

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Database *Guru***

***Platformă de învățare a limbajului SQL***

propusă de

***Alexandru Roșca***

**Sesiunea:** *iulie, 2019*

Coordonator științific

**Conf. Dr. Anca Vitcu**

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

# **Database *Guru***

***Platformă de învățare a limbajului SQL***

***Alexandru Roșca***

**Sesiunea:** *iulie, 2019*

Coordonator științific

***Conf. Dr. Anca Vitcu***

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_

Data \_\_\_\_\_ Semnătura \_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemnatul(a) .....

domiciliul în .....

născut(ă) la data de ....., identificat prin CNP .....,

absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de

..... specializarea ....., promoția

....., declar pe propria răspundere, cunoscând consecințele falsului în

declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr.

1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_elaborată sub îndrumarea dl. / d-na

\_\_\_\_\_, 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 în vederea facilitării falsificării 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.

Data azi, .....

Semnătură student .....

## ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, *Alexandru Roșca*.

Încheierea acestui acord este necesară din următoarele motive:

Doresc să continui dezvoltarea aplicației pentru continuarea studiilor avansate.

Iași,

Decan Adrian Iftene

---

(semnătura în original)

Absolvent *Alexandru Roșca*

---

(semnătura în original)

# Cuprins

Introducere .....	1
Obiective generale.....	1
Motivație .....	1
Aplicații similare.....	1
Contribuții .....	2
Aplicabilitate .....	3
Structura lucrării.....	3
1 Structura aplicației .....	4
1.1 Structura proiectului .....	4
1.2 Tehnologii utilizate .....	5
1.3 Arhitectura bazei de date a aplicației .....	6
1.4 Interfața .....	6
1.5 Încărcarea informațiilor.....	7
2 Module.....	8
2.1 Modulul <i>accounts</i> .....	8
2.1.1 Descriere .....	8
2.1.2 Modelarea bazei de date: .....	8
2.2 Modulul <i>exercises</i> .....	10
2.2.1 Descriere .....	10
2.2.2 Modelarea bazei de date: .....	11
2.2.3 Baza de date pentru testarea exercițiilor .....	12
3 Conturi .....	13
3.1 Tipuri.....	13
3.2 Utilizatorul neînregistrat .....	13

3.3	Utilizatorul normal .....	13
3.4	Profesorul .....	14
3.5	Adminul.....	14
3.5.1	Gestiunea cererilor pentru drepturi de profesori .....	14
3.5.2	Gestiunea profesorilor.....	15
3.6	Administratorul .....	15
4	Interacțiunea utilizatorului cu aplicația .....	16
4.1	Crearea unui cont .....	16
4.2	Profil.....	17
4.2.1	Informațiile prezente .....	17
4.3	Cererea pentru drepturi de profesor/ admin .....	18
4.4	Exerciții .....	18
4.4.1	Cum răspund la întrebare? .....	19
4.4.2	Istoric și verificarea răspunsului .....	22
4.4.3	Baremul și punctarea utilizatorului .....	23
4.4.4	Cum postez o întrebare ca profesor?.....	24
4.4.5	Gestionarea întrebărilor postate .....	26
4.4.6	Statistici.....	27
	Concluzii .....	29
	Dificultăți întâlnite .....	29
	Dezvoltare ulterioară .....	30
	Bibliografie .....	31
	Cărți.....	31
	<i>Link-uri</i> .....	31

# Introducere

## Obiective generale

Aplicația „*Database Guru*” a fost realizată din dorința de a ajuta studenții sau oamenii pasionați în învățarea limbajului *SQL*, dar și testarea cunoștințelor deja dobândite despre acesta, într-un mod cât mai interactiv, interesant și aparte. Aplicația poate fi utilizată atât de studenți, cât și de profesori sau alte persoane care sunt interesate de acest domeniu. Astfel, utilizatorii își pot îmbogăți bagajul de cunoștințe, iar profesorii pot găsi nivelul optim de dificultate în crearea unei întrebări, cât și în modul cel mai atractiv de a o formula.

## Motivație

Ceea ce am observat în decursul anilor la ceilalți studenți și mai ales la mine, a fost capacitatea scăzută de concentrare față de lucrurile ce nu reprezentau un interes nici prin aspect (*design*), cât nici prin metodele de interacțiune abordate.

Este foarte important pentru un tânăr, cu o capacitate crescută de învățare și memorare, ca ceea ce reprezintă un punct de interes să fie abordat în diferite moduri astfel încât să îi stârnească curiozitatea și atenția. Procesul de învățare trebuie să fie unul simplu, interactiv, atractiv, fără presiuni asupra studentului și cu substraturi motivaționale.

Plecând de la aceste caracteristici și însușiri, am creat aplicația „*Database Guru*”, prin care am reușit să îmbin tot ceea ce mi-am propus, astfel încât învățarea să se facă într-un mod plăcut, relaxant și deschis.

Profesorii, studenții, chiar și firmele care au diferite *training*-uri, pot folosi această aplicație care combină utilul cu plăcutul și mai ales care le oferă tuturor șansa de a acumula informațiile necesare și de a crește încrederea în cunoștințele proprii.

## Aplicații similare

În primă instanță m-a atras ideea aplicației de pe „<https://flukeout.github.io/>”. Aceasta este o platformă de învățare a limbajului *CSS*, printr-o modalitate mai inedită și anume prin reprezentarea

întrebărilor sub formă de text dar și sub formă de obiect 2D cu animație. Inițial am fost atras foarte mult de concept, dar și de *design*-ul minimalist ce prezintă nuanțe închise la culoare.

O altă aplicație deja existentă cu o idee similară este cea de la obiectul Baze de Date din anul al II-lea, de la Facultatea de Informatică din Iași. Aceasta are rolul de a testa cunoștințele studenților prin răspunderea la întrebări și posibilitatea de a crea chiar ei altele noi pentru colegii lor.

## Contribuții

Deși ideea este inspirată din cele două aplicații prezentate anterior, contribuția personală a constat în următoarele îmbunătățiri:

- ✓ Aplicația *Database Guru* are atât opțiune de rezolvare a exercițiilor fără înregistrare, cât și posibilitatea de a crea un cont și de a avea progresul salvat indiferent de *device*-ul de pe care se conectează utilizatorul;
- ✓ Nu există un număr limitat de întrebări și acestea nu sunt *hard coded*;
- ✓ Există o delimitare a nivelelor de acces asupra conturilor;
- ✓ Există o modalitate de a-ți verifica răspunsul și de a-l compara cu cel corect;
- ✓ Fiecare cont are un profil care poate fi editat;
- ✓ Există o pagină de istoric în cazul în care utilizatorul dorește să își revadă răspunsul pentru reîmprospătarea memoriei atunci când are nevoie;
- ✓ Profesorii pot să își personalizeze întrebările având posibilitatea de a modifica baremul de notare după necesitățile fiecăruia;
- ✓ Profesorii au la dispoziție statistici pentru fiecare întrebare individuală, dar și pe toate întrebările postate, pentru o privire de ansamblu;
- ✓ Profesorii pot ajuta utilizatorii, oferindu-le *hint*-uri;
- ✓ Există un clasament realizat în funcție de punctajul obținut în urma rezolvării exercițiilor;
- ✓ Baza de date pe care rulează exercițiile este preluată și afișată dinamic, pentru a ajuta utilizatorii în a-și forma o privire de ansamblu asupra acesteia.



## Aplicabilitate

Încă de la început am pornit cu ideea de a crea o aplicație care mai poate fi dezvoltată ulterior și care poate fi implementată atât în instituții precum licee și facultăți cât și în domeniul privat pentru *training*-uri sau *online* pentru publicul pasionat de subiectul abordat.

De asemenea, aplicația poate fi extinsă foarte mult și ușor datorită modularizării sale, astfel extinzându-se și aria de interes a publicului față de aceasta dar și categoriile de cunoștințe ce pot fi evaluate sau chiar aprofundate.

## Structura lucrării

Structura aplicației – prezentarea în ansamblu a aplicației, a structurii ei și un mic rezumat despre felul cum funcționează

Module – prezentarea structurii bazei de date a fiecărui modul și explicarea pe scurt a câmpurilor din fiecare tabel

Conturi – enumerarea tipurilor de conturi, descrierea acestora și nivelele lor de acces a informațiilor

Interacțiunea utilizatorului cu aplicația – o parcurgere a aplicației ce conține acțiunile pe care un utilizator le poate face și mici detalii de implementare a acestora

# 1 Structura aplicației

## 1.1 Structura proiectului

Datorită modului organizat de lucru în *framework*-ul *Django*, aplicația a fost împărțită în două module: *accounts* și *exercises* (Fig. 1 – Structura proiectului). Inițial aceste module au fost proiectate să funcționeze separat, în scopul de a fi cât mai ușor de modificat sau eliminat din aplicație. Totuși, acest lucru nu a fost posibil datorită dependenței modulului *exercises* față de *accounts*, ținând cont de faptul că este imposibil ca un exercițiu să fie rezolvat de către un cont inexistent. În schimb, modulul *accounts* este total independent față de celălalt modul, utilizatorul având posibilitatea de a explora restul *site*-ului în cazul în care modulul *exercises* este șters.

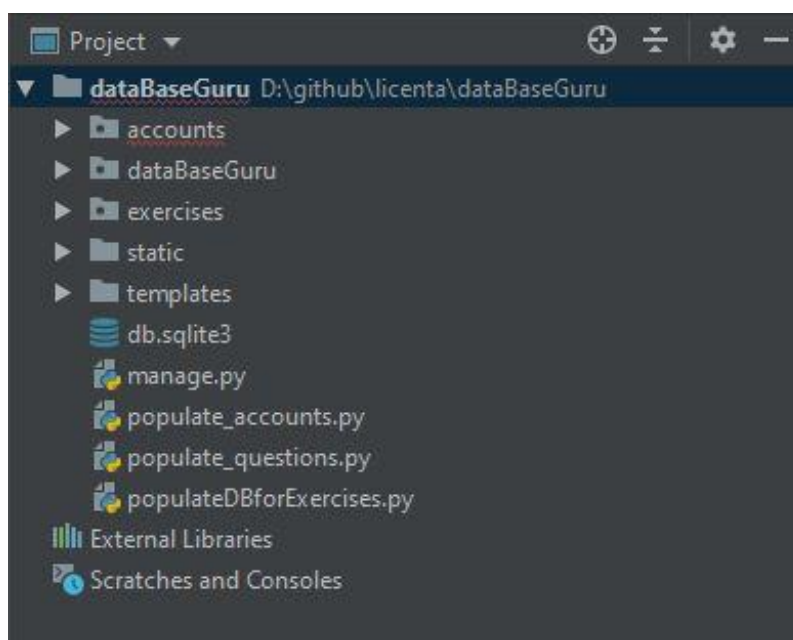


Fig. 1 – Structura proiectului

De asemenea, la finalul proiectului se află trei *script*-uri care sunt folosite pentru a popula bazele de date pentru cele două module și cea pentru a testa exercițiile.

## 1.2 Tehnologii utilizate

Aplicația a fost în principal proiectată cu ajutorul *framework*-ului *Django*, un *framework* pentru limbajul de programare *Python* (am folosit versiunea 3.x).

Auxiliar, am folosit următoarele:

1. Pentru interfață:

- HTML
- CSS
- *Bootstrap* – utilizat în mare parte a *design*-ului, deoarece ajută la crearea unui *site* cât mai *responsive*, dar totodată ajută și la încadrarea mai ușoară și precisă a obiectelor în pagină
- *JQuery* – predominant pentru funcționalitatea butoanelor dar nu numai
- *Font Awesome* – preluarea iconițelor sub formă de text (mai exact *<i>*), util pentru manipularea lor: poziție, culoare, *padding*
- *Animate.CSS* – pentru animații
- *Chart.js* – pentru crearea de statistici într-un mod cât mai plăcut vizual, dar totodată și mai eficient

2. Pentru *server*:

- *Django* – *web development framework* pentru *Python*

3. Pentru comunicarea interfață – *server*:

- *AJAX*

## 1.3 Arhitectura bazei de date a aplicației

Baza de date este formată din opt tabele, șase sunt din modulul *accounts* și două sunt din modulul *exercises* (Fig. 2 – Arhitectura bazei de date). Pentru fiecare tabel, dacă nu este specificat un *primary key*, *Django* pune automat un *id* de tip *integer* ca și *primary key*.

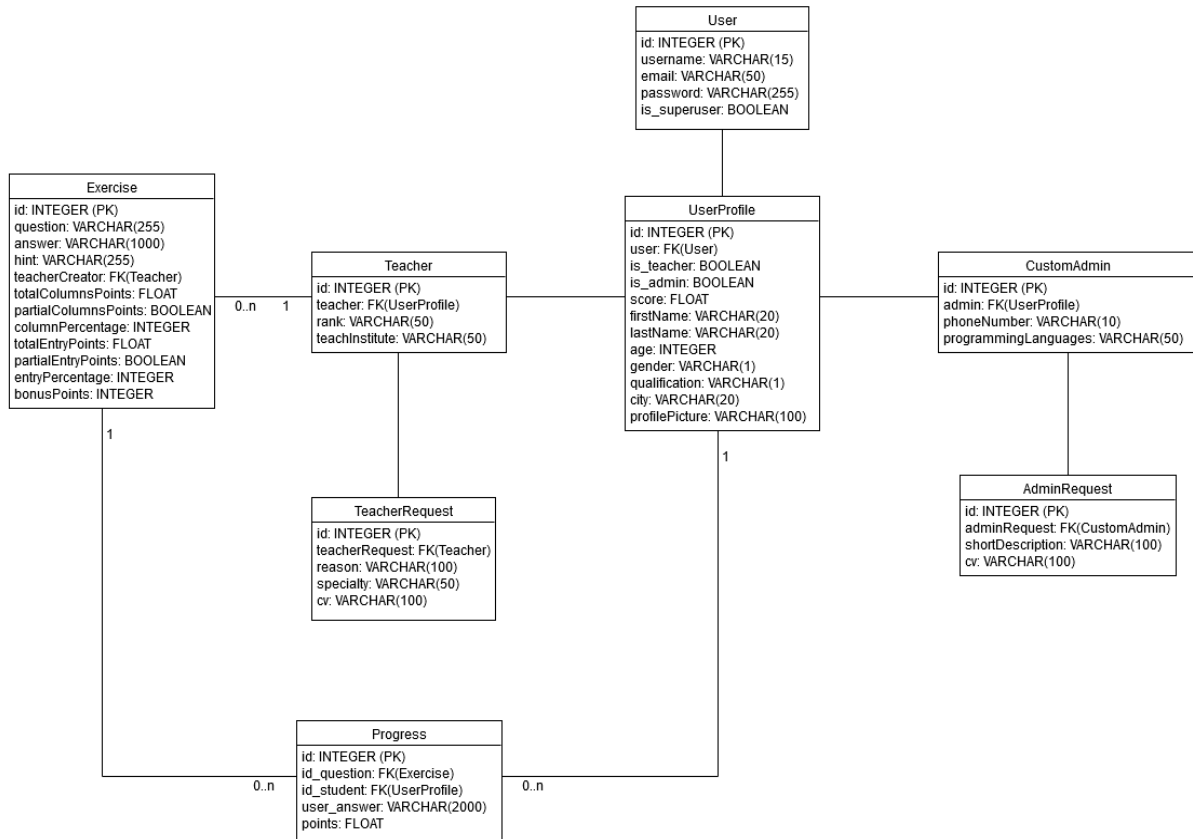


Fig. 2 – Arhitectura bazei de date

## 1.4 Interfața

În *Django*, o pagină *html* este denumită *template*, aceasta fiind un suport vizual pentru informațiile *view*-ului care o încarcă.

Pentru optimizare, am folosit moștenirea oferită de *Django* pentru *template*-uri (Fig. 3 - Extends).

```
{% extends 'dataBaseGuru/baseDesign.html' %}
```

Fig. 3 - Extends

Acest tag "extends" oferă posibilitatea de a extinde un *template* în alt *template*. Mai concret, majoritatea paginilor *html* din aplicație extind pagina "baseDesign.html". În această pagină se află toate elementele repetitive de pe *site* care trebuie încărcate mereu: *navbar*-ul, *background*-ul, culorile, *CDN*-urile pentru *Bootstrap*, *Font Awesome* și alte *script*-uri. Astfel, se îmbunătățește calitatea codului, având în vedere că nu trebuie scris același lucru pe fiecare pagină, se facilitează înțelegerea acestuia și crește viteza de încărcare a paginilor.

## 1.5 Încărcarea informațiilor

Pentru a încărca informațiile, *Django* folosește *view*-uri, acestea fiind funcții sau clase. Prin intermediul acestora sunt încărcate atât paginile cât și informațiile lor. În fiecare modul există un fișier "views.py" ce conține toate funcțiile și clasele modulului respectiv, astfel existând o organizare mult mai bună a proiectului.

De asemenea, *Django* are predefinite clase care pot fi adaptate la nevoile programatorului, folosite chiar și în aplicație, precum:

- *ListView* – oferă o listă de obiecte în funcție de parametrul *model* și de funcția *get\_queryset*
- *DetailView* – în general se transmite către *view* un *pk* și acesta oferă toate informațiile din baza de date a elementului cu *pk*-ul respectiv.
- *TemplateView* – încarcă *template*-ul specificat în parametrul *templated\_name*
- *LoginRequiredMixin* – aceasta se poate pune ca și parametru, obligatoriu primul, la orice *view* bazat pe o clasă și testează dacă utilizatorul care a făcut *request* la *view*-ul respectiv este autentificat sau nu. Dacă este autentificat, *view*-ul se încarcă, dacă nu, utilizatorul este redirecționat către pagina de *login*.

## 2 Module

### 2.1 Modulul *accounts*

#### 2.1.1 Descriere

Acest modul se ocupă de partea administrativă a aplicației: *login*-ul, *logout*-ul, crearea și ștergerea conturilor și gestionarea permisiunilor acestora.

#### 2.1.2 Modelarea bazei de date:

Baza de date a acestui modul este formată din 6 tabele (Fig. 4): *User*, *UserProfile*, *Teacher*, *CustomAdmin*, *TeacherRequest*, *AdminRequest*

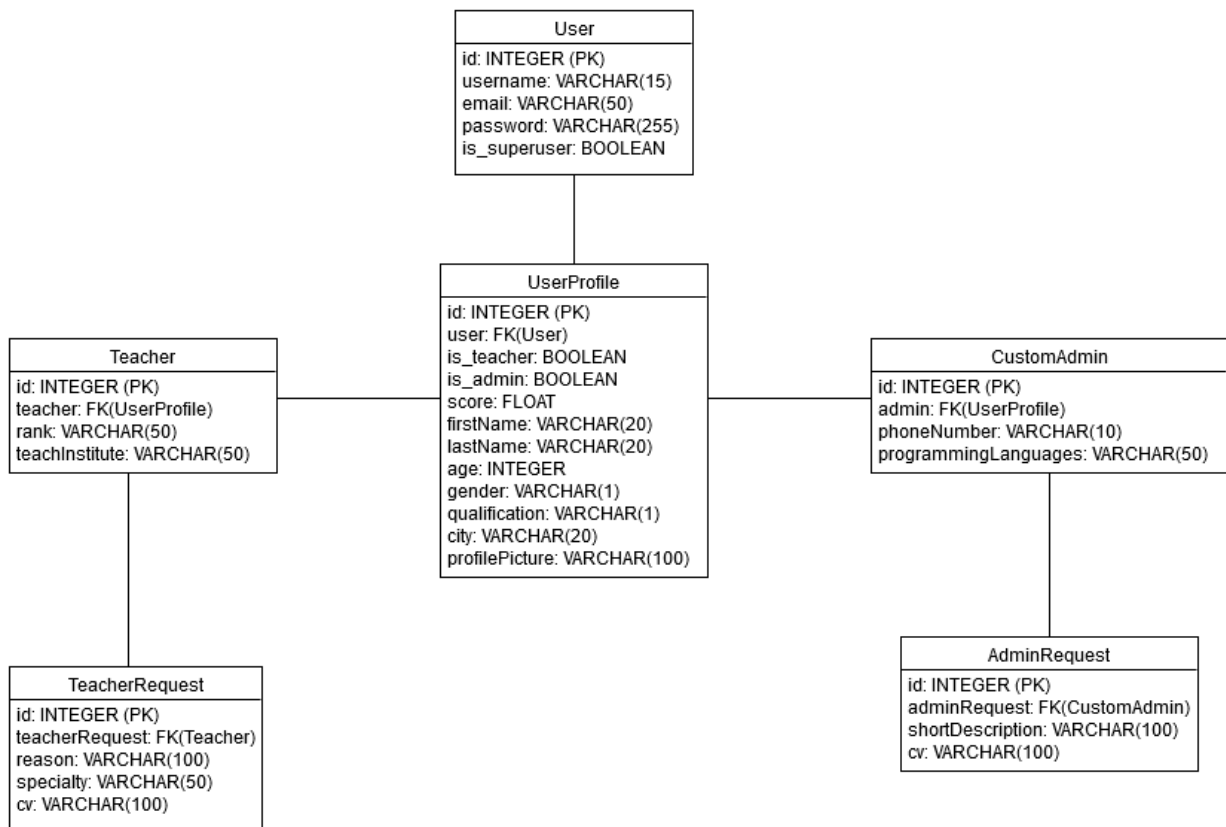


Fig. 4 – Accounts – Modelarea bazei de date

*Django* are predefinit tabelul *User* cu mai multe câmpuri, dintre care cele preluate sunt:

- *id* – *primary key-ul* generat automat
- *username* – ales de utilizator la înregistrare
- *password* – *Django* nu salvează niciodată parola așa cum o scrie utilizatorul, el generează un *hash* cu algoritmul *SHA256*
- *is\_superuser* – un *boolean* care are valoarea 1 când *user-ul* este administrator și 0 în caz contrar

Modelul *UserProfile* extinde modelul *User*, oferind:

- *id* – *primary key-ul* generat automat
- *user* – *foreign key* pentru modelul *User*
- *is\_teacher* – *boolean* care are valoarea 1 atunci când utilizatorul are drepturi de profesor sau valoarea 0 în caz contrar
- *is\_admin* – *boolean* care are valoarea 1 atunci când utilizatorul are drepturi de admin sau valoarea 0 în caz contrar
- *score* – coloana unde se înregistrează punctajul acumulat de la toate întrebările
- *firstName*
- *lastName*
- *age*
- *gender* – este reprezentat de un *varchar* de lungime maximă 1 care poate lua trei valori: N – *Neutral*; M – *Male*; F – *Female*
- *qualification* – este reprezentat de un *varchar* de lungime maximă 1 care poate lua patru valori: N – *Nothing*; P – *Primary School*; H – *High School*; U – *University*
- *city*
- *profilePicture*

Modelul *Teacher* extinde modelul *UserProfile*, oferind:

- *id* – *primary key-ul* generat automat
- *teacher* – *foreign key* pentru modelul *UserProfile*
- *rank* – gradul pe care îl are profesorul în cadrul de învățământ
- *teachInstitute* – locul unde predă profesorul

Modelul *TeacherRequest* extinde modelul *Teacher*, oferind:

- *id* – *primary key*-ul generat automat
- *teacherRequest* – *foreign key* pentru modelul *Teacher*
- *reason* – motivul pentru care utilizatorul dorește să aibă drepturi de profesor
- *specialty* – aria de specialitate al utilizatorului
- *cv*

Modelul *CustomAdmin* extinde modelul *UserProfile*, oferind:

- *id* – *primary key*-ul generat automat
- *admin* – *foreign key* pentru modelul *UserProfile*
- *phoneNumber*
- *programmingLanguages* – câteva limbaje de programare pe care *admin*-ul le stăpânește

Modelul *AdminRequest* extinde modelul *CustomAdmin*, oferind:

- *id* – *primary key*-ul generat automat
- *adminRequest* – *foreign key* pentru modelul *CustomAdmin*
- *shortDescription* – o scurtă descriere a personalității utilizatorului
- *cv*

## 2.2 Modulul *exercises*

### 2.2.1 Descriere

*Exercises* este modulul care se ocupă atât cu înregistrarea exercițiilor, cât și cu salvarea progresului fiecărui utilizator. De asemenea, tot acest modul conține și baza de date pe care se testează exercițiile.



## 2.2.2 Modelarea bazei de date:

Baza de date a acestui modul este formată din două tabele: *Exercise* și *Progress* (Fig. 5).



Fig. 5 – Exercises – Modelarea bazei de date

Modelul *Exercise* conține:

- *id* – primary key generat automat
- *question* – cerința întrebării
- *answer* – răspunsul corect al întrebării
- *hint* – o sugestie pentru cel ce rezolvă, dacă este nevoie
- *teacherCreator* – foreign key pentru modelul *Teacher*
- *totalColumnsPoints* – numărul total de puncte disponibile pentru selectarea tuturor coloanelor cerute
- *partialColumnsPoints* – boolean cu valoarea 1 în cazul în care profesorul dorește să puncteze parțial coloanele selectate de utilizator sau 0 în cazul contrar
- *columnPercentage* – dacă *partialColumnsPoints* are valoarea 1, atunci această opțiune se deblochează și numărul introdus reprezintă  $x\%$  din totalul de puncte puse disponibile pentru coloane
- *totalEntryPoints* – numărul total de puncte disponibile pentru selectarea tuturor înregistrărilor cerute
- *partialEntryPoints* – boolean cu valoarea 1 în cazul în care profesorul dorește să puncteze parțial înregistrările selectate de utilizator sau 0 în cazul contrar

- *entryPercentage* – dacă *partialEntryPoints* are valoarea 1, atunci această opțiune se deblochează și numărul introdus reprezintă  $x\%$  din totalul de puncte puse disponibile pentru înregistrări
- *bonusPoints* – puncte bonus în cazul în care profesorul dorește să le ofere utilizatorilor dacă înregistrările și coloanele au fost corecte și în ordine

Modelul *Progress* conține:

- *id* – *primary key* generat automat
- *id\_question* – *foreign key* pentru modelul *Exercises*
- *id\_student* – *foreign key* pentru modelul *UserProfile*
- *user\_answer* – răspunsul utilizatorului
- *points* – punctele obținute la acea întrebare

### 2.2.3 Baza de date pentru testarea exercițiilor

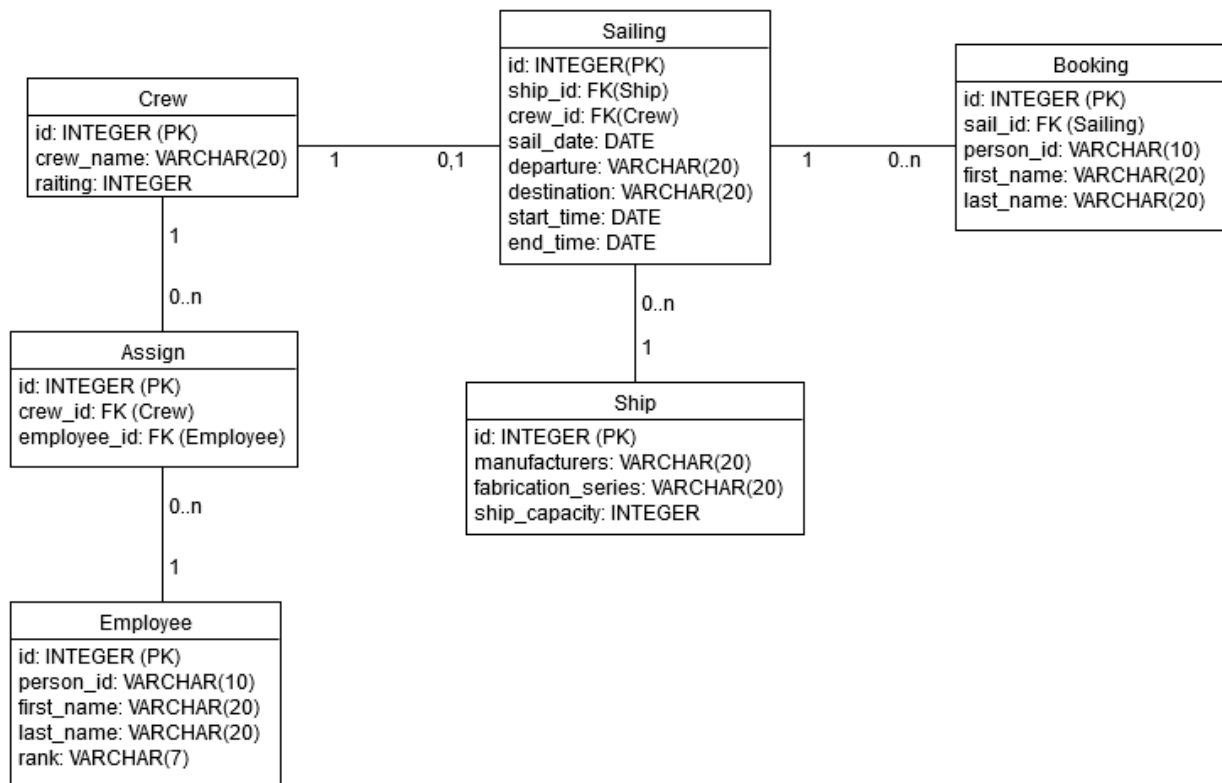


Fig. 6 – Baza de date pentru testarea exercițiilor

## 3 Conturi

### 3.1 Tipuri

Aplicația a fost construită să aibă patru nivele de acces, plus administratorul.

0. Utilizator neînregistrat
1. Utilizator normal
2. Profesor
3. Admin
4. Administrator

### 3.2 Utilizatorul neînregistrat

Toți utilizatorii care intră pe pagină și nu se autentifică intră automat în această categorie. Deși aceștia nu au niciun cont alocat, ei pot rezolva un număr limitat de întrebări, astfel această categorie este folosită pentru a oferi o imagine minimalistă a *user*-ului asupra aplicației, în ideea de a-i stârni interesul și de a-l motiva să își creeze un cont.

### 3.3 Utilizatorul normal

Utilizatorul normal moștenește *User*-ul *default* din *Django* și pe lângă attributele acestuia are în plus:

- un profil care este instanțiat automat la crearea contului
- acces la toate exercițiile disponibile
- acces la istoric
- acces la clasament
- acces la cererea pentru drepturi de admin
- acces la cererea pentru drepturi de profesor

### 3.4 Profesorul

Profesorul moștenește utilizatorul (în afară de accesul la cererea pentru drepturi de profesor), dar mai are în plus:

- posibilitatea de a posta exerciții
- posibilitatea de a-și gestiona propriile exerciții
- posibilitatea de a genera statistici

### 3.5 Adminul

Adminul moștenește utilizatorul (în afară de accesul la cererea pentru drepturi de admin), dar mai are în plus:

- gestionarea cererilor pentru drepturile de profesor (în afara de propria cerere)
- gestionarea conturilor
- ștergerea drepturilor de profesor a unui utilizator

#### 3.5.1 Gestiunea cererilor pentru drepturi de profesori

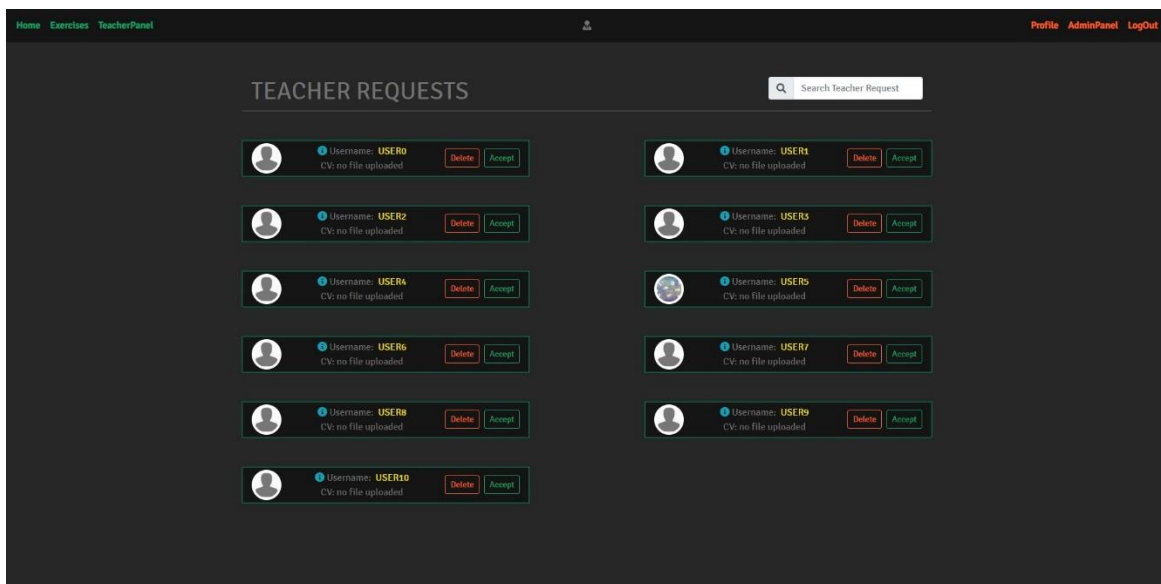


Fig. 7 –Cereri pentru dreptul de profesor

În această pagină (Fig. 7) *admin*-ul poate vedea cererile pentru dreptul de profesor. Fiecare card conține toate informațiile de care *admin*-ul are nevoie pentru a lua o decizie asupra cererii.

Pentru a avea o interfață cât mai simplă dar totuși completă, simbolul albastru ”i” are un hover aplicat, astfel *admin*-ul poate vedea toate informațiile pe care utilizatorul le-a completat în formular.

### 3.5.2 Gestiunea profesorilor

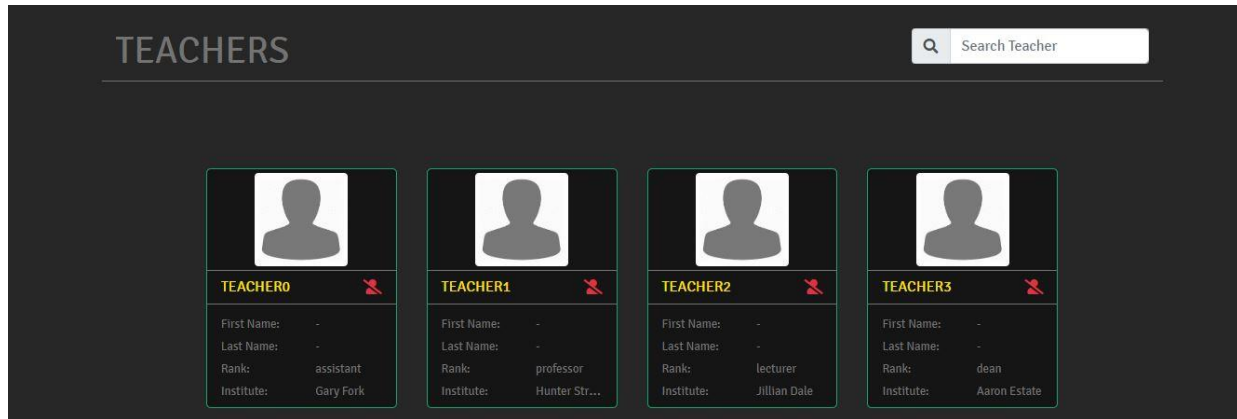


Fig. 8 – Gestiunea profesorilor

Menținând aceeași idee de „*simplitate completă*”, interfața pentru gestionarea profesorilor (Fig. 8) afișează informații minimaliste ale acestora, pentru a nu încărca prea mult pagina și pentru a nu fi prea obositoare pentru ochi și minte. Dacă adminul dorește să vadă mai multe informații legate de un anumit utilizator, poate da *click* pe numele acestuia și va fi redirecționat către pagina de profil.

### 3.6 Administratorul

Administratorul este contul cu cele mai mari privilegii, astfel încât singura modalitate de a-l crea este prin intermediul terminalului cu ajutorul comenzii `python manage.py createsuperuser`. Pe lângă drepturile deja existente din *Django*, acesta este totodată și singurul cont care poate gestiona din aplicație toate tipurile de conturi.

## 4 Interacțiunea utilizatorului cu aplicația

### 4.1 Crearea unui cont

Unul din principiile de bază a aplicației este să fie cât mai intuitivă și mai ușor de folosit pentru cât mai multă lume.

*Use case:* utilizatorul se conectează la aplicație. Acesta este întâmpinat de prima pagină a site-ului pentru *persoanele* neautentificate, unde sunt descriși pașii pe care îi poate urma.

Dacă utilizatorul decide să își creeze un cont, apăsând pe butonul *Register* va fi redirecționat pe pagina de înregistrare (Fig. 9).

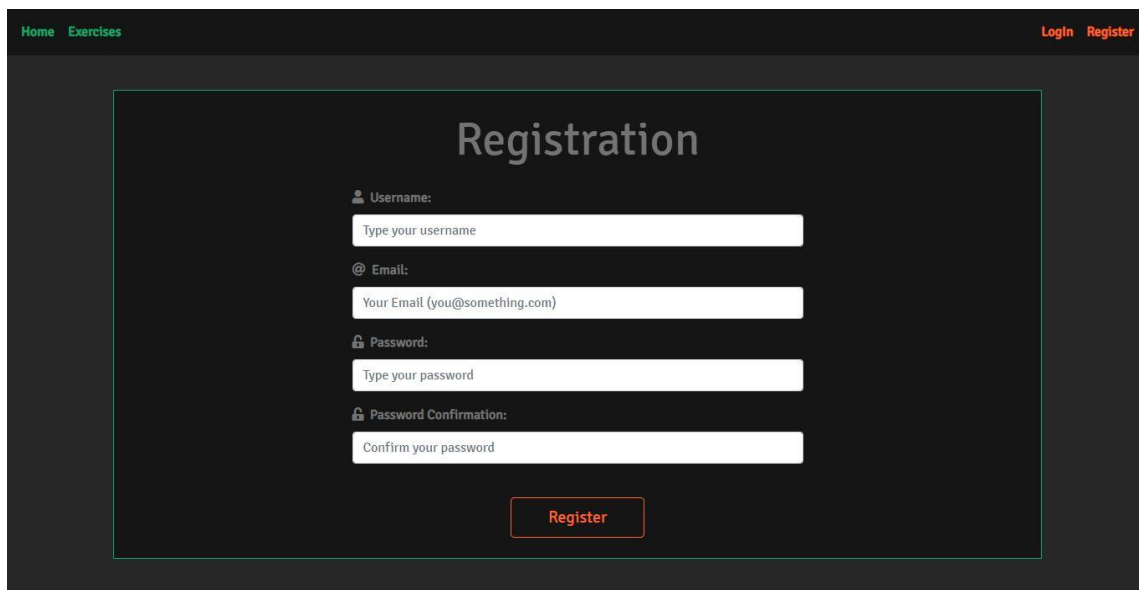
A screenshot of a web application's registration page. The page has a dark background. At the top, there are links for 'Home' and 'Exercises' on the left, and 'Login' and 'Register' on the right. The main heading is 'Registration'. Below it, there are four input fields: 'Username:' with a placeholder 'Type your username', '@ Email:' with a placeholder 'Your Email (you@something.com)', 'Password:' with a placeholder 'Type your password', and 'Password Confirmation:' with a placeholder 'Confirm your password'. Each field has a small icon to its left (person, email, and two locks respectively). At the bottom of the form is a red 'Register' button.

Fig. 9 – Înregistrare

Câmpuri prezente:

- *username*: necesar pentru autentificare
- *e-mail*
- *password*: pentru securitatea propriului cont, utilizatorul va trebui să respecte câteva cerințe minime, cum ar fi lungimea de cel puțin 8 caractere
- *password confirmation*: imediat ce termină de completat formularul și va apăsa pe butonul *Register*, dacă totul este în regulă, contul se va crea și utilizatorul va fi redirecționat pe pagina de autentificare, unde acesta poate să se autentifice cu *username*-ul și parola scrisă în formularul de înregistrare.

## 4.2 Profil

De fiecare dată când un utilizator își crează un cont pentru prima dată, acestuia i se instanțiază un profil cu toate câmpurile setate cu valorile implicite.

Profilul utilizatorului poate fi văzut de oricine și poate diferi de la cont la cont, deoarece în funcție de drepturile pe care le are acesta, se afișează mai multe sau mai puține informații și pentru fiecare tip de cont este atribuit o mini-imagine.

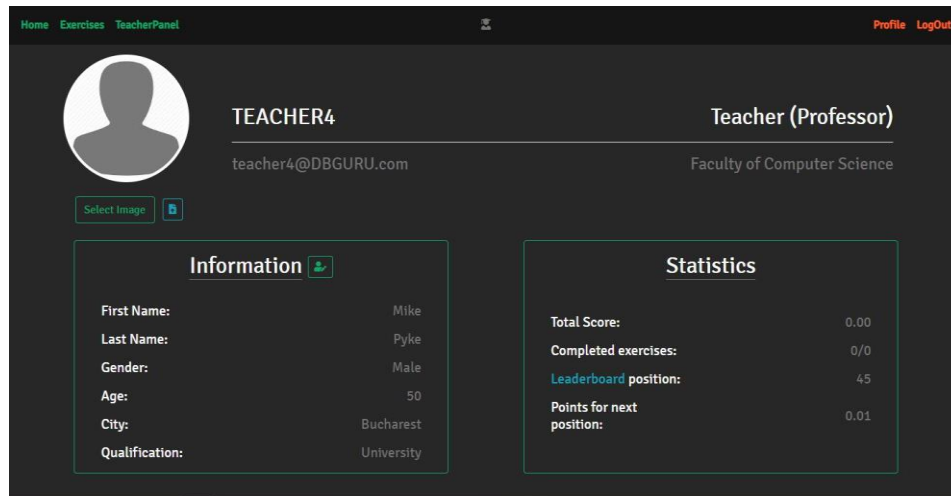


Fig. 10 – Profilul utilizatorului

### 4.2.1 Informațiile prezente

- Chenarul *Information*:
  - Conține informațiile generale despre persoana care a creat contul. Totuși, nu este obligatoriu ca totul să fie completat, toate câmpurile *editable* pot fi lăsate libere.
- Chenarul *Statistics*:
  - Conține informații legate de progresul utilizatorului în rezolvarea exercițiilor:
    - *Total Score*: după fiecare răspuns dat la un exercițiu, punctele acumulate se adună la numărul total de puncte
    - *Completed exercises*: numărul de exerciții rezolvate/ numărul total de exerciții
    - *Leaderboard position*: locul ocupat în clasamentul general
    - *Points for next position*: numărul de puncte necesare pentru a trece la următorul loc în clasament

### 4.3 Cererea pentru drepturi de profesor/ admin

Orice utilizator poate face o cerere pentru drepturi de profesor sau admin, cu două condiții:

- să nu aibă deja drepturile respective
- să nu aibă deja o cerere activă

Pentru a găsi *link*-urile cu respectivele cereri, utilizatorul trebuie doar să apese pe iconița de pe bara de meniu (Fig. 11), aceea care reprezintă tipul de cont pe care îl are.

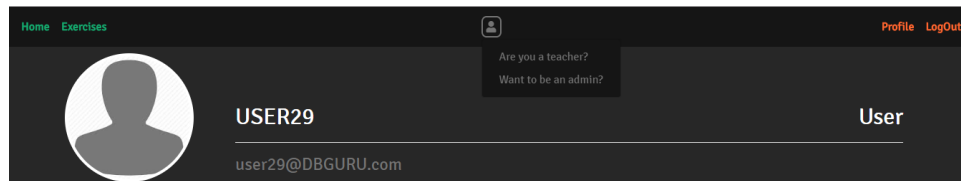


Fig. 11 – Cereri pentru drepturi de profesor/ admin

### 4.4 Exerciții

Indiferent de drepturile contului, fiecare utilizator poate vedea în bara de meniu un *dropdown menu* numit *Exercises*.

Această pagină prezintă utilizatorului toată lista de exerciții disponibilă în momentul respectiv, inclusiv numărul total de puncte acumulate în urma rezolvării exercițiilor.

Fiecare întrebare are un card (Fig. 12) al ei în care sunt afișate informații din baza de date precum: enunțul exercițiului, creatorul său, un simbol reprezentativ ce reprezintă starea întrebării în concordanță cu utilizatorul curent (dacă a răspuns sau nu la ea) și punctajul acumulat în urma rezolvării ei (sau *0 pts.* în caz contrar).

EXERCISES		
Total Score: 154.00 pts.		
1	Select first name and last name of all the captains who were in charge on August 25	150.00 pts. Solved: ✓ <small>Created by: teacher4</small>
2	Select all the ships which set sailed in August 2019	0 pts. Solved: ✗ <small>Created by: teacher4</small>
3	Show how many employees are in every crew available	4.00 pts. Solved: ✓ <small>Created by: teacher7</small>

Fig. 12 – Listă exerciții



### 4.4.1 Cum răspund la întrebare?

Fiecare card din lista de întrebări este de fapt un *link*, care odată apăsă, redirecționează utilizatorul pe pagina unde poate să răspundă la aceasta.

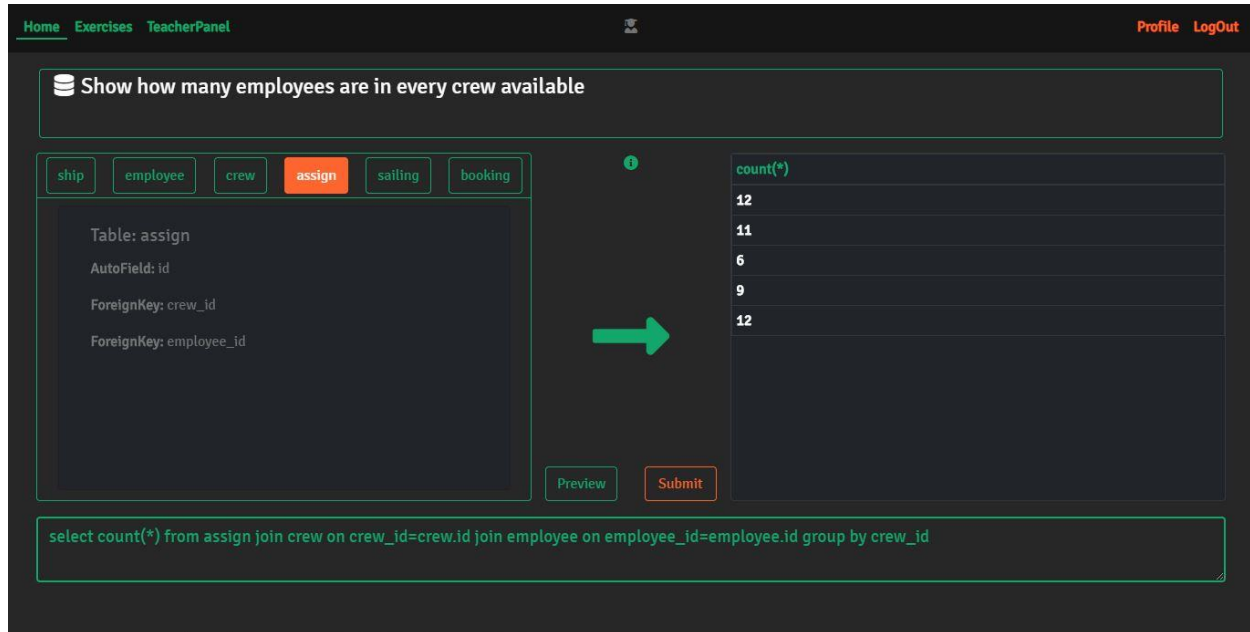


Fig. 13 – Pagină pentru răspuns la întrebare

Elemente principale prezente în pagină:

- Cerința exercițiului
- Chenarul de răspuns
- Chenarul cu detalii despre baza de date
- Chenarul cu *output*-ul utilizatorului
- Butoanele de control

1. Cerința exercițiului este preluată din baza de date
2. Chenarul de răspuns este un *textarea* care poate fi redimensionat după nevoile și preferința utilizatorului
3. Chenarul cu detalii despre baza de date

Este generat automat după modelul bazei de date pentru testarea exercițiilor (Fig. 6 – Baza de date pentru testarea exercițiilor). Numele tabelor sunt sub formă de butoane, în urma apăsării lor,

utilizatorul având acces la informațiile necesare (tipul *field*-ului și denumirea acestuia) formulării unui răspuns.

Chiar dacă baza de date pentru testare este deocamdată fixă, fără posibilitatea de a o modifica din interfață, *script*-ul care afișează chenarul cu informații despre aceasta nu este dependent de modelul bazei de date, acesta funcționând indiferent de model (Fig. 12).

```
def get_models_info():
    tableListName = []
    tableFields = []
    allTableFieldsDictList = []

    excModels = apps.get_app_config('exercises').get_models()
    for table in excModels:
        if 'Exercise' not in str(table) and 'Progress' not in str(table):
            tableListName.append(table._meta.db_table)
            tableFields.append(table._meta.get_fields())

    for allFields in range(0, len(tableFields)):
        fieldList = re.findall(r'\.fields\.related\.'*([A-Z][a-z]*): ([A-Za-z_-]*)>', str(tableFields[allFields]))
        tupleFieldList = ()
        listOffields = []
        tableFieldsDict = {}
        for field in fieldList:
            tupleFieldList = (field[1], field[3])
            listOffields.append(tupleFieldList)
        tableFieldsDict = {tableListName[allFields]: listOffields}
        allTableFieldsDictList.append(tableFieldsDict)

    return allTableFieldsDictList
```

Fig. 14 - Script pentru preluarea informațiilor din model

- se preiau informațiile modulului *exercises*
- se parcurge lista de obiecte și se salvează adițional alte două liste:
  - *tableListNames*: listă cu denumirea tabelelor
  - *tableFields*: listă cu liste ce conțin informații despre *field*-urile tabelelor
- se parcurge lista *tableFields* ce conține informații despre *field*-uri și cu ajutorul *regex*-ului se selectează doar informațiile utile
- pentru fiecare *field* se creează un *tuple* format din tipul și numele acestuia, ulterior toate *tuplele* unui tabel urmând să fie adăugate într-o listă
- se adaugă într-un dicționar sub forma:
  - cheie: numele tabelul
  - valoare: *tuple*-u cu informații
- dicționarul se adaugă într-o listă
- formă finală: listă[*{nume tabel: [(tipul field-ului, numele field-ului), ...]}, ...]*

4. Chenarul cu *output*-ul utilizatorului afișează informațiile generate de *query*-ul scris de utilizator
5. Butoanele de control:
  - *Preview*
  - *Submit*
  - *Next*

Butonul de *preview*:

- execută o cerere asincronă către server
- apelează o funcție care:
  - testează *query*-ul introdus de utilizator și returnează trei liste:
    - o listă cu numele coloanelor
    - o listă cu *entry*-urile
    - o listă cu erori
  - returnează o eroare în cazul în care utilizatorul folosește în *query* "*drop*", "*insert*", "*delete*", "*update*"
- în funcție de existența erorilor, se returnează o listă cu acestea în cazul în care există, sau se returnează lista cu numele coloanelor și lista cu *entry*-urile în caz contrar
- cu *JQuery* se preiau informațiile și se afișează sub formă de tabel

Butonul de *submit* folosește aproape aceeași structură ca cel de *preview*, numai că, după ce se trimite informațiile către *JQuery*, în cazul în care nu există nici o eroare:

- se testează și se generează aceeași structură de liste pe baza *query*-ului pus de profesor
- se compară rezultatele
- se salvează în baza de date progresul cu:
  - *id*-ul întrebării
  - *userProfile*-ul utilizatorului
  - *query*-ul scris de utilizator
  - punctajul acordat în urma aplicării baremului făcut de profesor

- se generează un *pk* pentru butonul *Next* astfel:
  - se creează o mulțime de obiecte de tip *exercises* cu toate întrebările existente
  - se creează o mulțime (*remainingQuestions*) de obiecte de tip *exercises* cu toate întrebările la care a răspuns utilizatorul
  - se intersectează cele două mulțimi
  - dacă cele două mulțimi sunt identice, i se atribuie *pk*-ului numărul 1
  - dacă *remainingQuestions* are un singur element, i se atribuie *pk*-ului *id*-ul elementului
  - se parcurge mulțimea *remainingQuestions* și se caută primul element cu *id*-ul mai mic ca *id*-ul întrebării curente
  - dacă se găsește un astfel de element, i se atribuie *pk*-ului *id*-ului elementului respectiv
  - dacă nu există un astfel de element, se ia primul element din mulțime și i se atribuie *pk*-ului *id*-ului elementului respectiv

Butonul *Next* redirecționează utilizatorul către următoarea întrebare la care nu s-a răspuns. Dacă utilizatorul a răspuns la toate întrebările, butonul va fi blocat.

#### 4.4.2 Istoric și verificarea răspunsului

Această pagină încarcă toate obiectele din tabelul *Progress* care au ca *id\_student* *id*-ul utilizatorului curent.

Asemănător paginii cu lista de întrebări, această pagină oferă detalii precum:

- enunțul întrebării
- răspunsul dat de utilizator
- punctele acumulate
- numărul total de puncte care au fost puse la dispoziție
- creatorul întrebării

De asemenea, fiecare card ce conține informațiile precedente, este un *link* spre pagina de verificare. Această pagină este asemănătoare paginii de răspuns la întrebare (4.4.1 *Cum răspund la întrebare?*), cu următoarele diferențe:

- chenarul pentru răspunsul utilizatorului este blocat
- butoanele de control sunt "*Your Answer*" și "*The Solution*"

Butonul "*Your Answer*" face o cerere asincronă către *server* care ulterior încarcă în chenarul pentru răspuns *query*-ul utilizatorului și în chenarul pentru *output*-uri informațiile rezultate după rularea *query*-ului respectiv. Asemănător butonul "*The Solution*", doar că acesta încarcă *query*-ul profesorului și *output*-ul acestuia.

#### 4.4.3 Baremul și punctarea utilizatorului

Atunci când se postează o întrebare, profesorul are la dispoziție un barem pe care îl poate modifica după cum dorește, existând următoarele variabile:

- `totalColumnsPoints` – numărul total de puncte disponibile pentru coloane
- `partialColumnsPoints` – *boolean* cu valoarea 1 dacă profesorul dorește să puncteze parțial coloanele găsite de utilizator, 0 în caz contrar
- `columnPercentage` –  $x$  procente din numărul total de puncte disponibile pentru coloane acordate pentru o coloană găsită
- `totalEntryPoints` - numărul total de puncte disponibile pentru *entry*-uri
- `partialEntryPoints` - *boolean* cu valoarea 1 dacă profesorul dorește să puncteze parțial *entry*-urile găsite de utilizator, 0 în caz contrar
- `entryPercentage` –  $y$  procente din numărul total de puncte disponibile pentru *entry*-uri acordate pentru un *entry* găsit
- `bonusPoints` – puncte bonus acordate celor ce au răspuns corect 100%

Toate variabilele prezentate mai sus pot lua orice valoare dorește profesorul din intervalul 0.00-999.99. De asemenea, se acceptă valori de tip *float*.

Cum se aplică baremul:

- se instanțiază variabila *points* cu 0:
- dacă utilizatorul a găsit toate coloanele cerute, se adună la variabila *points* numărul total de puncte disponibil pentru coloane
- dacă utilizatorul a găsit toate *entry*-urile cerute, se adună la variabila *points* numărul total de puncte disponibil pentru *entry*-uri

- dacă utilizatorul nu a găsit toate coloanele cerute și profesorul vrea să punteze parțial coloanele găsite ( $partialColumnsPoints = True$ ), atunci se adună la variabila  $points$  numărul rezultat după calcularea punctajului pentru coloane cu formula:  $(columnPercentage/100) * totalColumnsPoints$
- dacă utilizatorul nu a găsit toate  $entry$ -urile cerute și profesorul vrea să punteze parțial  $entry$ -urile găsite ( $partialEntryPoints = True$ ), atunci se adună la variabila  $points$  numărul rezultat după calcularea punctajului pentru  $entry$ -uri cu formula:  $(entryPercentage/100) * totalEntryPoints$
- în final,  $bonusPoints$  se adaugă la variabila  $entry$

#### 4.4.4 Cum postează o întrebare ca profesor?

În meniul pentru profesori există un buton care redirecționează utilizatorul către formularul de înregistrare a unei întrebări (Fig. 15).

Fig. 15 – înregistrare întrebare

Elemente principale în pagină:

- Chenar cu detalii despre baza de date
- Chenar cu *output*-ul utilizatorului
- Chenar pentru enunțul întrebării

- Chenar pentru răspunsul întrebării
- Chenar pentru *hint*
- Barem
- Butoane de control

Chenarul cu detalii despre baza de date și chenarul cu *output*-ul utilizatorului au aceeași funcționalitate ca cel de pe pagina pentru răspuns la întrebare (Fig. 13 – Pagină pentru răspuns la întrebare).

Chenarul pentru enunțul întrebării și pentru răspuns sunt obligatorii.

Baremul este modificabil 100%, profesorul având posibilitatea de a-l schimba după cum dorește (explicație barem: 4.4.3 Baremul și punctarea utilizatorului).

Mod de utilizare:

- se introduce textul întrebării
- se introduce răspunsul întrebării
- dacă se dorește, se introduce un *hint*
- se modifică baremul dacă este nevoie
- se apasă pe butonul *Test*
- dacă totul este bine și nu se mai doresc alte schimbări, se apasă pe butonul *Submit*

De menționat este că, butonul *Test* se apasă obligatoriu înainte de *Submit* sau ori de câte ori se fac modificări asupra răspunsului sau a întrebării, altfel butonul de *Submit* nu se deblochează. Acest lucru a fost necesar din punctul de vedere al siguranței, deoarece un *Use Case* destul de plauzibil ar fi:

- se introduce răspunsul întrebării
- se introduce textul întrebării
- se modifică răspunsul întrebării cu unul care generează o eroare
- se apasă *Submit*

Astfel, se înregistrează o întrebare cu o eroare în baza de date și în cel mai bun caz, ori de câte ori un utilizator rezolvă întrebarea și apasă pe *Submit*, răspunsul ori va fi invalidat sau punctat cu 0 deoarece răspunsul profesorului va genera o eroare la rulare. Datorită implementării acestui

*workflow* de testare a posibilei soluții și apoi înregistrarea formularului, acest gen de posibile erori sunt eliminate.

Butoanele de control sunt:

- *Test* – face o cerere asincronă către server și încarcă o funcție asemănătoare ca cea de la butonul *Preview* de pe pagina pentru răspuns la întrebare (Fig. 13 – Pagină pentru răspuns la întrebare)
- *Submit* – preia și validează formularul, apoi înregistrează datele în baza de date

#### 4.4.5 Gestionarea întrebărilor postate

Această pagină (Fig. 16) poate fi accesată folosind butonul "*Manage Questions*" din meniul pentru profesori.

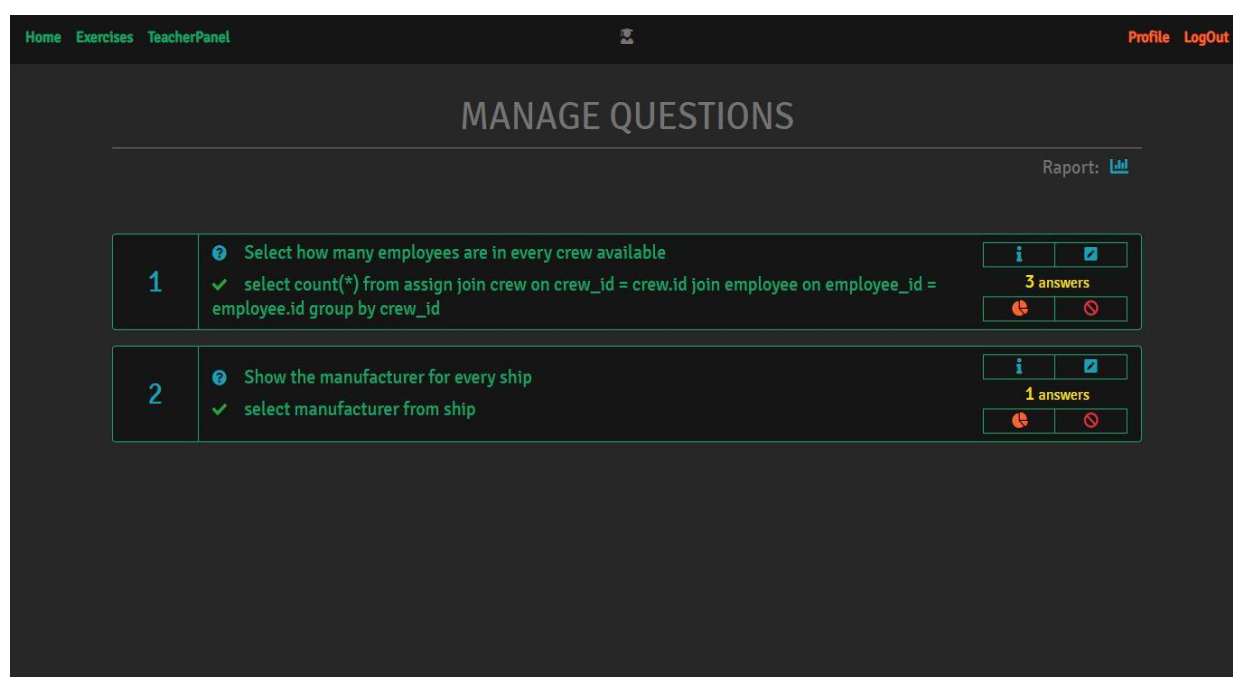


Fig. 16 – Gestiunea întrebărilor postate

Fiecare întrebare are propriul card ce conține:

- enunțul întrebării
- răspunsul întrebării
- numărul de utilizatori ce au răspuns la întrebare



- două simboluri *hoverable*:
  - ”i” – afișează textul *hint*-ului introdus la înregistrarea întrebării. Dacă, nu există se afișează un mesaj reprezentativ
  - cel din dreapta simbolului ”i” – afișează informații despre barem
- un buton sub formă de diagramă radială care redirecționează utilizatorul spre o pagină cu statistici despre întrebarea respectivă
- un buton pentru a șterge întrebarea

Dacă profesorul șterge întrebarea, progresul pentru fiecare utilizatorul asociat întrebării respective se va șterge automat, iar punctajul primit la acea întrebare se va scădea din scorul total tuturor celor ce au răspuns întrebarea respectivă.

Totodată, în colțul din dreapta, partea de sus a paginii, se află un simbol pentru raport. Acesta redirecționează utilizatorul pe o pagină de statistică diferită față de cea accesată de pe butonul de pe card, unde se află informații utile legate de toate întrebările postate de profesor.

#### 4.4.6 Statistici

Pagina de statistici este de două tipuri:

1. Pentru fiecare întrebare în parte, ce oferă informații precum:
  - numărul de răspunsuri corecte
  - numărul de răspunsuri greșite
  - o diagramă radială reprezentând datele de mai sus
  - un tabel cu primii 10 utilizatori care au luat cel mai mare punctaj
  - un tabel cu ultimii 10 utilizatori care au luat cel mai mic punctaj

De asemenea, tabelele conțin informații precum:

- *username*-ul utilizatorului
  - prenumele utilizatorului
  - numele utilizatorului
  - răspunsul utilizatorului
  - punctele primite
2. Pentru toate întrebările se oferă informații precum (Fig. 17):
    - grafic bazat pe numărul total de răspunsuri înregistrate la fiecare întrebare

- întrebarea la care s-a răspuns de cele mai multe ori, alături de *id*-ul și răspunsul ei
- grafic bazat pe numărul total de răspunsuri corecte înregistrate la fiecare întrebare
- întrebarea la care s-a răspuns corect de cele mai multe ori, alături de *id*-ul și răspunsul ei
- grafic bazat pe numărul total de răspunsuri greșite înregistrate la fiecare întrebare
- întrebarea la care s-a răspuns greșit de cele mai multe ori, alături de *id*-ul și răspunsul ei

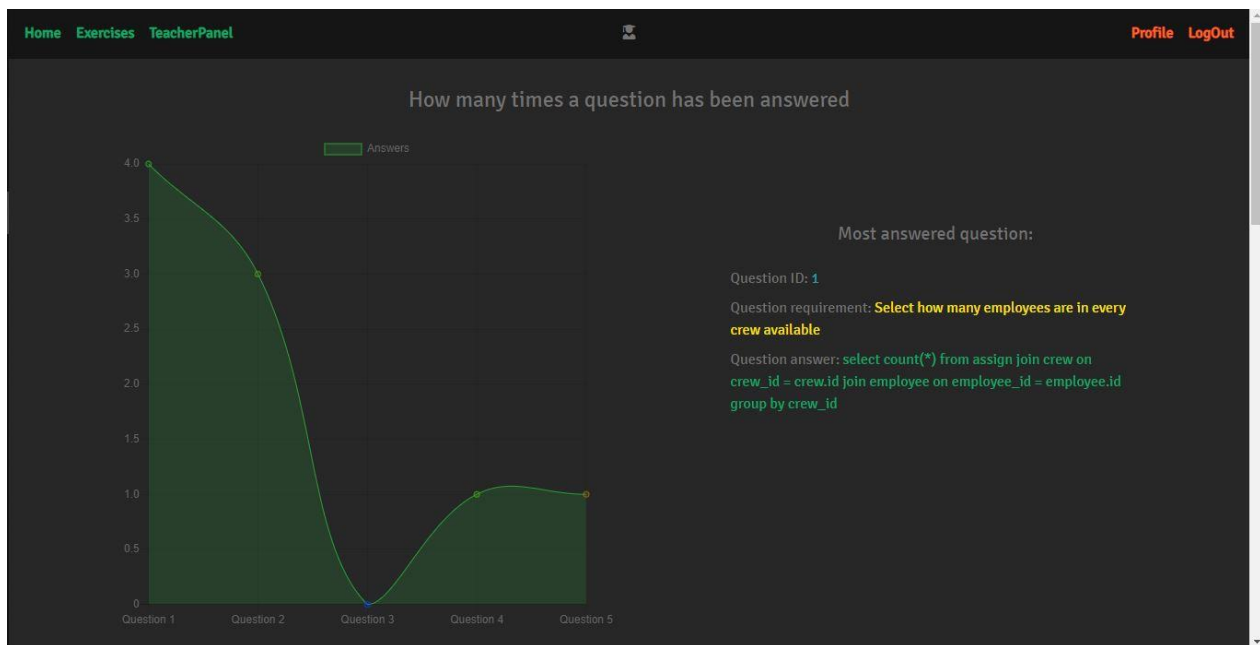


Fig. 17 – statistică pentru întrebarea la care s-a răspuns cel mai mult

## Concluzii

### Dificultăți întâlnite

La început mi-a fost foarte greu să îmi dau seama care este cea mai bună abordare pentru a parsă și a testa comanda SQL preluată de la utilizator. Din cauza lipsei de experiență cu *framework*-ul *Django* în această privință am încercat mai multe metode, unele nu tocmai strălucite, precum: parsarea manuală a *string*-ului, apoi aplicarea lui cu *ORM*-ul din *Django*. Evident, acest lucru nu a funcționat, principalele probleme fiind viitoarele *update*-uri ale aplicației care ar fi putut include sintaxa nouă, cât și securitatea, care nu prea ar fi existat, lăsând cale liberă pentru *SQL injection*. Apoi am încercat o combinație între parsare manuală și aplicarea cu *ORM*-ul *Django*. Rezultatele au fost mai îmbucurătoare, modalitățile de *SQL injection* fiind cât de cât eliminate datorită *ORM*-ului, dar tot exista problema viitoarelor *update*-uri, plus că, schimbarea bazei de date pentru testarea exercițiilor ar fi fost foarte problematică, în funcție de nivelul de *hard-codare* a acesteia în funcția de test. Într-un final, după multe căutări, am găsit metoda *cursor()* implementată de *Django* care lasă utilizatorul să folosească *raw SQL* direct pe baza de date. Am filtrat *query*-ul introdus de utilizator și problemele au fost rezolvate.

Apoi, pe tot parcursul aplicației, cu cât o dezvoltam mai mult cu atât apăreau probleme de logică, precum: un utilizator cu drepturi de profesor putea să facă alte cereri de profesor, un admin putea să își accepte propriile cereri pentru drepturi de profesor, un utilizator putea răspunde de mai multe ori la exerciții. Nu erorile logice erau cele care au cauzat problemele greu de remediat, ci mai degrabă rezolvările acestora, care generau *bug*-uri noi, deoarece trebuiau adaptate la versiunea curentă a aplicației.

O altă problemă care a fost semnificativă în raport cu timpul de rezolvare al acesteia a fost generarea dinamică a informațiilor din modulul *Exercises*. La început a fost *hard-codat* 100%, apoi am orientat aplicația spre o arhitectură care să poată fi dezvoltată și pe viitor, așa că am generat chenarul cu ajutorul unui XML. Deși era o variantă mai acceptabilă decât prima, nu era o soluție de viitor, fiind necesar să se facă de două ori același lucru: scrierea codului pentru modelul bazei de date, rescrierea acestuia în XML. Într-un final, am găsit în documentația *Django* o funcție pentru returnarea tuturor tabelelor dintr-un modul sub formă de obiecte care m-a ajutat în generarea complet dinamică a chenarului cu informații.

De asemenea, am întâmpinat probleme la sincronizarea animațiilor cu funcționalitatea elementelor din interfață și cu timpii de încărcare a datelor. Datorită portabilității aplicației, nu am avut cum să mă bazez pe faptul că o funcție X se încarcă în Y secunde, mai ales că soluția trebuia să fie independentă de configurația calculatorului pe care rula aplicația.

## Dezvoltare ulterioară

Deși aplicația este într-o stare bună de funcționare, acesteia i se vor mai adăuga următoarele îmbunătățiri:

- Adăugarea unui modul cu rol de ajutor pentru utilizatori în rezolvarea întrebărilor, care să aibă la bază un algoritm de *machine learning*, care în funcție de *query*-urile scrise de utilizator, acesta să poată oferi sfaturi despre cum ar putea să rezolve mai bine exercițiile următoare.
- Dezvoltarea statisticilor pe mai multe arii de interes, în conformitate cu nevoile profesorilor.
- Resetarea parolei prin *e-mail*
- Adăugarea unui buton pentru dezactivarea animațiilor de pe *site*
- Adăugarea unui meniu în care profesorul să își poată alege ce animație să ruleze la întrebările sale
- Adăugarea posibilității pentru profesori de a crea tabele noi
- Adăugarea posibilității pentru profesori de a introduce *entry*-uri noi în baza de date pentru testarea exercițiilor
- Adăugarea unui modul nou în care profesorii să poată încărca cursuri sau materiale
- Adăugarea unui sistem de notificare pentru admini atunci când sunt cereri noi
- Adăugarea posibilității de a raporta probleme la întrebări
- Adăugarea unui sistem de mesagerie între utilizatori
- Adăugarea unui sistem de înregistrare a activităților adminilor
- Adăugarea posibilității de a schimba tema interfeței aplicației
- Adăugarea unui sistem de notare a profesorilor de către utilizatori
- Adăugarea posibilității de a ține sesiuni de *live stream* pentru profesori
- Extinderea aplicației către mai multe limbaje de programare

## Bibliografie

### Cărți

- William S. Vincent, (07.03.2018), *Django for Beginners: Build websites with Python and Django*, Independently published
- Paul Barry, (03.12.2016), *Head First Python: A Brain-Friendly Guide 2nd Edition*, O'Reilly Media

### Link-uri

- Documentație *Python*: <https://docs.python.org/3/>
- Documentație *Django*: <https://docs.djangoproject.com/en/2.2/>
- Curs *Django*: <https://www.udemy.com/python-and-django-full-stack-web-developer-bootcamp/>
- *Animate.CSS*: <https://daneden.github.io/animate.css/>
- *FontAwesome*: <https://fontawesome.com/>
- Documentație *Chart.js*: <https://www.chartjs.org/docs/latest/>
- Documentație *JQuery*: <https://api.jquery.com/>
- Documentație *Bootstrap*: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
- Pentru mici probleme: <https://stackoverflow.com/>