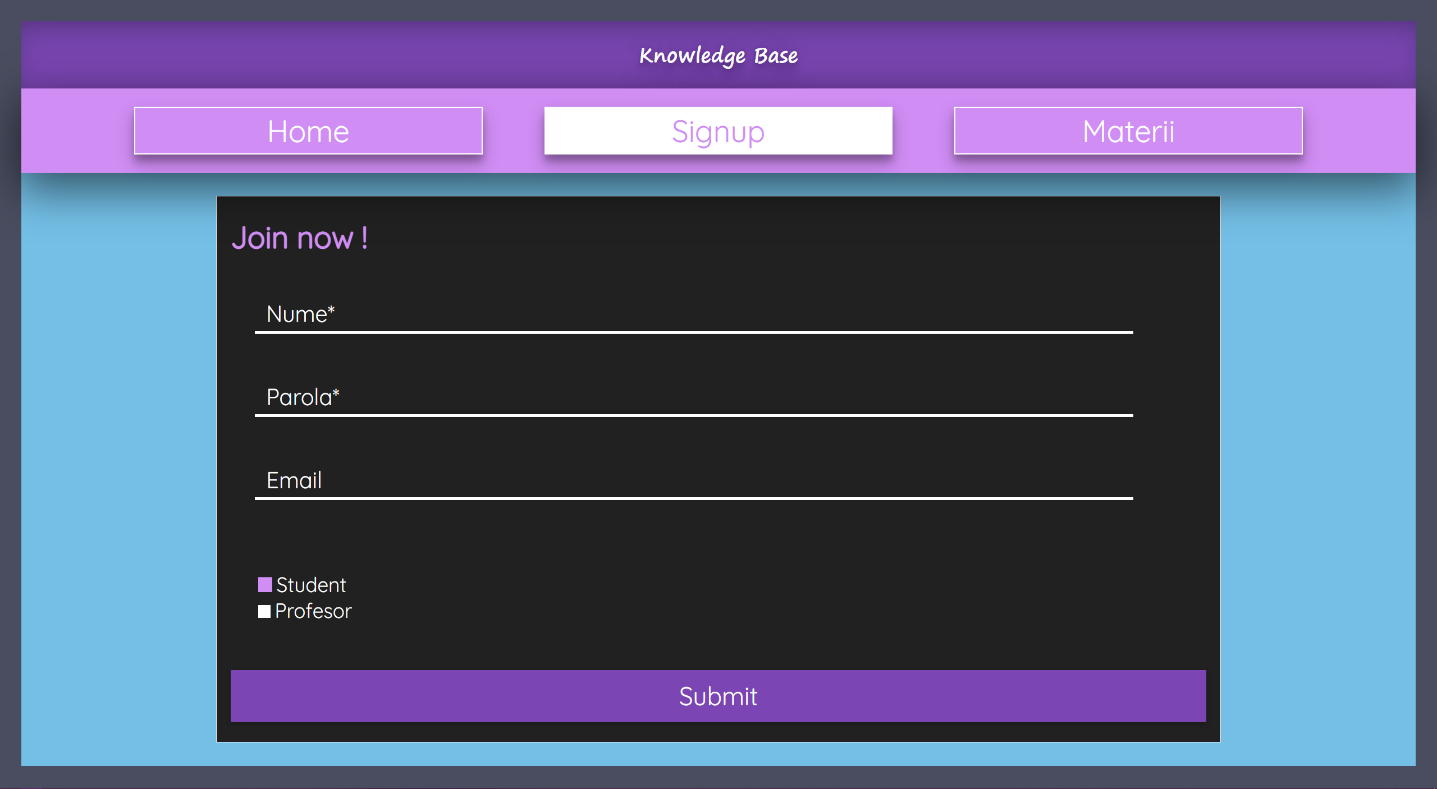
# Manual de utilizare

In cele ce urmează voi prezenta în detaliu modul în care aplicația poate fi utilizată pentru efectuarea diverselor acțiuni.

## Crearea unui cont

Pentru a ajunge la pagina destinată creării unui cont utilizatorul va trebui să apese butonul de “Signup” din meniul principal al aplicației.

Odată ajuns aici, el va trebui să specifice un nume și o parola în mod obligatoriu. Va avea posibilitatea să specifice și o adresa de email, dar acest lucru nu este obligatoriu sau necesar în vederea creării unui cont. De asemenea, el va trebui să specifice dacă este student sau profesor prin intermediul unor checkbox-uri.



*Creare cont 1 O vedere de ansamblu asupra interfetei destinate crearii unui cont*

Numele ales de utilizator va trebui să aibă maxim 50 caractere alfanumerice în timp ce parola sa va trebui să aibă maxim 100 caractere alfanumerice. Numele de utilizator va trebui să fie unic, să nu mai fi fost folosit de alt utilizator. Acest format este specificat utilizatorului la hover peste câmpurile respective. Emailul introdus, dacă utilizatorul optează să facă acest lucru, trebuie să fie valid. Dacă acest format nu este respectat, câmpurile de date își vor schimba culoarea border-ului din partea de jos din alb în roșu, pentru a identifica utilizatorului ca input-ul dat de el nu corespunde constrângerilor impuse.



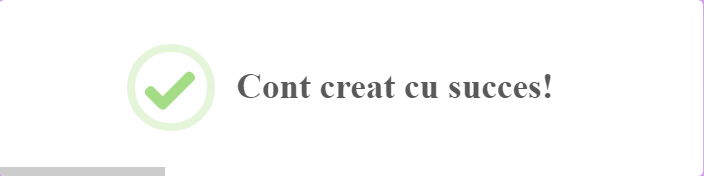
*Creare cont 2 Un astfel de exemplu, când formatul specificat nu este respectat*

Dacă numele de utilizator a fost folosit deja, utilizatorul va fi notificat de acest lucru prin intermediul unui pop-up.



*Creare cont 3 Numele introdus exista deja*

Dacă contul a fost creat cu succes, utilizatorul va fi redirecționat către pagina de start unde acesta se va putea loga cu noul sau cont.



*Creare cont 4 Contul a fost creat cu succes*

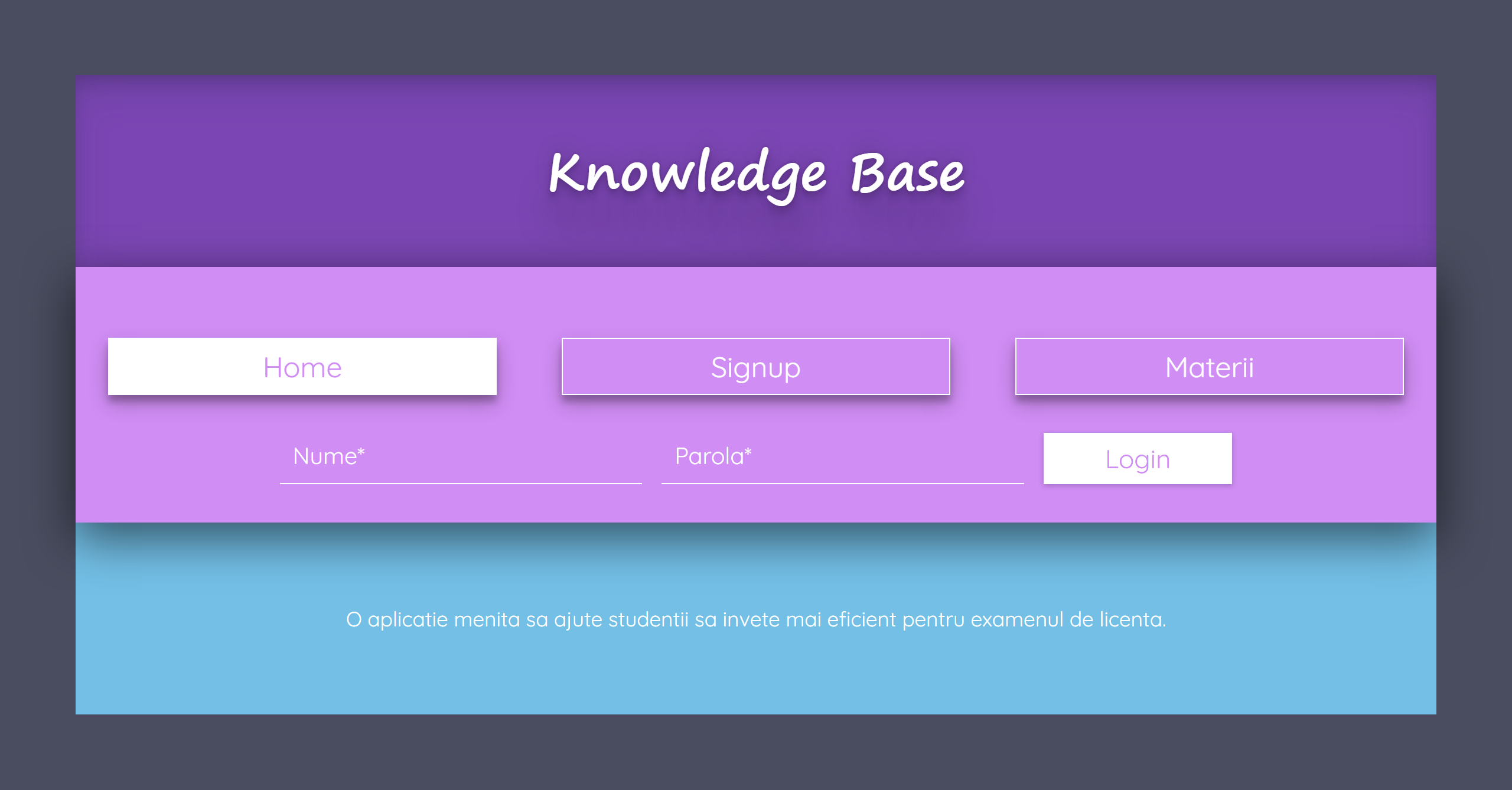
**Logarea și delogarea**

Logarea cat și delogarea se fac de pe pagina principala a aplicației.

Pentru a se loga utilizatorul va trebui să își introducă numele și parola în doua câmpuri de date iar apoi să apese butonul “Login”.

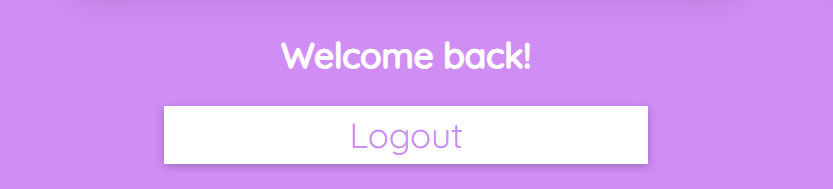
Datele introduse în aceste câmpuri de date vor trebui să respecte un anumit format, 100 caractere alfanumerice pentru cel destinat parolei și 50 caractere alfanumerice pentru cel destinat numelui. Formatul datelor ce trebuie introduse este exact aceleași ca la crearea unui cont. În cazul în care datele introduse nu respecta acest format, borderul de jos a câmpului de date se va colora în roșu.

In caz ca datele introduse de utilizator nu sunt corecte(nu corespund vreunui cont existent în aplicație) el va primi o eroare și va rămâne pe aceeași pagina.



*Logare 1 Câmpurile de date necesare logării*

Dacă logarea decurge cu succes, utilizatorul va fi logat în aplicație și el va putea să se delogheze dacă dorește acest lucru.Zona destinata introducerii datelor de logare se va schimba astfel: Va apărea un text cu conținutul “Welcome” și va apărea un buton de logout. Odată apăsat respectivul buton de logout, utilizatorul va fi deconectat.



*Delogare 1 Butonul de logout*

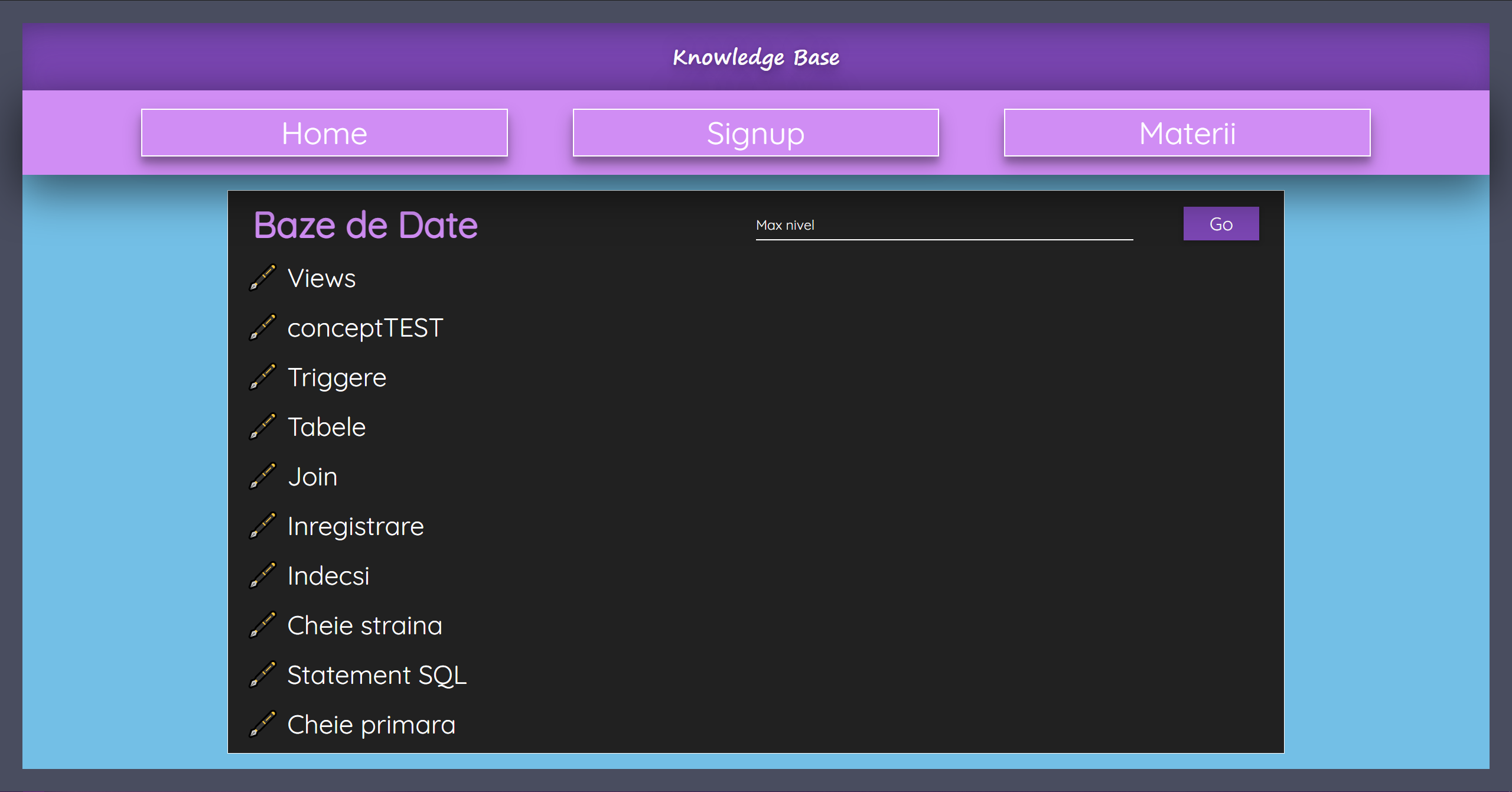
## Vizualizarea conceptelor asociate unei materii

Pentru a vizualiza conceptele asociate unei materii, utilizatorul trebuie să apese butonul “Materii” aflat în meniu pe oricare pagina a aplicației. Odată apăsat acest buton, utilizatorul va fi redirecționat către o pagina numita “Materii” unde el va putea să aleagă materia a cărei concepte dorește să le vizualizeze dintr-o lista.



*Vizualizarea conceptelor unei materii 1. Pagina “Materii”*

Odată ce utilizatorul apasă pe denumirea uneia dintre materii el va fi redirecționat către pagina dedicata vizualizării conceptelor asociate materiei respective.



*Vizualizarea conceptelor unei materii 2 Pagina destinata vizualizării conceptelor unei materii*

Odată ajuns aici, utilizatorul va putea seta adâncimea la care se va face căutarea prin intermediul câmpului de date “Max nivel” și apăsând apoi butonul “Go”.

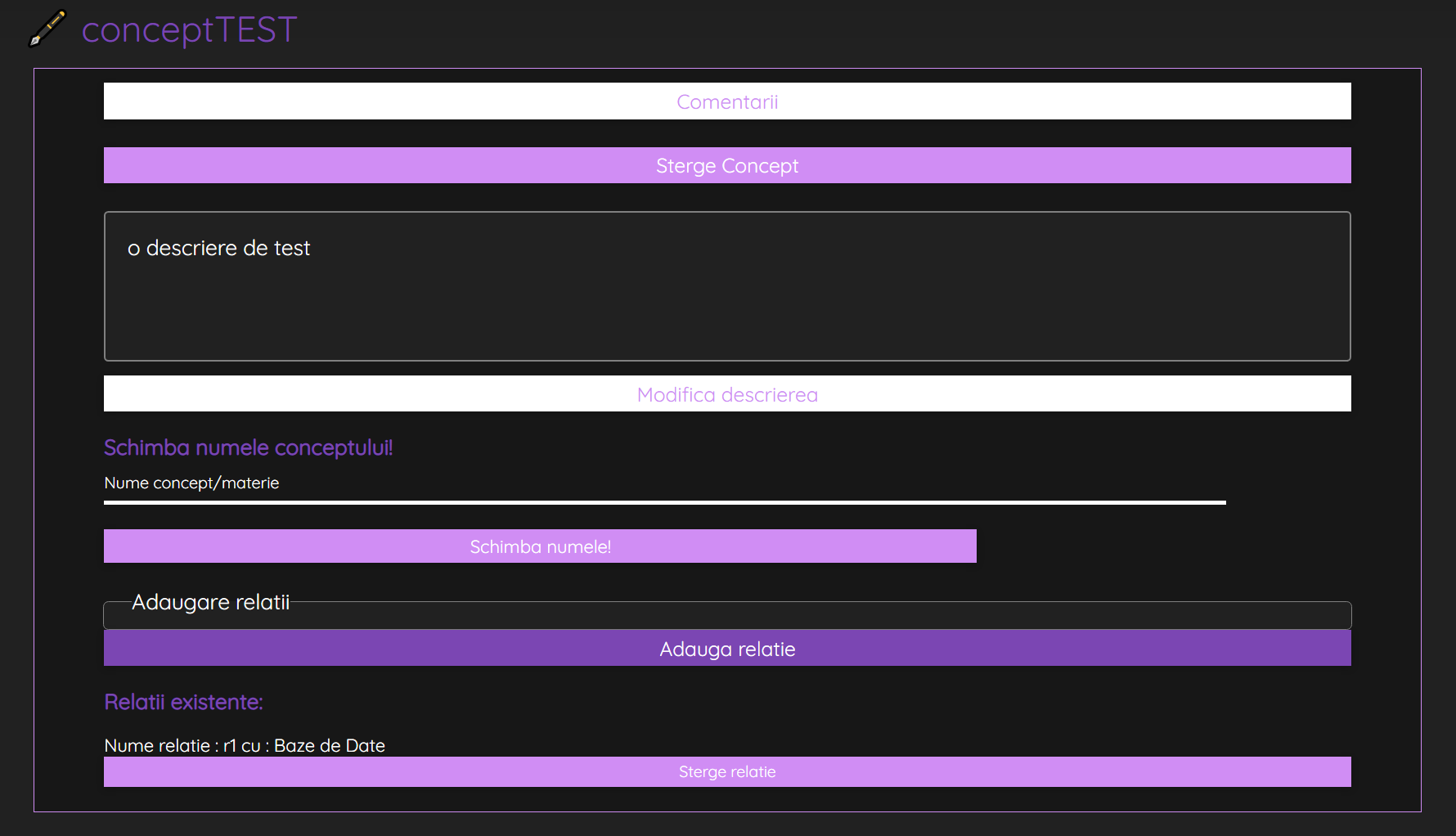
Utilizatorul va putea apoi să apese pe oricare dintre concepte pentru a afla mai multe despre el.

La apăsare pe un concept anume, studenții vor putea doar să vizualizeze descrierea conceptului, dacă acesta are una asociata cat și relațiile respectivului concept cu alte concepte/materii.

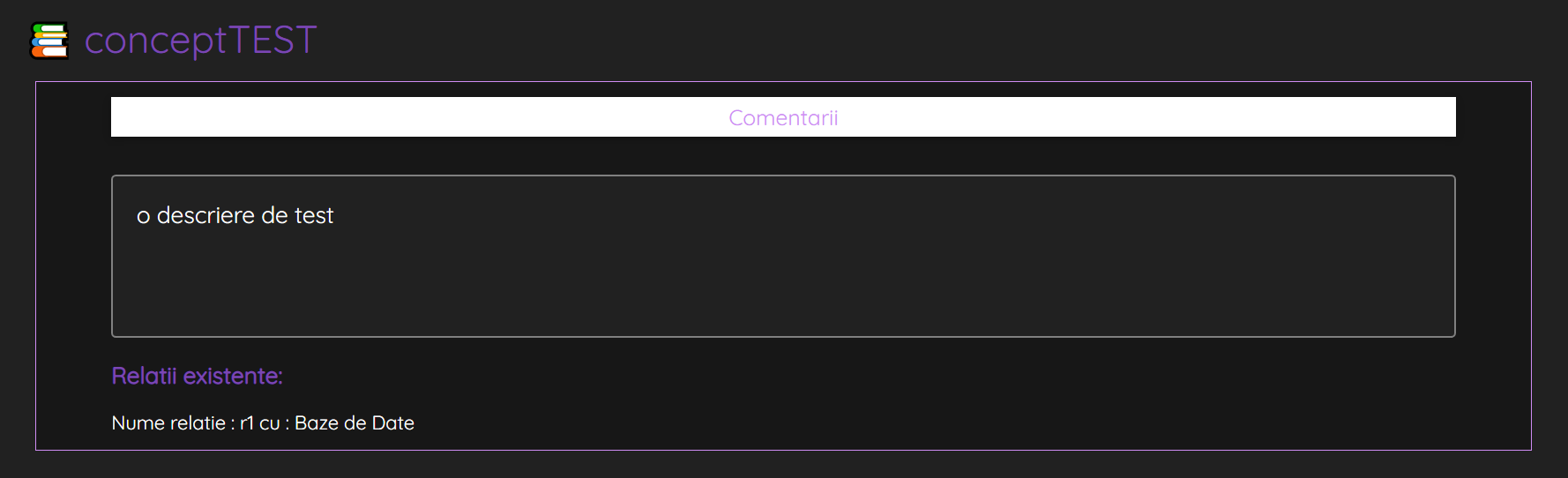
Totodată ei vor putea apasă pe butonul “Comentarii” și vor fi redirecționați către o pagina unde vor putea lăsa comentarii respectivului concept.

Profesorii vor avea la dispoziție mai multe funcționalități. În acest sens, ei vor putea schimba numele conceptului, modifica descrierea asociata lui, șterge relații existente dar și adăuga noi relații asociate respectivului concept. Ei vor putea chiar să șteargă și întregul concept.

Toate câmpurile de date au un format anume ce va fi specificat utilizatorului la hover peste ele. Dacă datele introduse în interiorul lor nu respecta acel format, border-ul din partea inferioara a lor se va colora în roșu.

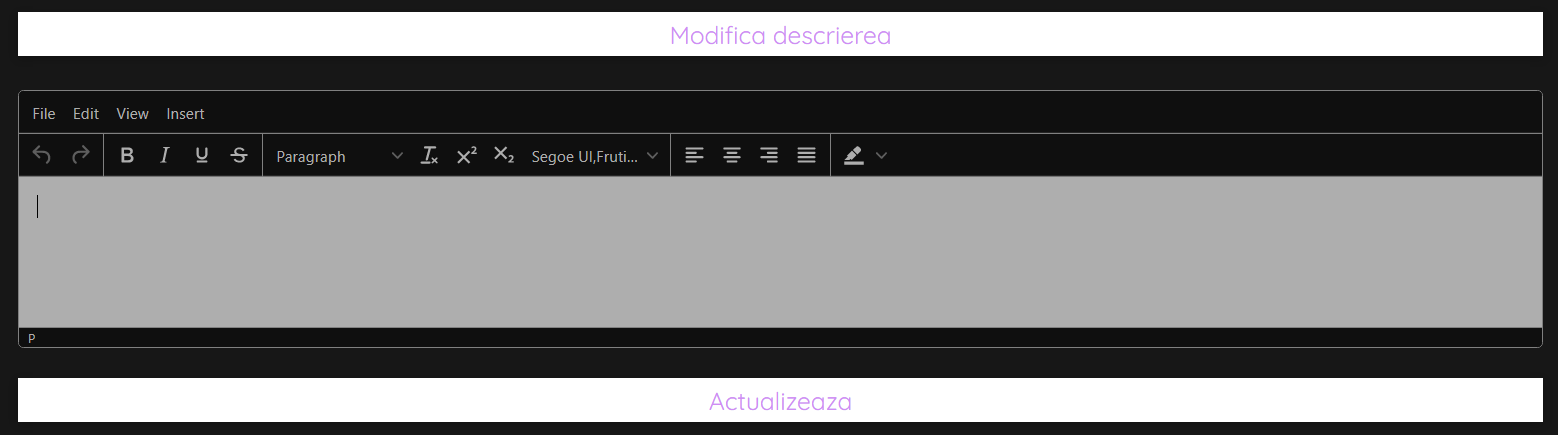


*Vizualizarea conceptelor unei materii 3 Vizualizarea unui concept(Profesor)*



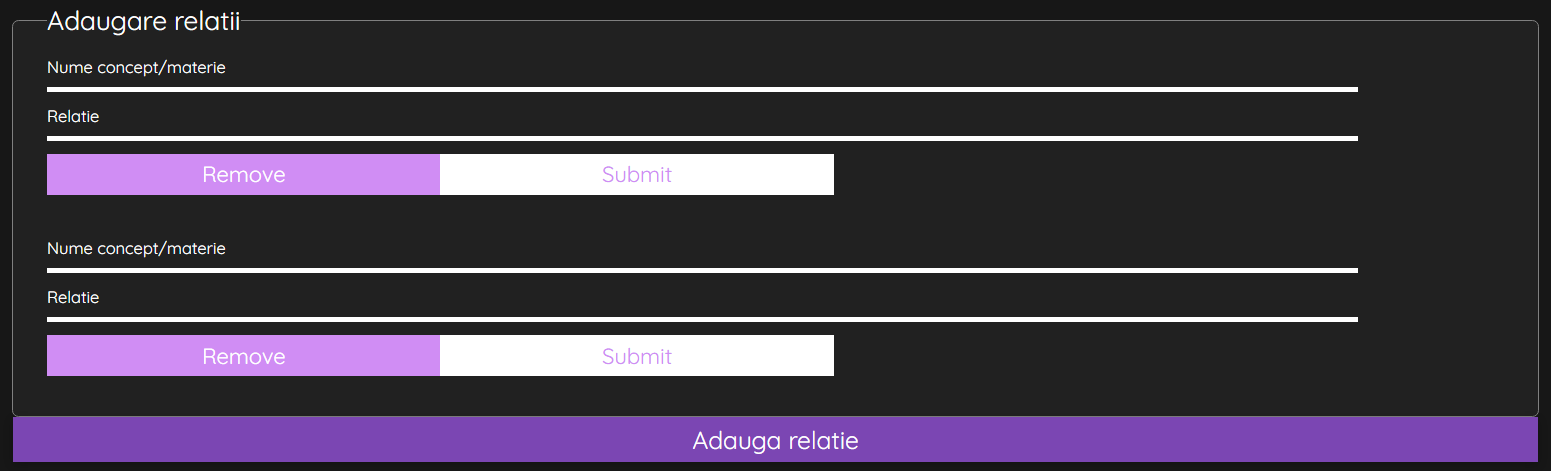
*Vizualizarea conceptelor unei materii 4 Vizualizarea unui concept(Student)*

De exemplu, pentru modificarea descrierii unui concept utilizatorii vor trebui să apese butonul “Modifica descrierea”. Odată apăsat acest buton utilizatorii vor putea să scrie noua descriere în editorul de tip WYSIWYG(what you see is what you get). Când utilizatorul apasă butonul “actualizează” descrierea corespunzătoare conceptului va fi actualizata.



*Vizualizarea conceptelor unei materii 5 Modificarea descrierii conceptului*

Pentru adăugarea de noi relații conceptului va trebui apăsat butonul “Adaugă relație”. Odată apăsat acest buton, va apărea un formular format din 2 câmpuri(“Nume concept/materie” și “Relație”) și 2 butoane(“Remove” și “Submit”). Acest lucru se va întâmpla la fiecare apăsare a butonului.Primul dintre câmpuri va desemna numele conceptului sau a materiei către care se dorește a se adăuga relația. Al doilea câmp va desemna numele care se dorește a se asocia relației. Butonul “Remove” va șterge respectivul mini-formular iar butonul “Submit” va adăuga relația dorita.



*Vizualizarea conceptelor unei materii 6 Adăugarea de noi relații*

Dacă relația respectiva exista deja, ea nu va fi adaugata iar utilizatorul va primi un mesaj de eroare sub forma unui pop-up.

In mod similar, dacă utilizatorul încearcă să schimbe numele unui concept intr-unul deja folosit de un alt concept el va primi o eroare iar aceasta schimbare nu va fi realizata.

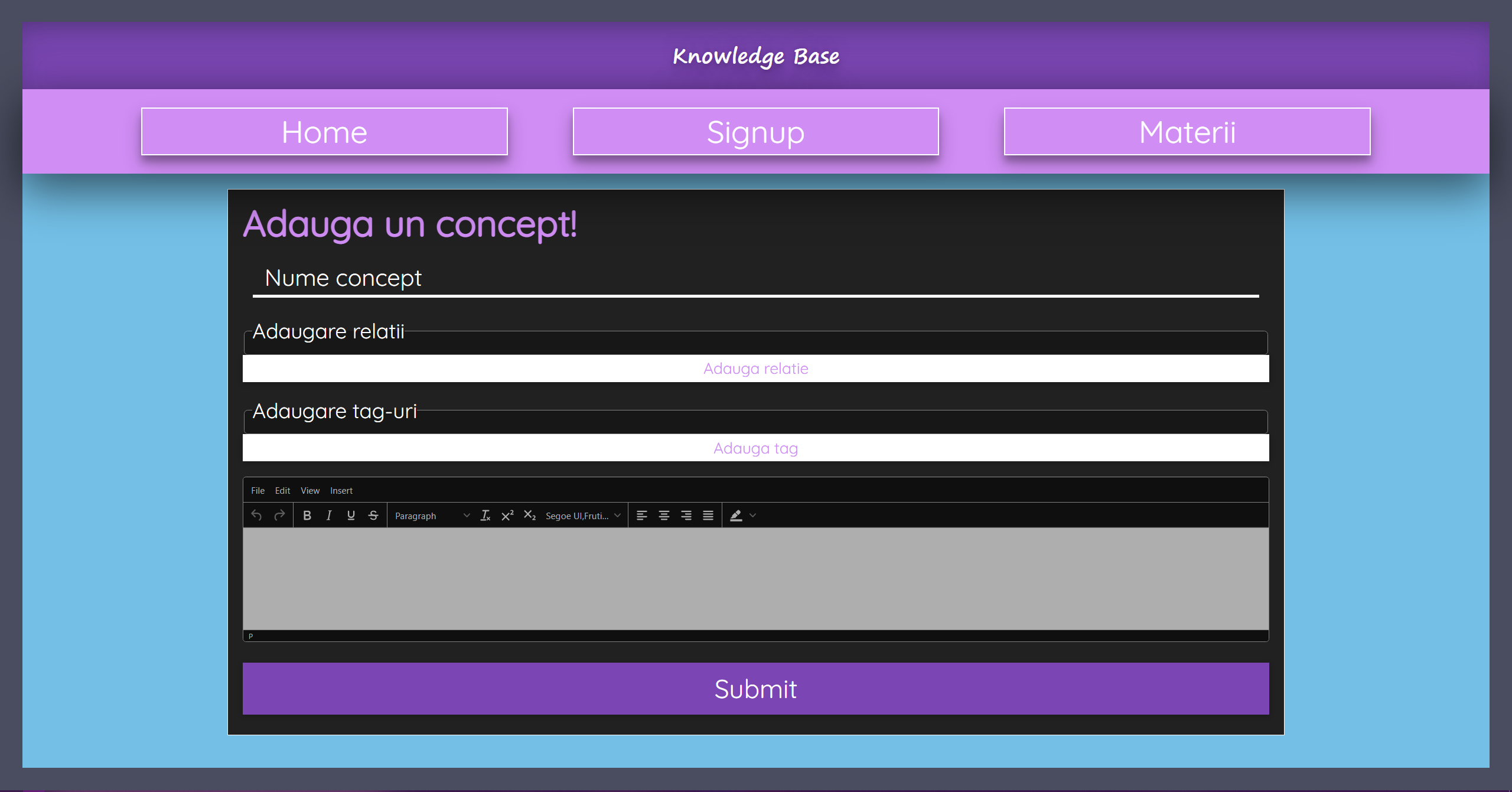


*Vizualizarea conceptelor unei materii 7 Exemplu de eroare*

## Adăugarea unui nou concept

Doar profesorii vor putea adăuga un nou concept.

Pentru a adăuga un nou concept utilizatorii vor trebui să apese butonul “Materii” aflat în meniul aplicației iar apoi să apese butonul “Adaugă un concept” aflat mai jos de meniu pe respectiva pagina. După apăsarea acestui ultim buton ei vor fi redirecționați pe o pagina care va permite adăugarea unui nou concept în aplicație.

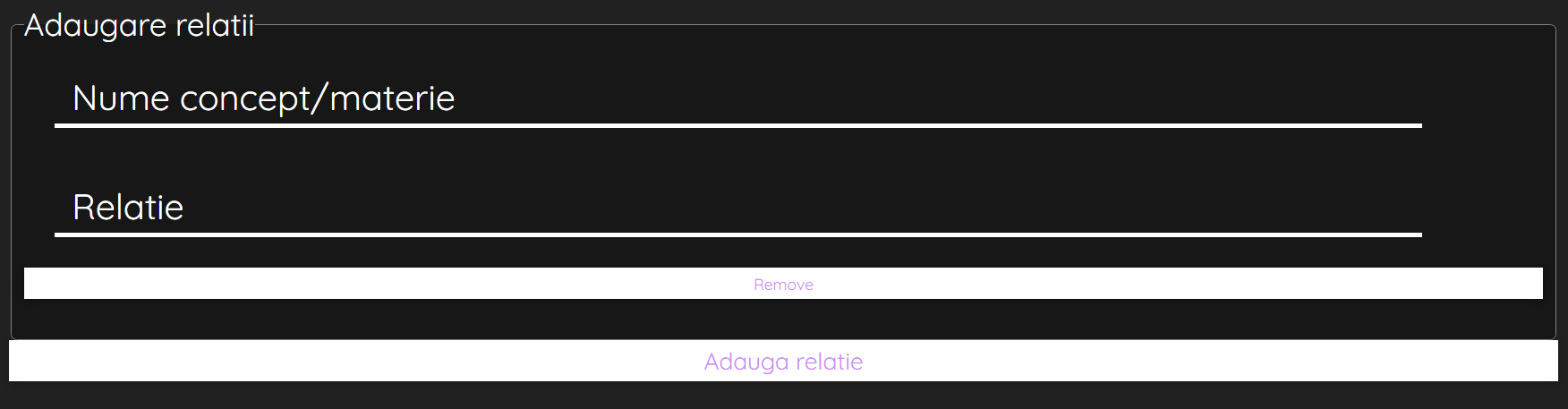


*Adăugare concept 1 Pagina destinata adăugării unui nou concept*

Câmpurile de date vor avea un anumit format ce va fi specificat utilizatorului la hover peste ele.

Un concept nou trebuie să aibă neapărat un nume iar acel nume va trebuie să fie unic, să nu fi folosit deja în aplicație pentru denumirea unui alt concept.Completarea acestui câmp este obligatorie.

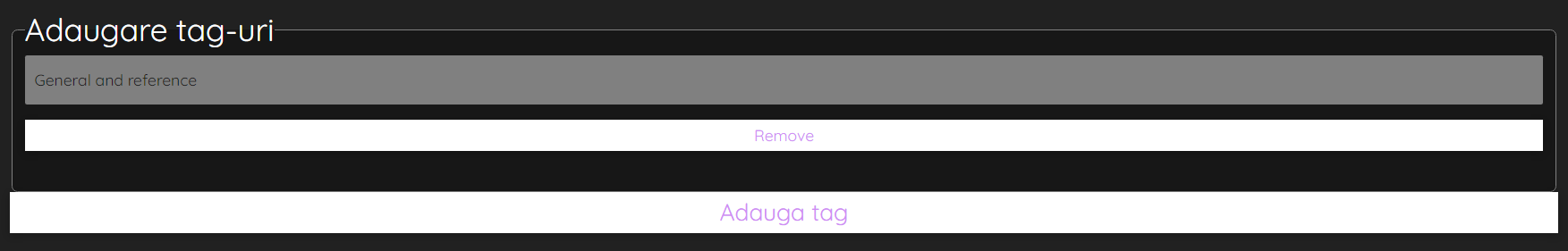
In momentul creării unui nou concept, acestuia i se vor putea asigna oricâte relații. La fiecare apăsare a butonului “Adaugă relație” va fi adaugat un cate nou mini-formular în care vor putea fi specificate numele conceptului cu care noua relație se dorește a fi adaugata cat și numele relației ce se dorește a fi construita. Dacă se apasă butonul “Remove” mini-formularul căruia butonul ii aparține va fi sters.



*Adaugare concept 2 Adaugare de noi relații*

In mod similar, utilizatorul va putea adăuga oricâte taguri dorește unui concept la crearea acestuia.

Tag-urile ce se doresc a fi adăugate vor fi selectate prin intermediul unui meniu drop-down ce va suporta auto-complete.



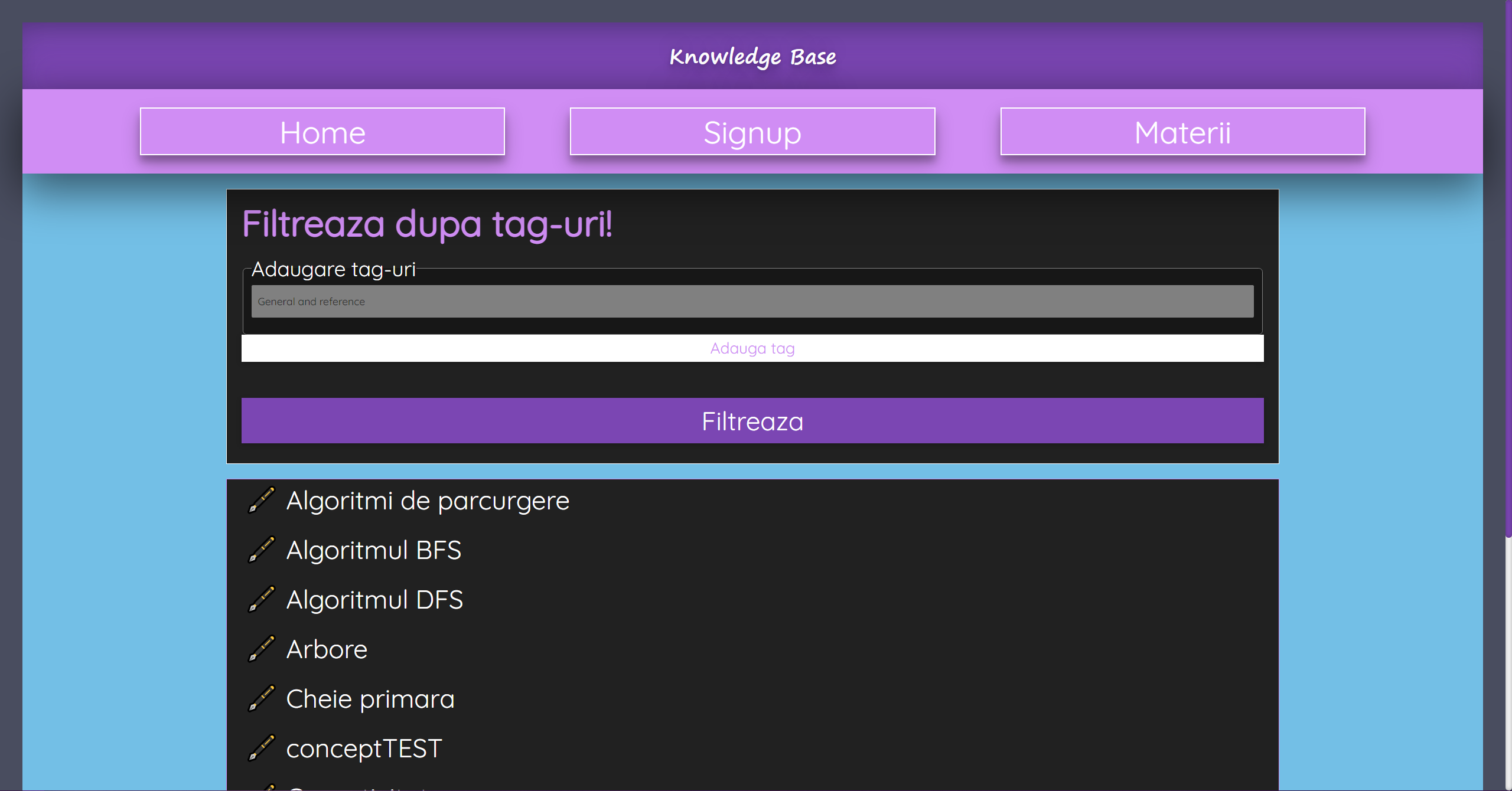
*Adăugare concept 3 Adăugare de noi tag-uri*

Utilizatorul poate specifica și o descriere careva fi asociata conceptului dacă dorește.

La final, odată apăsat butonul “Submit” din partea de jos a paginii conceptul va fi adaugat. Dacă numele introdus de utilizator a mai fost folosit, conceptul nu va fi creat iar utilizatorul va primi o eroare.

## Filtrarea conceptelor după tag-uri

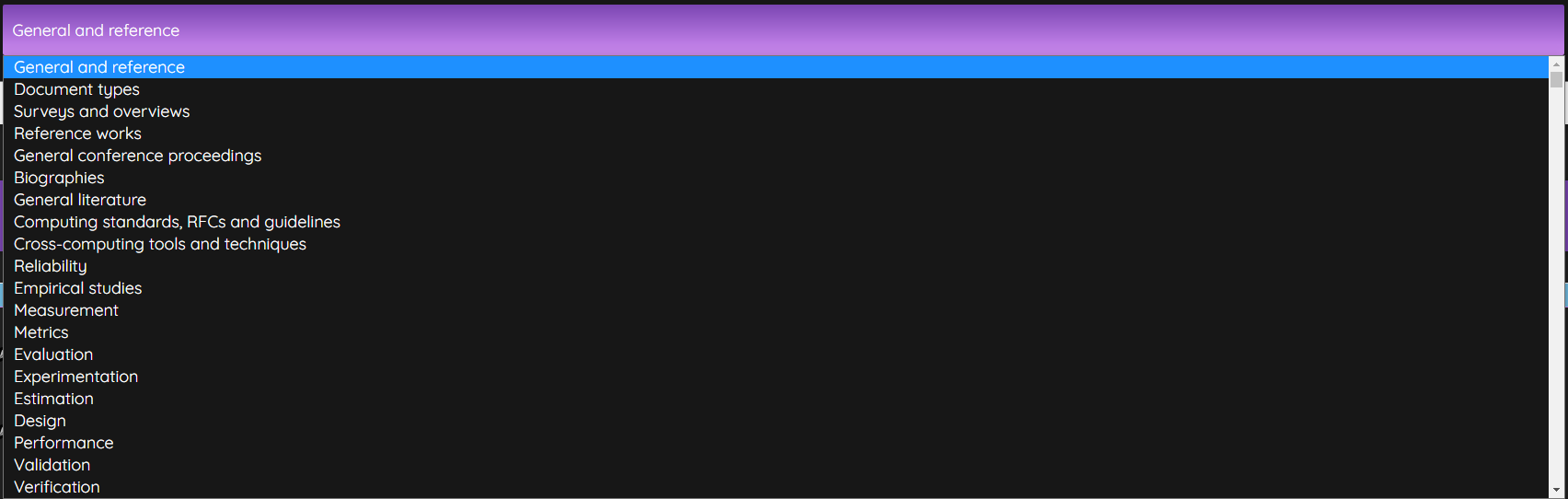
Pentru a filtra conceptele după taguri utilizatorul va trebui să apese butonul “Materii” aflat în bara de meniu iar apoi să apese butonul “Filtrează după tag”.După apăsarea celui din urma buton, el va fi redirecționat către o pagina unde va putea filtra conceptele existente după unul sau mai multe tag-uri.



*Filtrare după tag-uri 1 Privire de ansamblu asupra paginii*

La apăsarea butonului “Adaugă tag” va fi adaugat un nou meniu de tip drop-down prin intermediul căruia va putea fi selectat un nou tag după care să se facă filtrarea.

La apăsarea butonului “Remove” de sub un meniu drop-down acesta va fi șters.



*Filtrare după tag-uri 2 Conținutul meniului drop-down*

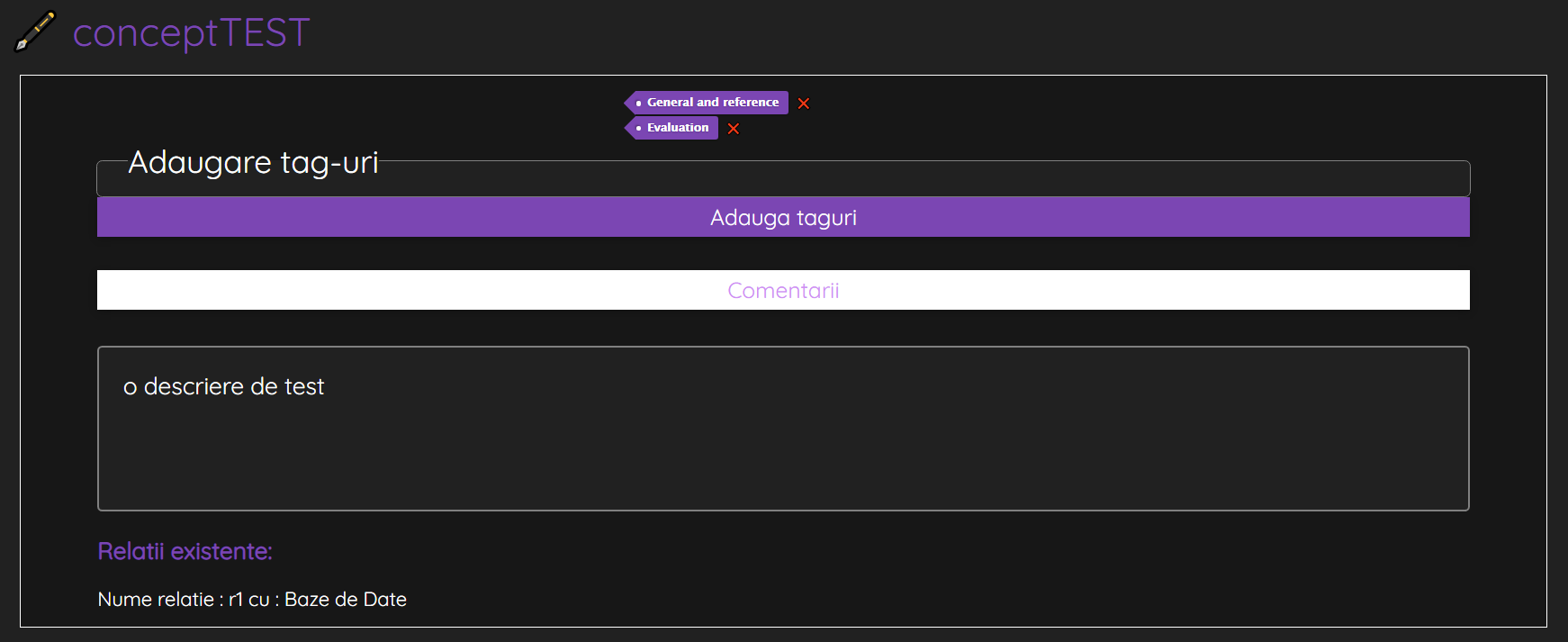
La apăsarea butonului “Filtrează” filtrarea va fi făcută după toate tag-urile specificate. În partea de jos a paginii vor apărea denumirile unor concepte. Fiecare din aceste concepte va avea asociate toate tagurile după care filtrarea a fost făcută.

In caz ca nu a fost găsit niciun concept care să aibă asociate toate tag-urile după care filtrarea a fost făcută în locul listei conceptelor va apărea un mesaj sugestiv.



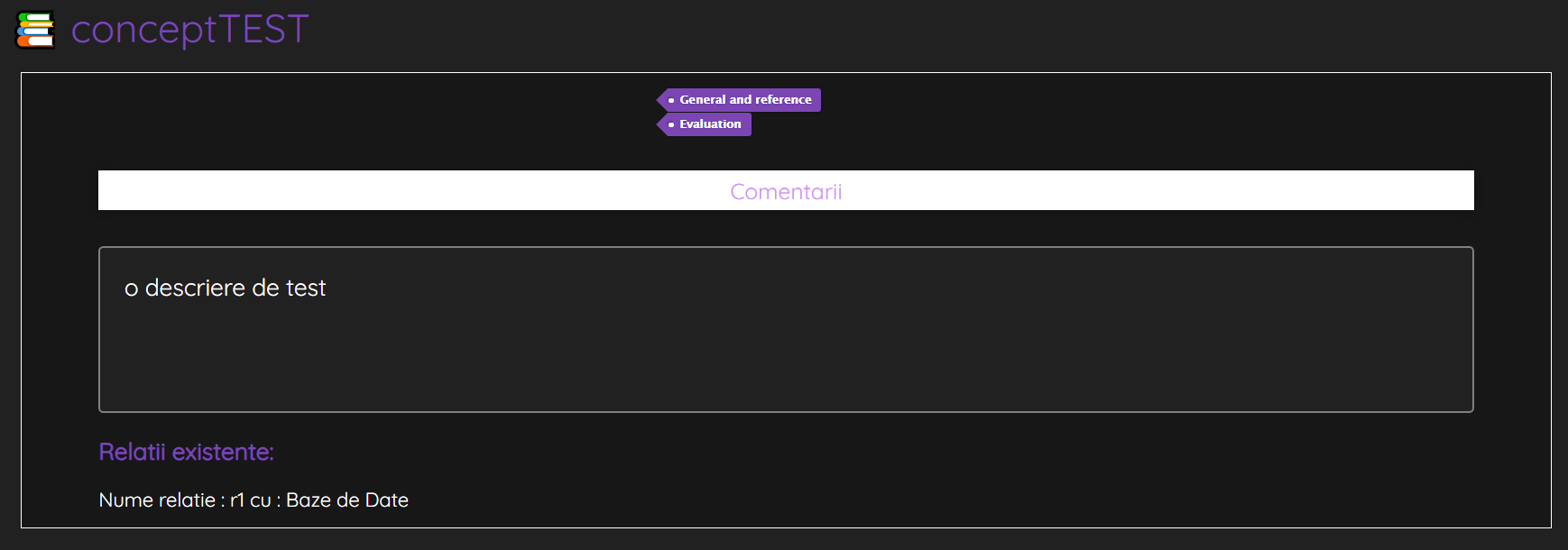
*Filtrare după tag-uri 3 Filtrarea nu a produs niciun rezultat*

La apăsarea pe numele unui concept vor fi afișate mai multe informații despre acesta. Profesorii vor putea să adauge tag-uri noi conceptelor sau să șteargă din tag-urile existente.



*Filtrare după tag-uri 4 Vizualizarea unui concept(Profesor)*

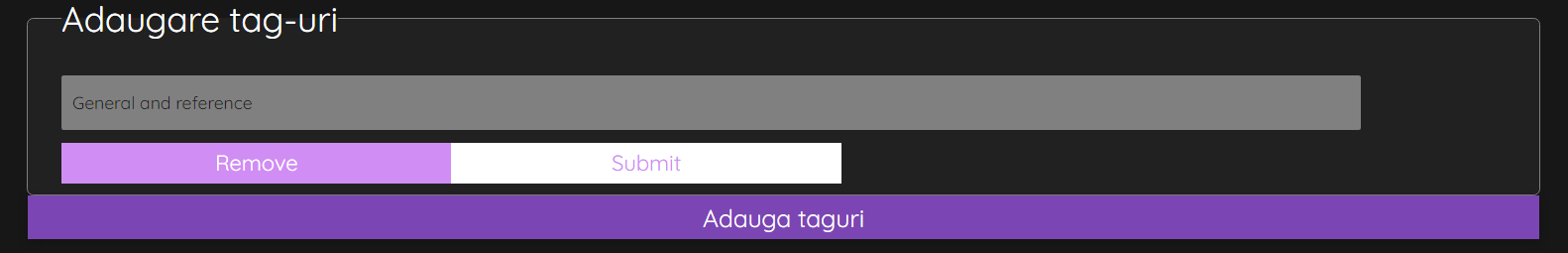
Studenții vor putea doar vizualiza care sunt tag-urile asociate unui concept. Aceștia vor putea de asemenea să vadă descrierea respectivului concept sau care sunt relațiile asociate lui.



*Filtrare după tag-uri 5 Vizualizarea unui concept(Student)*

La click pe X-ul roșu din dreptul fiecărui tag acesta va fi șters.

La fiecare apăsare a butonului “Adaugă taguri” va fi adaugat un nou mini-formular ce va conține un meniu drop-down prin intermediul căruia se va putea selecta denumirea tag-ului ce se dorește a fi adaugat. Prin intermediul butonului “Remove” acest mini-formular va fi șters iar prin intermediul butonului “Submit” tag-ul va fi adaugat conceptului.



*Filtrare după tag-uri 6 Adăugare de noi tag-uri (Profesor)*

Dacă tag-ul respectiv este deja asociat conceptului, utilizatorul va primi o eroare iar operațiunea nu se va realiza.

## Vizualizarea comentariilor asociate unui concept și postarea de noi comentarii

Atât profesorii cat și studenții vor putea lasă comentarii diferitelor concepte.

Pentru a lasă un comentariu unui concept sau pentru a vizualiza comentariile corespunzătoare lui utilizatorul trebuie să apese butonul “Comentarii” asociat respectivului concept.Acest buton poate fi regăsit de fiecare data când se da apasă numele unui concept(in secțiunea de filtrare a conceptelor după tag-uri sau în cea de vizualizare a conceptelor asociate unei materii).

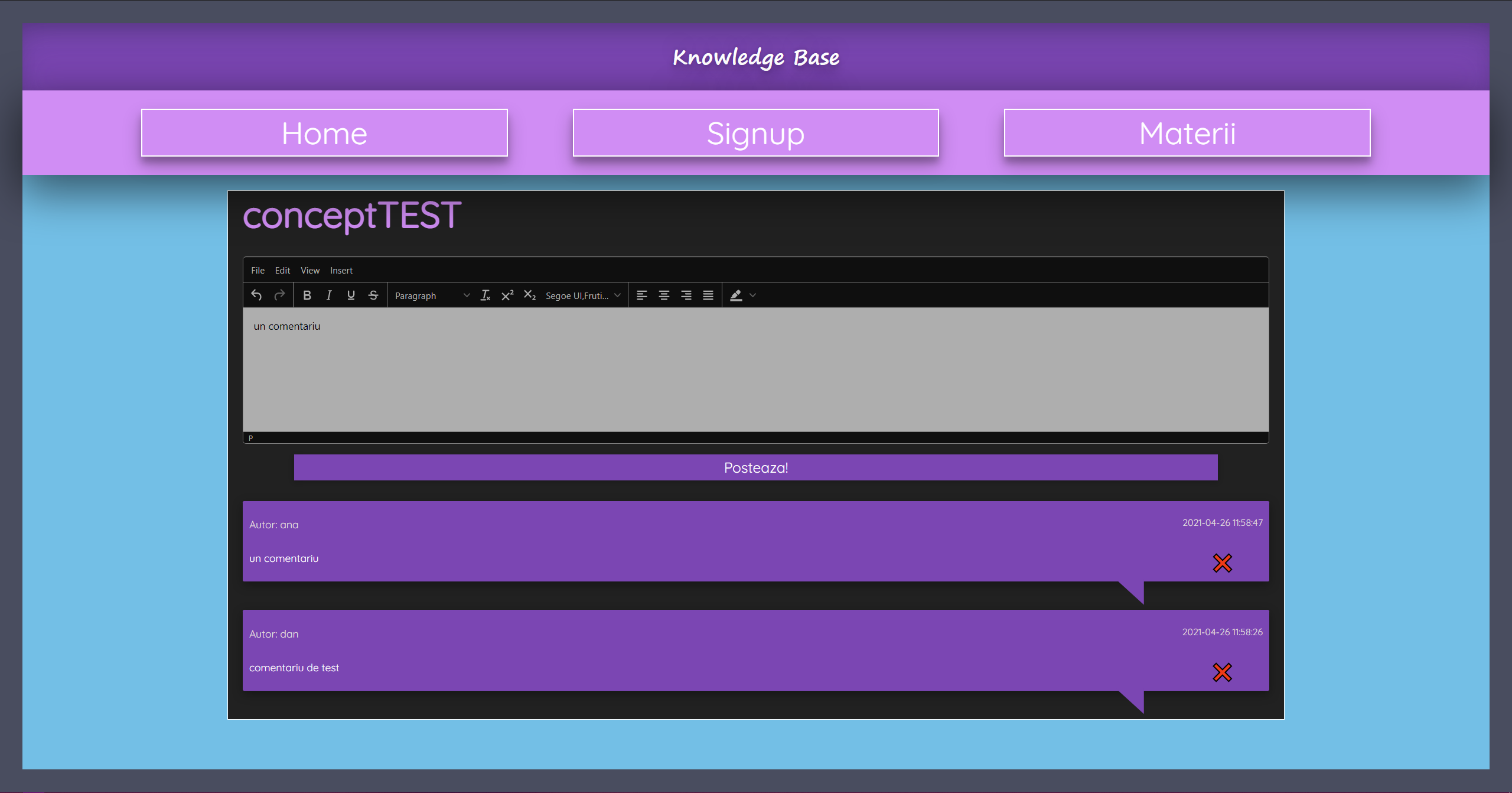
Odată ajunși pe aceasta pagina utilizatorii vor putea scrie comentarii prin intermediul editorului de tip WYSIWYG(what you see is what you get).

Pentru a posta comentariul proaspăt scris aceștia vor putea apasă butonul “Postează!” aflat în partea inferioara a editorului de text.

Profesorii vor avea și opțiunea de a șterge din comentariile postate în timp ce studenții vor putea doar posta comentarii noi și vizualiza comentariile deja postate.

Fiecare comentariu în parte va avea afișat numele autorului, ora la care a fost postat cat și conținutul sau.

Apăsarea X-ului roșu aflat în partea dreapta a comentariului de către profesori va duce la ștergerea acestuia. În cazul studenților acest X roșu nu va fi prezent.



*Comentarii 1 Privire de ansamblu asupra paginii*

Doar utilizatorii logati vor putea adăuga un comentariu. Dacă un utilizator nelogat va încerca să posteze un nou comentariu acesta va primi o eroare. Totuși, utilizatorii nelogati vor putea vizualiza comentariile postate.

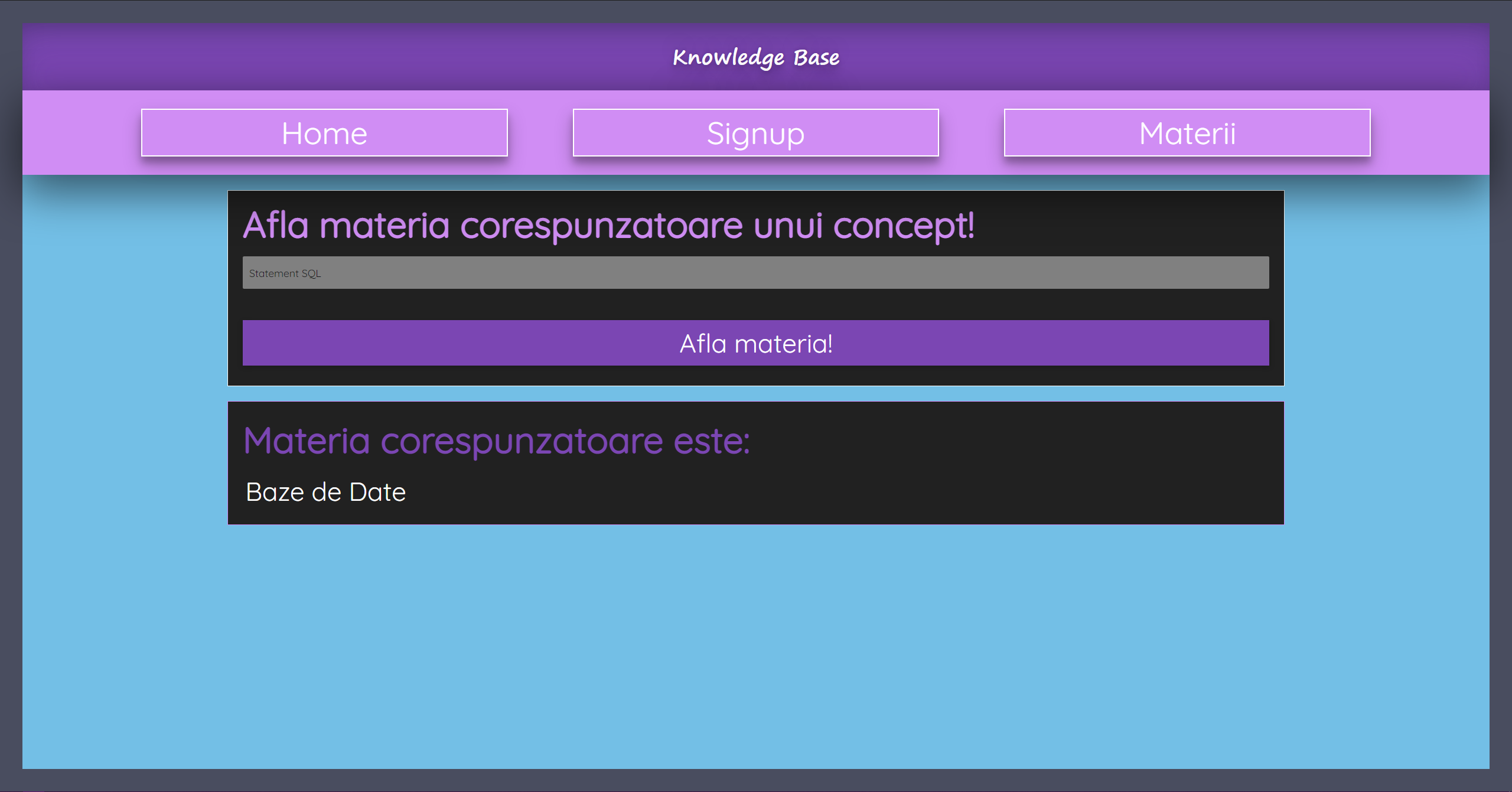


*Comentarii 2 Eroarea primita de un utilizator nelogat când încearcă să posteze un comentariu*

## Aflarea materiei corespunzătoare unui concept anume

In cazul în care un utilizator își aduce aminte numele unui concept studiat în facultate dar nu mai știe exact materia în cadrul căreia l-a studiat, aplicația ii va pune la dispoziție opțiunea de a o face.

Pentru a ajunge pe aceasta pagină, utilizatorul trebuie să apese pe butonul “Materii” din meniul principal al aplicației iar apoi pe butonul “Afla materia corespunzătoare”.



*Aflarea materiei corespunzătoare 1 Privire de ansamblu asupra paginii*

Utilizatorul va avea la dispoziție un meniu drop-down cu autocomplete de unde va putea selecta numele conceptului pentru care dorește să afle materia corespunzătoare. Apoi, la apăsarea butonului “Afla materia!” materia corespunzătoare conceptului selectat va fi afisata.

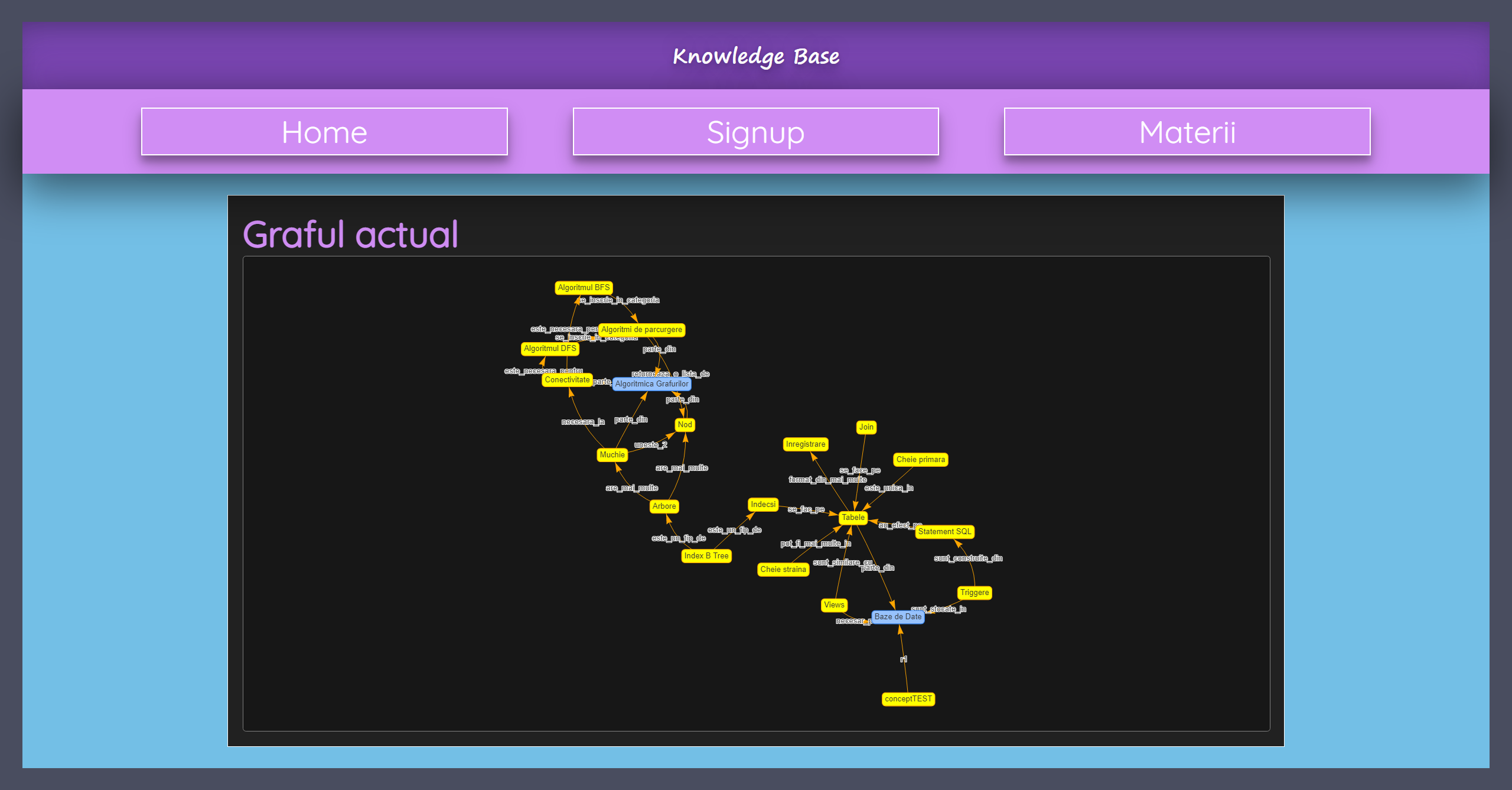
Dacă utilizatorul apasă numele materiei găsite, acesta va fi redirecționat către pagina corespunzătoare materiei, unde va putea vizualiza toate conceptele asociate respectivei materii.

## Vizualizarea grafului

Pentru a ajunge pe aceasta pagina, utilizatorul trebuie să apese pe butonul “Materii” din meniul principal al aplicației iar apoi pe butonul “Vizualizare graf”.

Odată ajuns aici, utilizatorul va avea opțiunea de a vizualiza graful stocat de aplicație pe partea de back-end.

El va putea face zoom și pan pe acest graf.



*Vizualizare graf 1 Privire de ansamblu*

## Vizualizarea statisticilor

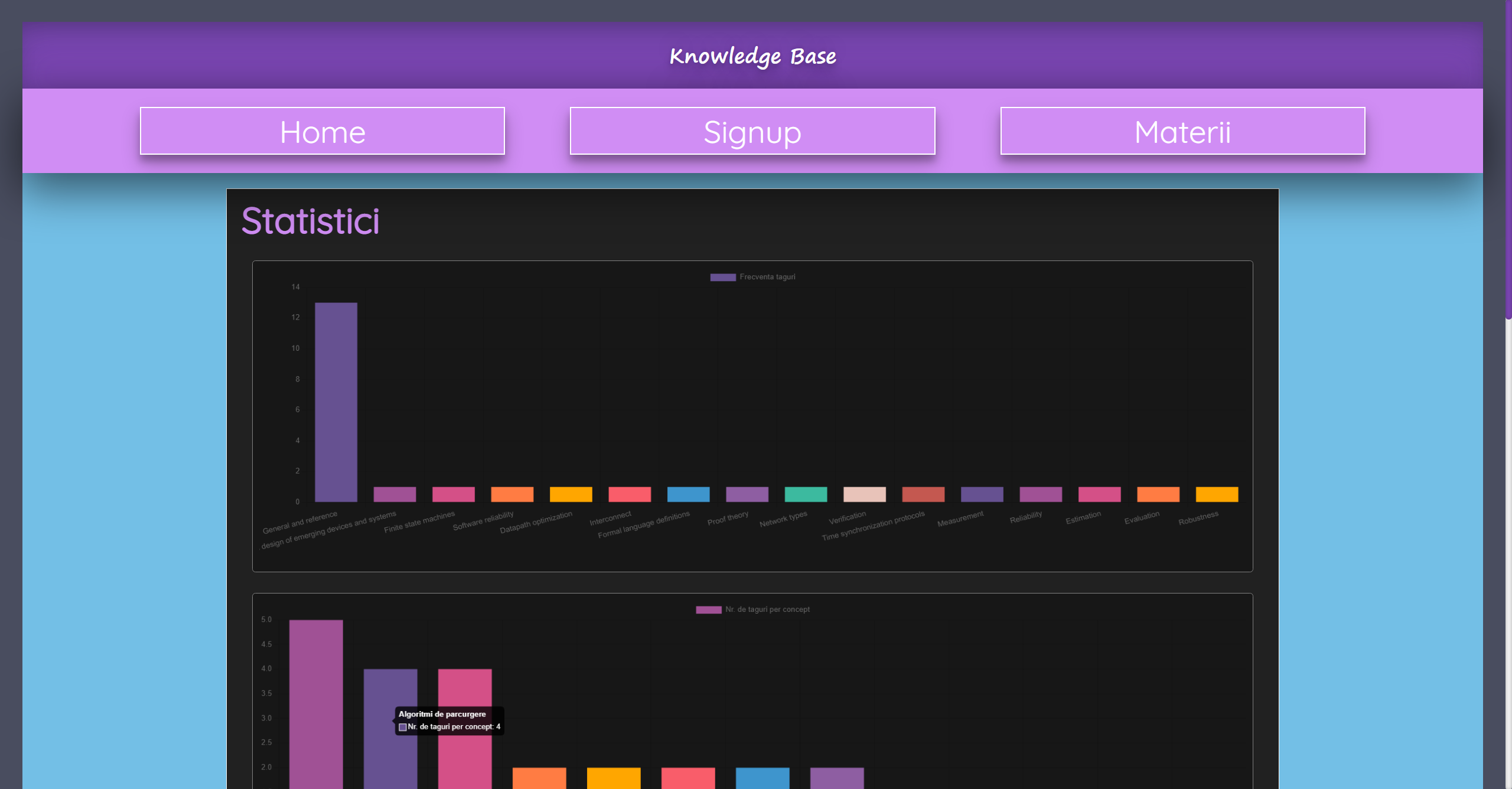
Prin intermediul acestei pagini utilizatorul va putea consulta anumite statistici.

Pentru a ajunge pe aceasta pagina, utilizatorul trebuie să apese pe butonul “Materii” din meniul principal al aplicației iar apoi pe butonul care arata ca un bar-chart sau pe butonul “Statistici” (varianta mobile).

Graficele afișate suporta atât funcționalitatea de pan cat și cea de zoom, lucruri foarte utile pentru a oferi o experiență plăcută utilizatorului în cazul în care acesta folosește un dispozitiv mobil.

Statisticile care pot fi vizualizate de către utilizator sunt:

* Frecventa tag-urilor, care sunt tag-urile cele mai folosite pentru a caracteriza concepte
* Conceptele cu cele mai multe tag-uri
* Nr. de comentarii per concept
* Nr. de relații per concept
* Nr. de comentarii per utilizator



*Vizualizare statistici 1 Privire de ansamblu*

# Concluzii și direcții viitoare

Consider că aplicația Knowledge Base este o aplicație foarte utilă datorită funcționalităților pe care le oferă atât studenților cat și profesorilor: adăugarea unor concepte, stabilirea relațiilor intre ele ori intre ele și materii, adăugarea de comentarii, taguri asociate diverselor concepte, vizualizarea tuturor acestor date, filtrare după diverse tag-uri, etc.

Pe viitor, aplicația ar putea fi extinsă ca nivel de funcționalitate și de asemenea ar putea fi transformată într-o aplicație ce să poată fi instalată pe mai multe platforme:Windows, Android,IOS, etc.

Ca funcționalități, ar putea fi adăugate următoarele:

* Crearea automata de relații intre anumite concepte
* Concatenarea automata a conceptelor extrem de similare dar cu denumiri puțin diferite

# Bibliografie

<https://dl.acm.org/ccs>

<http://clgiles.ist.psu.edu/pubs/AAAI2017-course.pdf>

<https://en.wikipedia.org/wiki/Neo4j>

<https://neo4j.com/developer/graph-database/#:~:text=Neo4j%20is%20an%20open%2Dsource,transactional%20backend%20for%20your%20applications.&text=Neo4j%20is%20referred%20to%20as,down%20to%20the%20storage%20level>.