

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI  
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**EloLearn – Software Educațional**

propusă de

***Andrei-Valerio Mihăilă***

**Sesiunea:** *iulie, 2018*

Coordonator științific

**Lector Dr. Cristian Frăsinaru**

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI  
FACULTATEA DE INFORMATICĂ

# **EloLearn – Software Educațional**

***Andrei-Valerio Mihăilă***

**Sesiunea:** *iulie, 2018*

Coordonator științific

***Lector Dr. Cristian Frăsinaru***

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 .....

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*EloLearn – Software Educațional*”, 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,

Absolvent *Andrei-Valerio Mihăilă*

---

(semnătura în original)

# Cuprins

MOTIVAȚIE .....	7
INTRODUCERE .....	9
CAPITOLUL I: APLICAȚII SIMILARE .....	10
I.1 Codecademy .....	10
I.2 Udacity.....	12
I.3 HackerRank.....	13
I.4 Concluzii.....	15
CAPITOLUL II: ARHITECTURA APLICAȚIEI .....	16
II.1 Model View Controller .....	16
II.2 Arhitectura de tip Onion .....	18
III.3 Concluzii.....	20
CAPITOLUL III: TEHNOLOGII UTILIZATE.....	21
III.1 Flask.....	21
III.1.1 Despre .....	21
III.1.2 Instalare și rulare.....	21
III.1.3 Extensii .....	22
III.1.4 Utilizări.....	23
III.2 Jinja2.....	23
III.3 MaterializeCSS .....	23
III.4 Ace.....	24
II.5 Chartist.js.....	24

<b>II.6 MySQL .....</b>	<b>24</b>
<b>II.7 SQLAlchemy.....</b>	<b>24</b>
<b>III.8 Docker .....</b>	<b>25</b>
III.8.1 Despre .....	25
III.8.2 Containere versus mașini virtuale .....	26
III.8.3 Instalare pe sistemul de operare Windows .....	27
III.8.4 Utilizare Docker pe sistemul de operare local.....	29
<b>III.9 Concluzii.....</b>	<b>30</b>
<b>CAPITOLUL IV: DETALII DE IMPLEMENTARE.....</b>	<b>31</b>
<b>IV.1 Sistemul de evaluare Elo.....</b>	<b>31</b>
IV.1.1 Baza teoretică .....	31
IV.1.2 Integrarea sistemului de evaluare în cadrul aplicației.....	33
IV.1.3 Adaptarea la rezolvarea unor funcții.....	35
<b>IV.2 Modulul de exerciții.....</b>	<b>37</b>
IV.2.1 Structura unui exercițiu.....	37
IV.2.3 Recomandarea unui exercițiu .....	38
<b>IV.3 Concluzii .....</b>	<b>39</b>
<b>CONCLUZIILE LUCRĂRII ȘI DIRECȚII VIITOARE .....</b>	<b>40</b>
<b>Concluzii generale.....</b>	<b>40</b>
<b>Îmbunătățiri și direcții viitoare .....</b>	<b>40</b>
<b>BIBLIOGRAFIE .....</b>	<b>42</b>

## Motivație

După cum spunea și Steve Jobs:

*”Toată lumea din această țară ar trebui să știe cum să programeze un calculator deoarece te învață cum să gândești”.*

În prezent, făcând o paralelă cu situația de acum un deceniu sau două, din ce în ce mai multă lume își dorește să învețe programare, nu doar cu scopul de a putea avea o carieră în domeniu ci chiar pentru a înțelege minimul de noțiuni sau pentru a automatiza unele activități monotone de zi cu zi.

Multor persoane le este dificil să înceapă de unii singuri să învețe bazele unei arii în care nu au deloc expertiză, multe cursuri online pierzându-se în detalii în loc de a rămâne pe subiect, utilizatorii renunțând în cele din urmă la inițiativa lor. Alte cursuri necesită instalarea a diferite module, pachete, kit-uri de dezvoltare direct pe mașina locală a utilizatorului, care câteodată necesită pași în plus, fiind dificil pentru un utilizator neexperimentat.

Aplicația este destinată persoanelor non-tehnice care doresc să debuteze în aria programării, cursurile fiind mai mult de natură practică decât teoretică, utilizatorul fiind încurajat să scrie cod într-un editor de text urmând instrucțiuni cu posibilitatea de a-l testa și de a-l trimite aplicației pentru a-l putea evalua. Logica aplicației este gândită în așa fel încât să simuleze metodologiile și abordările unui laborator dintr-o instituție, având 4 secțiuni: **Lecții**, **Exerciții**, **Teme** și **Progres**.

Secțiunea **Lecții** cuprinde lecții grupate pe capitole ce vor fi parcurse iterativ de către utilizator, fiecare având un nivel de dificultate, acestea ajustându-se după efortul utilizatorului în încercarea de a le rezolva. De exemplu, cu cât un utilizator are mai multe încercări eșuate la o lecție, cu atât nivelul de dificultate al lecției va crește.

Spre deosebire de anumite platforme online de învățat programare, prin secțiunea **Exerciții** aplicația va recomanda utilizatorului un exercițiu ales în funcție de progresul acestuia acumulat la secțiunea **Lecții**. Exercițiile având corespondență directă cu lecțiile aplicației.

Periodic vor fi sugerate teme pentru utilizator, acesta având un termen limită pentru ele pentru a obține un punctaj, iar cu cât trece mai mult timp după termenul limită, punctajul va scădea.

Cea de-a patra secțiune: **Progres**, va constitui un set de grafice ce vor evidenția evoluția/regresul utilizatorului pe parcursul aplicației cât și numărul de lecții, exerciții și teme completate de acesta.

Limbajul de programare ce va fi folosit pentru întregul proces de învățare a utilizatorului este Python, varianta 3.6. Am ales acest limbaj deoarece a început să fie folosit de multe departamente de informatică de la MIT și UC Berkeley în cadrul cursurilor introductive de programare. Un alt aspect este sintaxa ușoară pe care acesta o are, fiind un avantaj deoarece aplicația este destinată învățării conceptelor de bază ale programării ci nu detalii tehnice, lecțiile și exercițiile fiind axate pe rezolvare de probleme simple, nu complexe.



## Introducere

Lucrarea cuprinde patru capitole în care sunt prezentate și detaliate (acolo unde este cazul) noțiuni, tehnici cât și tehnologii utilizate în cadrul aplicației.

**Capitolul I** prezintă aplicații similare ce sunt destinate învățării programării în mediul online, fiecare fiind unică în felul ei și folosind diferite tehnici pentru a ajuta utilizatorul să înțeleagă anumite noțiuni din domeniul programării.

**Capitolul II** deține informații despre arhitecturile software folosite în cadrul dezvoltării aplicației, punând în evidență utilitatea acestora prin exemple din codul sursă al proiectului și structura directoarelor și a fișierelor.

**Capitolul III** menționează tehnologiile utilizate în cadrul aplicației, punând accent pe cele care dețin un element de noutate (cum ar fi Flask sau Docker) și au ajutat la dezvoltarea aplicației, acestea fiind soluții ale unor probleme întâlnite pe parcursul dezvoltării.

**Capitolul IV** conține detalii despre sistemul de evaluare folosit în cadrul aplicației cât și despre sistemul de recomandare al exercițiilor din cadrul aplicației, aceste detalii fiind esențiale aplicației.

Ultima parte pune în evidență concluziile lucrării și îmbunătățirile ce ar putea fi aduse aplicației.

## Capitolul I: Aplicații similare

În acest capitol vor fi ilustrate aplicații ce fac parte din categoria aplicațiilor web de tip e-learning axate de asemenea pe învățarea noțiunilor de bază a unor limbaje de programare.

### I.1 Codecademy

Codecademy<sup>1</sup> este o platformă online și gratuită, având unele limitări, ce oferă utilizatorilor o varietate de cursuri pe diverse limbaje de programare, axându-se pe noțiuni de bază ale unor limbaje de programare cunoscute cum ar fi: Java, Javascript, Python etc.

Aplicația folosește într-un mod minimal tehnica de *gamification* prin răsplătirea utilizatorului pentru progresul și activitatea sa în cadrul cursurilor prin insigne. De asemenea reține progresul pentru fiecare curs în parte (vezi Figura 1).

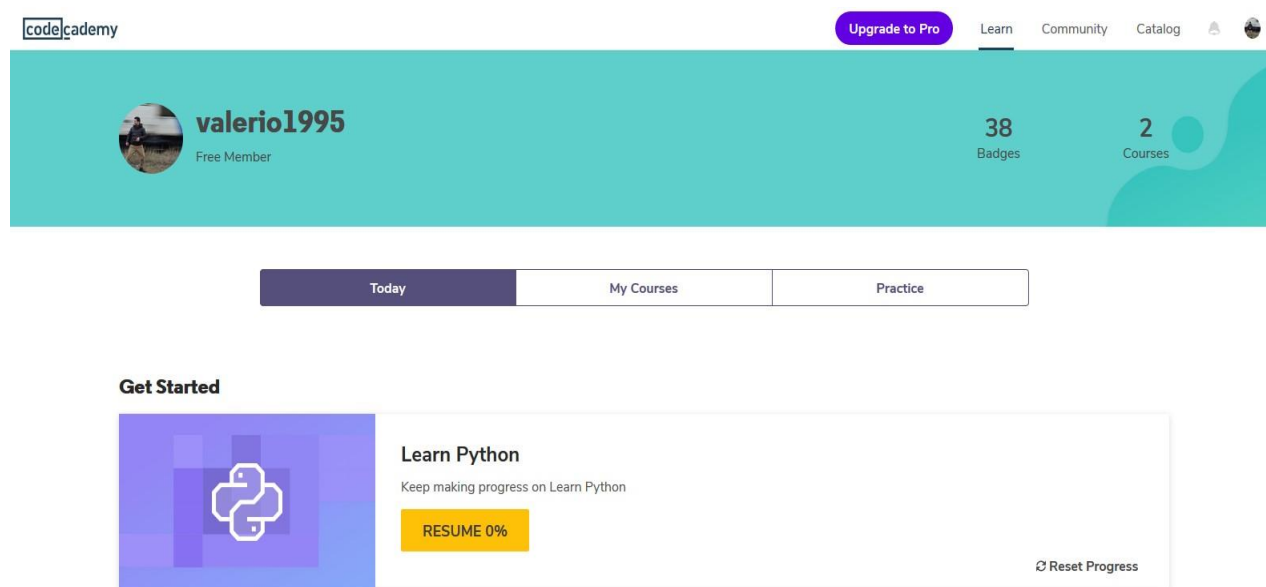
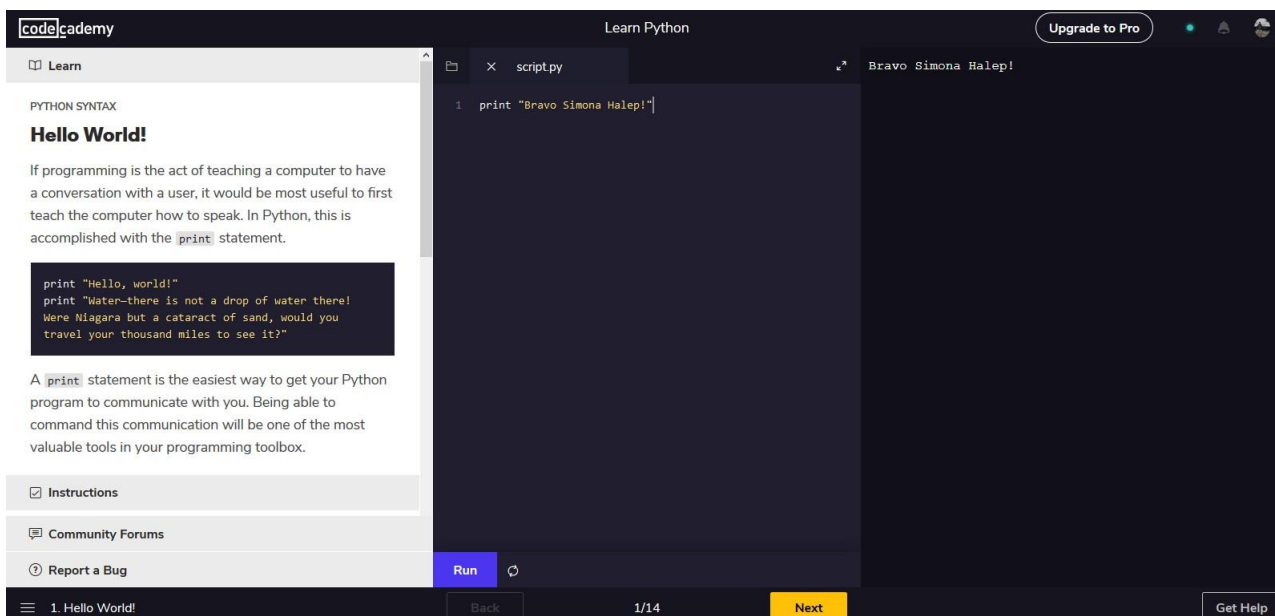


Figura 1: Dashboard-ul principal al platformei Codecademy

<sup>1</sup> <https://www.codecademy.com/>

Lecția unui curs urmează un format standard ce este respectat de multe platforme online când vine vorba de design. Acesta având instrucțiunile în stânga ecranului, un editor de text unde utilizatorul își va scrie soluția, butonul de rulare al codului și în partea dreaptă consola unde va fi afișat output-ul execuției codului. (vezi Figura 2).



**Figura 2: Consola platformei Codecademy**

Pe lângă cursurile minimaliste, platforma oferă o gamă de proiecte ce necesită cunoștințe minime dobândite în cadrul cursurilor pentru a putea fi realizate, soluțiile oferite de utilizatori rezolvând probleme întâlnite în practică cum ar fi: o aplicație web ce face conversia din Kelvin în Fahrenheit sau crearea unui design pentru orarul unui eveniment. Însă, pentru a avea acces la lista de proiecte, utilizatorul va trebui să își reînnoiască contul, trecând pe versiunea PRO a platformei, această versiune având diverse tarife, în funcție de planul ales. (vezi Figura 3).

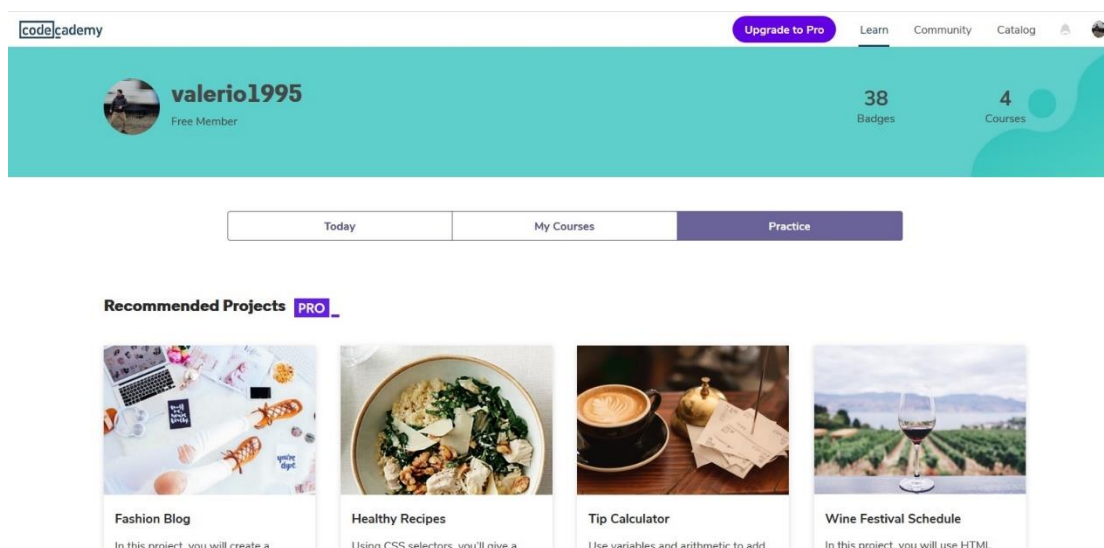


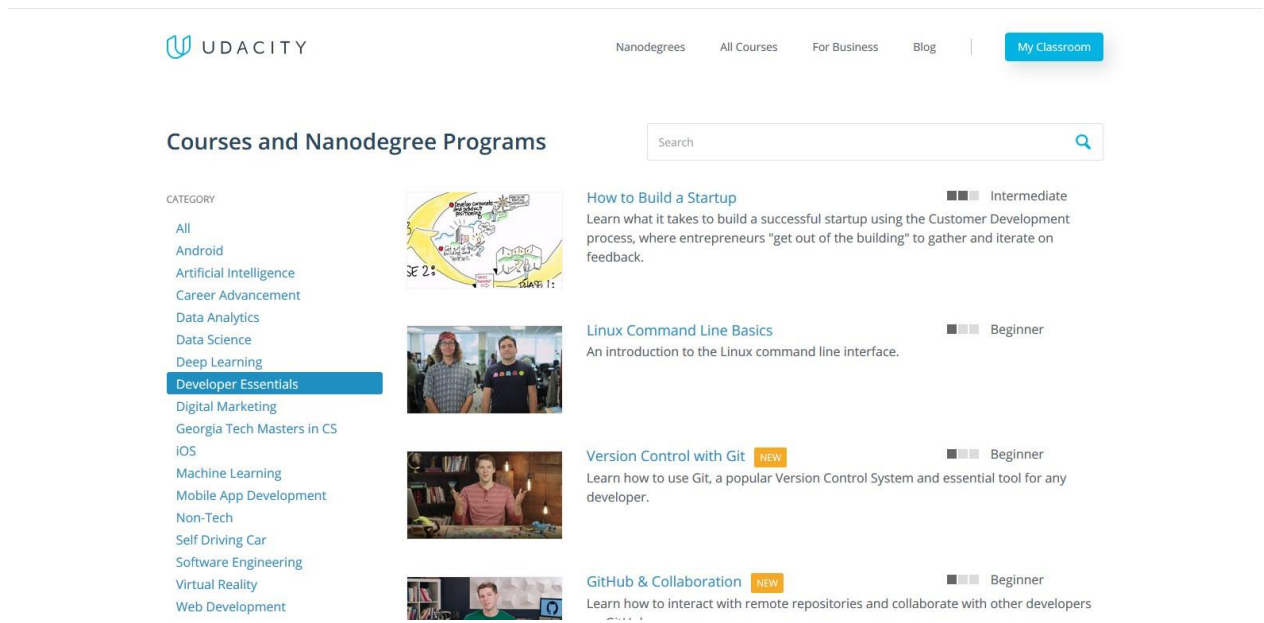
Figura 3: Lista proiectelor de pe Codecademy

## I.2 Udacity

*Udacity*<sup>2</sup> a început ca un experiment al profesorilor Sebastian Thrun și Peter Novig de la Stanford atunci când aceștia au oferit online cursul lor de *Introducere în Inteligență Artificială*, stârnind interes pentru peste 160.000 de studenți.

În ziua de azi Platforma online Udacity cuprinde o foarte mare varietate de cursuri din multe subdomenii ale informaticii cum ar fi: inteligență artificială, învățare automată, dezvoltarea aplicațiilor web. Pe lângă cursurile tehnice, se regăsesc și cursuri din alte domenii cum ar fi: fizică, psihologie, management sau pentru dezvoltare personală, acestea fiind categorizate pe subdomeniul corespunzător, fiecare curs fiind etichetat cu un nivel de dificultate (vezi Figura 4).

<sup>2</sup> <https://eu.udacity.com/>



**Figura 4: Structura cursurilor de pe platforma Udacity**

Spre deosebire de *Codecademy*, această platformă nu oferă un mediu de dezvoltare pregătit pentru utilizator unde acesta ar putea încă de la prima accesare să-și testeze codul, ci este nevoie ca utilizatorul să-și configureze mașina local, conform explicațiilor din primele lecții ale unui curs.

Cursurile sunt structurate ca fiind o serie de videoclipuri pe care utilizatorul le parcurge secvențial, urmărind instrucțiunile îndrumătorilor, aceștia fiind persoane tehnice de la companii renumite cum ar fi Amazon, Google, Microsoft etc.

Accesul cursurilor și completarea acestora este gratuit, însă dacă utilizatorul dorește ca la finalizarea unui curs acesta să îi fie recunoscut, atunci va trebui să plătească pentru eliberarea unei diplome.

### **I.3 HackerRank**

*HackerRank*<sup>3</sup> este o aplicație online ce oferă servicii atât pentru utilizatori cât și pentru companii. Platforma oferă utilizatorilor posibilitatea de a învăța multiple limbaje de programare cât și concepte din aria algoritmicii sau a structurilor de date, prin rezolvarea unor probleme propuse (vezi Figura 5).

<sup>3</sup> <https://www.hackerrank.com/>

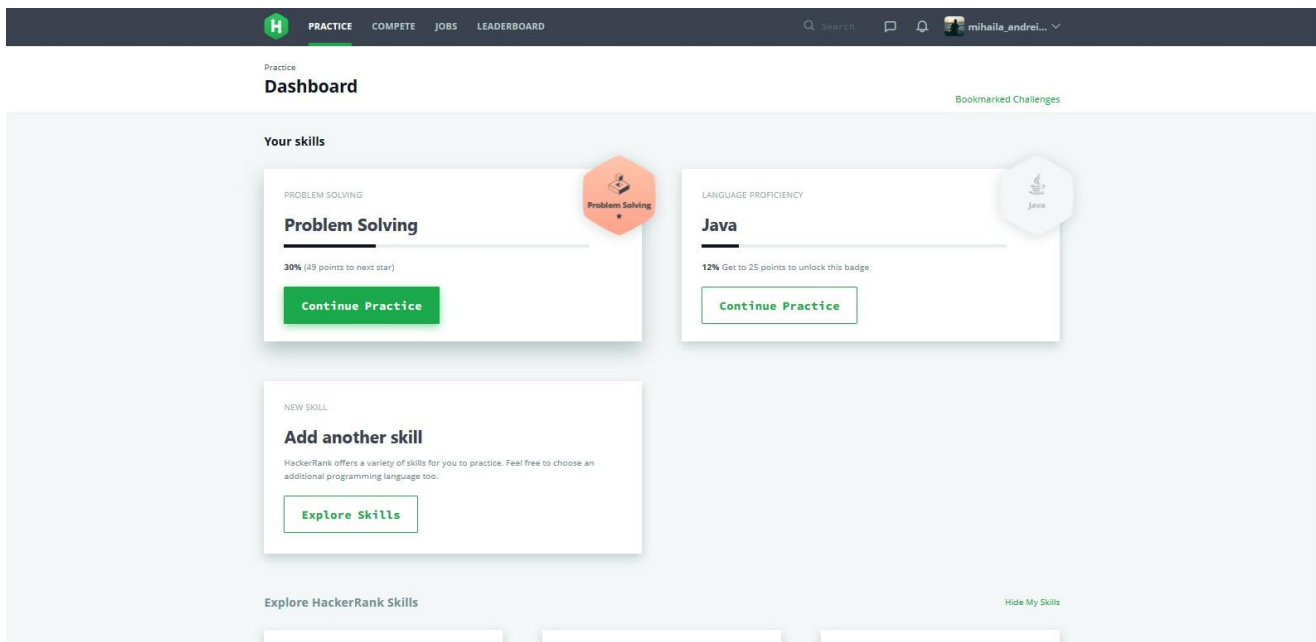


Figura 5: Dashboard-ul aplicației HackerRank

De asemenea pe lângă modulul de învățat, aplicația oferă și suport pentru competiții (vezi Figura 6) , fiind organizate destul de des și stabilite la o dată precisă, utilizatorii având posibilitatea să își testeze abilitățile și cunoștințele împotriva altor utilizatori, acest lucru conturând aspectul de *gamification* pe care îl adoptă HackerRank.

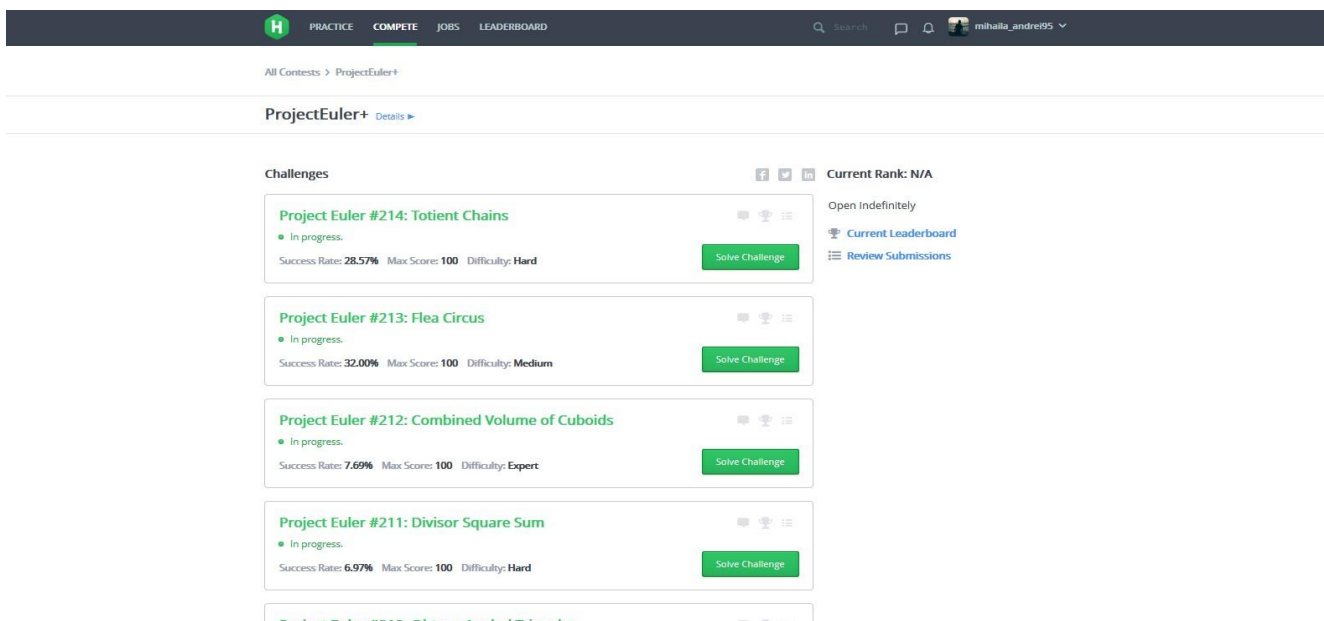


Figura 6: Competițiile din cadrul HackerRank

## **I.4 Concluzii**

Învățarea online s-a dezvoltat foarte puternic în ultimii ani, iar din ce în ce mai multe platforme apar, acestea având o multitudine de cursuri, multe din acestea fiind și cursuri specializate de la oameni cu experiență lucrând în domeniu sau chiar cursuri de la universități de prestigiu. Multe domenii sunt acoperite de către aceste platforme, însă cea pe care se pune cel mai mult accent este domeniul Informaticii, mai ales partea aplicată al acestuia, utilizatorii având posibilitatea de a învăța atât partea de proiectare a unui site web cât și rezolvarea unor probleme folosind diverși algoritmi.

Când vine vorba de a învăța ceva nou, o mare majoritate de persoane se simt descurajate, mai ales dacă sunt puse într-o situație în care informațiile sunt prea explicite și greu de înțeles. Multe aplicații de învățare online necesită din partea utilizatorului instalarea și configurarea unor medii de dezvoltare sau în caz că utilizatorul rămâne blocat într-un anumit pas al unui curs, acesta este incapabil să avanseze de la acel impediment, fiind într-o stare blocantă. De asemenea o altă problemă ar fi faptul că multe platforme oferă o funcționalitate standard, însă pentru o experiență mai diferită, utilizatorul trebuie să treacă pe o versiune plătită a aplicației.

## Capitolul II: Arhitectura aplicației

În capitolul curent vor fi prezentate modelele arhitecturale care stau la baza aplicației și au fost de ajutor cu privire la structurarea proiectului, separând responsabilitățile fiecărei entități, conducând astfel la o dezvoltare rapidă și lăsând loc de extensie, funcționalități noi putând fi integrate ușor.

### II.1 Model View Controller

Una din arhitecturile folosite în cadrul proiectului este Model-View-Controller. Această arhitectură separă entitățile din aplicație, categorizându-le drept modele, făcând structurat accesul și afișarea acestora în interfața reprezentată de pagini HTML.

După cum se poate observa și în Figura 11, când utilizatorul dorește să citească o resursă (în cadrul aplicației, orice resursă este reprezentată de un model), acestuia i se va furniza un view. În cazul în care modelul va suferi o modificare, acea modificare va fi interceptată de Controller-ul asociat modelului și va trimite modificarea cerută mai departe la nivelul logic al aplicației.

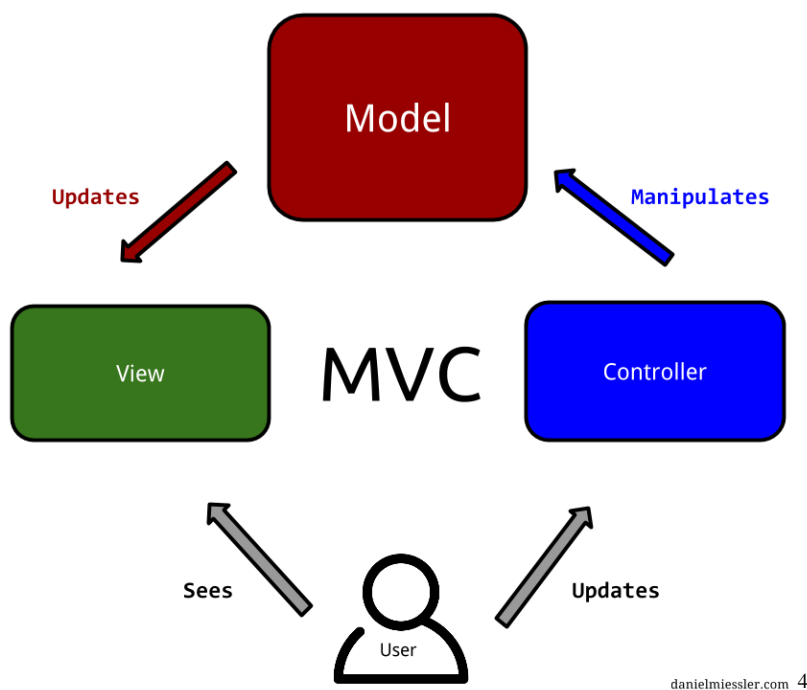


Figura 11: Arhitectura MVC

Când utilizatorul va dori să vizualizeze o lecție din cadrul aplicației, acesta o va accesa

<sup>4</sup> <https://danielmiessler.com/images/MVC1.png>



interacționând cu un obiect HTML având o rută definită:

```
<a href="/lesson/{{ lesson[0].id }}" ... />
```

În exemplul de față, utilizatorul va cere o resursă de tip *Lesson* ce este stocată în baza de date. Aceasta resursă fiind reprezentată în logica aplicației prin următoarea clasă:

```
class Lesson(db.Base):
    __tablename__ = 'lesson'

    id = Column(Integer(), primary_key=True)
    title = Column(String(50))
    content = Column(MEDIUMTEXT())
    instructions = Column(MEDIUMTEXT())
    source_code = Column(MEDIUMTEXT())
    user_lesson_difficulty = relationship(UserLessonDifficulty, uselist=False)
    default_elo_rating = Column(Float())
```

Cererile HTTP ce conduc la ruta */lesson/<lesson\_id>* sunt interceptate de un controller ce va apela la partea de logică a aplicație pentru a aduna toate componentele ce sunt necesare paginii cerute de utilizator, după cum se poate vedea în secvența de cod de mai jos:

```
@lessons.route("/lesson/<lesson_id>", methods=['GET'])
def get_lesson_by_id(lesson_id):

    lesson = _lesson_service.get_lesson_by_id(lesson_id)
    is_current_completed = _lesson_service.is_lesson_completed_by_user(lesson.id)

    previous_lesson = _lesson_service.get_previous_lesson(lesson_id)

    is_previous_completed = False
    if previous_lesson:
        is_previous_completed = _lesson_service.is_lesson_completed_by_user(previous_lesson.id)

    next_lesson = _lesson_service.get_next_lesson(lesson_id)
    is_next_completed = False
    if next_lesson:
        is_next_completed = _lesson_service.is_lesson_completed_by_user(next_lesson.id)

    temporary_code = _lesson_service.get_temporary_code(lesson_id)

    return render_template("lesson.html", lesson=lesson,
                           is_current_completed=is_current_completed,
                           previous_lesson=(previous_lesson, is_previous_completed),
                           next_lesson=(next_lesson, is_next_completed),
                           temporary_code=temporary_code)
```

Controller-ul returnează template-ul paginii html asociat entității *Lesson* împreună cu un dicționar ce conține datele necesare de pe pagina cerută.

## II.2 Arhitectura de tip Onion

A doua arhitectură folosită în cadrul proiectului este arhitectura de tip Onion<sup>5</sup> ce a avut un impact asupra business-ului aplicației, separând funcționalitățile și lăsând loc de extindere a aplicației.

Arhitectura structurează logica aplicației pe nivele, astfel încât orice nivel nu va depinde de nivelurile exterioare ci doar de cele interioare (vezi Figura 12), fiecare nivel având semnificația sa în logica aplicației.

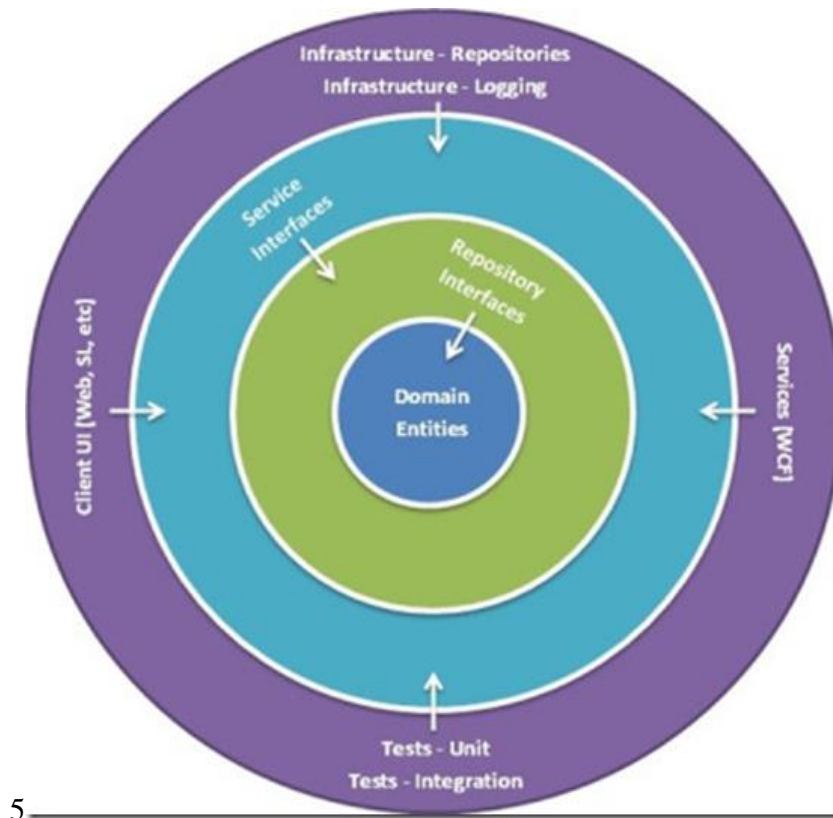


Figura 12: Arhitectura de tip Onion

Inițial proiectul este împărțit în trei subdirectoare: Business, Data și Presentation (vezi Figura 13). Vom explica pe scurt conținutul fiecare subdirector și cum a fost implementată arhitectura de tip Onion, pornind de la modulul cel mai exterior la cel mai interior.

<sup>5</sup> <http://blog.thedigitalgroup.com/understanding-onion-architecture>

```

licenta/
├── Business/
│   ├── Repositories/
│   └── Services/
├── Data/
│   ├── Domain/
│   └── Persistence/
├── Presentation/
│   ├── static/
│   │   ├── aceditor/
│   │   ├── chartist/
│   │   ├── css/
│   │   ├── img/
│   │   └── js/
│   └── templates/

```

**Figura 13: Structura pachetelor din proiect**

- **Presentation**

Conține clasele de tip controller (menționate mai devreme la arhitectura MVC) împreună cu toate resursele necesare pentru a reda interfața (foi de stiluri, script-uri Javascript, template-urile HTML etc.)

- **Business**

După cum se poate vedea din structura pachetelor din proiect (vezi Figura 13), acest nivel conține două tipuri de clase: *Repository* și *Service*.

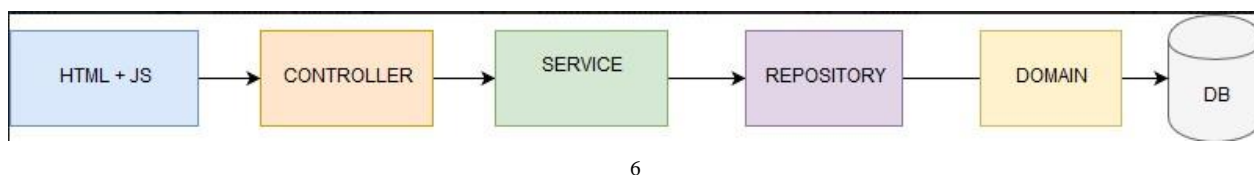
Clasele de tip *service* sunt apelate în cadrul clasele de tip controller pentru executa o acțiune cerută de către utilizator, aceste clase având implementate în mare parte logica aplicației.

Pe de altă parte clasele de tip *repository* sunt folosite de clasele de tip *service* atunci când este nevoie de o resursă din baza de date sau se cere modificarea uneia dintre ele, acest nivel fiind menit să fie mai *low-level*.

- **Data**

Este pachetul care are de-a face cu stocarea și translatarea datelor din bază de date în obiecte de tip Python, având modele definite în *Domain*, iar obiecte ce ajută la persistența datelor (de exemplu sesiunea bazei de date, managerul de persistență etc.) se află în *Persistence*.

Arhitectura Onion reușește să separe responsabilitățile claselor și să definească un flow ce poate fi aplicat oricărei funcționalități (vezi Figura 14).



**Figura 14: Flow-ul funcționalităților din aplicație**

### III.3 Concluzii

Arhitecturile specifice aplicațiilor software au un rol esențial când vine vorba de separarea funcționalităților și de scrierea într-un mod organizat a codului. Proiectul, fiind bazat pe cele două arhitecturi discutate, poate fi extins cu ușurință dacă am presupune că altcineva ar dori să contribuie cu îmbunătățiri sau cu funcționalități noi.

---

<sup>6</sup> <https://www.draw.io>

## Capitolul III: Tehnologii utilizate

Aplicația a fost dezvoltată folosind Flask pentru partea de back-end împreună cu Jinja2 pentru template-urile HTML, MySQL pentru a stoca datele și Docker pentru a rula codul sursă preluat din interfață într-un mediu izolat de sistemul de operare de pe care rulează server-ul principal.

### III.1 Flask

#### III.1.1 Despre

*Flask*<sup>7</sup> este un microframework scris în Python ce face dezvoltarea unei aplicații web să pară foarte ușoară, acesta având o sintaxă intuitivă și ușor de înțeles. Termenul de microframework face referire la faptul că Flask este decuplat și nu necesită instalarea altor componente pentru a începe procesul de implementare a unei aplicații.

#### III.1.2 Instalare și rulare

Acesta poate fi instalat ușor și rulat imediat folosind următoarele instrucțiuni (valabile pentru sistemul de operare Windows):

1. Se rulează următoarea comandă în terminal (este necesar ca o versiune de python și pip să fie deja instalată pe mașina locală)

```
pip install Flask
```

2. Se scrie într-un fișier .py (să considerăm fișierul cu numele exemplu.py) următoarea secvență de cod, practic un server care returnează un mesaj standard.

---

<sup>7</sup> <http://flask.pocoo.org/>

```
from flask import Flask
app = Flask("my flask app")

@app.route("/hello")
def say_hello():
    return "Ciao Mondo!"

if __name__ == "__main__":
    app.run()
```

3. Se rulează fișierul din mediul de dezvoltare sau direct de la linia de comandă folosind următoarele comenzi:

```
set FLASK_APP = <cale_fișier>/exemplu.py
flask run
```

4. Se deschide un browser la adresa <http://127.0.0.1:5000/hello> și observăm mesajul scris în funcția de *say\_hello()* de la pasul 3.

### III.1.3 Extensii

Deși Flask nu depinde de librării externe, pentru a ușura dezvoltarea unor funcționalități de bază a unei aplicații web cum ar fi autentificarea, înregistrarea sau validarea formularelor, Flask dispune de extensii ce pot fi instalate și integrate în aplicația curentă, cum ar fi:

- **Flask WTF**<sup>8</sup>, folosit pentru construirea formularelor, făcând validări pe acestea și având mecanisme de protecție împotriva atacurilor de tip CSRF.
- **Flask Login**<sup>9</sup> asigură un mecanism pentru autentificare a utilizatorului și oferă mecanisme de protecție/expirare a sesiunii.
- **Flask-Migrate**<sup>10</sup> este folosit pentru a actualiza schema bazei de date folosită de aplicație, făcând posibilă revenirea la o versiune anterioară a bazei de date, deși această extensie a fost înlocuită de modulul **Alembic**, fiind dezvoltat tot de către autorii Flask.

---

<sup>8</sup> <http://flask-wtf.readthedocs.io/en/stable/>

<sup>9</sup> <http://flask-login.readthedocs.io/en/latest/>

<sup>10</sup> <https://flask-migrate.readthedocs.io/en/latest/>

- **Flask-Admin**<sup>11</sup> pentru a oferi o interfață de administrator cu toate entitățile din baza de date ce pot fi listate, modificate, adăugate sau șterse.

### III.1.4 Utilizări

Chiar dacă la prima vedere Flask pare o soluție pentru aplicații de dimensiuni mici sau prototipuri, de menționat este faptul că acesta este folosit în dezvoltarea unor aplicații cunoscute de dimensiuni mari cum ar fi: LinkedIn, Twilio sau Pinterest.<sup>12</sup>

## III.2 Jinja2

Pentru a reda conținut pe paginile HTML ale aplicației, s-a folosit Jinja2<sup>13</sup>, un motor de templetizare, ce are rol de a abstractiza paginile printr-un limbaj propriu de templetizare, paginile HTML fiind populate cu date pe partea de server a aplicației.

Acesta oferă și posibilitatea de a modularize conținutul HTML prin includerea unui fragment HTML într-o altă pagină. Un exemplu din codul sursă al proiectului:

```
{% include 'navbar.html' %}

<div>
  <div id = "lesson-title"><h3>{{ exercise.title }}</h3></div>
</div>
```

Componenta *navbar.html* fiind cea unde este design-ul meniului ce trebuie să apară pe toate paginile, iar pentru a evita cod duplicat, s-a folosit Jinja2 pentru a include această componentă acolo unde este nevoie de ea, iar conținutul din interiorul tag-ului `<h3>` este transmis de către partea de back-end a aplicației.

## III.3 MaterializeCSS

MaterializeCSS<sup>14</sup> este un framework de CSS ce vine în ajutor cu diverse componente ce oferă un plus de uzabilitate utilizatorului cum ar fi: butoane cu o gamă largă de iconițe și efecte vizuale (de exemplu pulsare).

---

<sup>11</sup> <http://flask-admin.readthedocs.io/en/latest/>

<sup>12</sup> <https://www.quora.com/What-is-the-largest-site-created-using-Flask>

<sup>13</sup> <http://jinja.pocoo.org/>

<sup>14</sup> <https://materializecss.com>

Acesta împrumută multe elemente de design din spiritul aplicațiilor Android, cât și unele funcționalități cum ar fi Toast<sup>15</sup>-ul.

### III.4 Ace

Ace<sup>16</sup> este o soluție ce rezolvă problema scrierii unei secvențe de cod pe o aplicație web, oferind un API ce pune la dispoziția programatorului posibilitatea de a customiza un editor de text în așa fel încât imită cât mai bine senzația scrierii codului într-un mediu de dezvoltare. Acesta dispune de funcționalități cum ar fi: colorarea codului în funcție de limbajul de programare care este scris, marcarea liniilor unde s-a sesizat o eroare în cod, auto completarea codului etc.

Ace este o soluție potrivită pentru editarea de text online, printre cei care folosesc această soluție numărându-se: Amazon Web Services, Rstudio, Codecademy și ShareLatex.

### II.5 Chartist.js

Chartist.js<sup>17</sup> este o librărie de Javascript ce oferă soluții pentru redarea unor statistici sub formă de grafice, acestea având posibilitatea de a fi personalizate de la design până la animații.

### II.6 MySQL

S-a folosit MySQL<sup>18</sup> versiunea 5.7.20 pentru a stoca datele generate de aplicație și mediul de dezvoltare MySQL Workbench versiunea 6.3 pentru a interoga și a performa operații de tip DDL pe datele existente, cu scop de testare.

### II.7 SQLAlchemy

SQLAlchemy<sup>19</sup> este un Object Relational Mapper ce este folosit pentru a lucra cu clase în loc de tabele SQL, având un sistem ce oferă soluții atunci când vine vorba de persistența datelor. Cu ajutorul acestui framework, putem obține date din baza de date sub forma unor obiecte de tip Python acestea fiind ușor de manipulat.

---

<sup>15</sup> <https://materializecss.com/toasts.html>

<sup>16</sup> <https://ace.c9.io/>

<sup>17</sup> <https://gionkunz.github.io/chartist-js/>

<sup>18</sup> <https://www.mysql.com/>

<sup>19</sup> <https://www.sqlalchemy.org/>



Conform ultimului release (28 mai 2018), SQLAlchemy poate fi folosit pentru următoarele tipuri de baze de date<sup>20</sup>:

- Oracle
- Mysql
- PostgreSQL
- SQLite
- Microsoft SQL Server
- Sybase
- Firebird

## III.8 Docker

### III.8.1 Despre



21

**Figura 6: Logo-ul oficial Docker**

Docker<sup>22</sup> este o soluție software ce rezolvă problema virtualizării unui sistem de operare, tehnica purtând numele de *containerizare*. Este adesea folosit când vine vorba de deployment-ul unei aplicații ce are nevoie de diferite dependențe într-o singură unitate virtuală, acea unitate fiind deja configurată cu tool-urile necesare pentru a rula aplicația.

---

<sup>20</sup> <http://docs.sqlalchemy.org/en/latest/dialects/index.html>

<sup>21</sup> [https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/Docker\\_%28container\\_engine%29\\_logo.svg/2000px-Docker\\_%28container\\_engine%29\\_logo.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/Docker_%28container_engine%29_logo.svg/2000px-Docker_%28container_engine%29_logo.svg.png)

<sup>22</sup> <https://www.docker.com/>

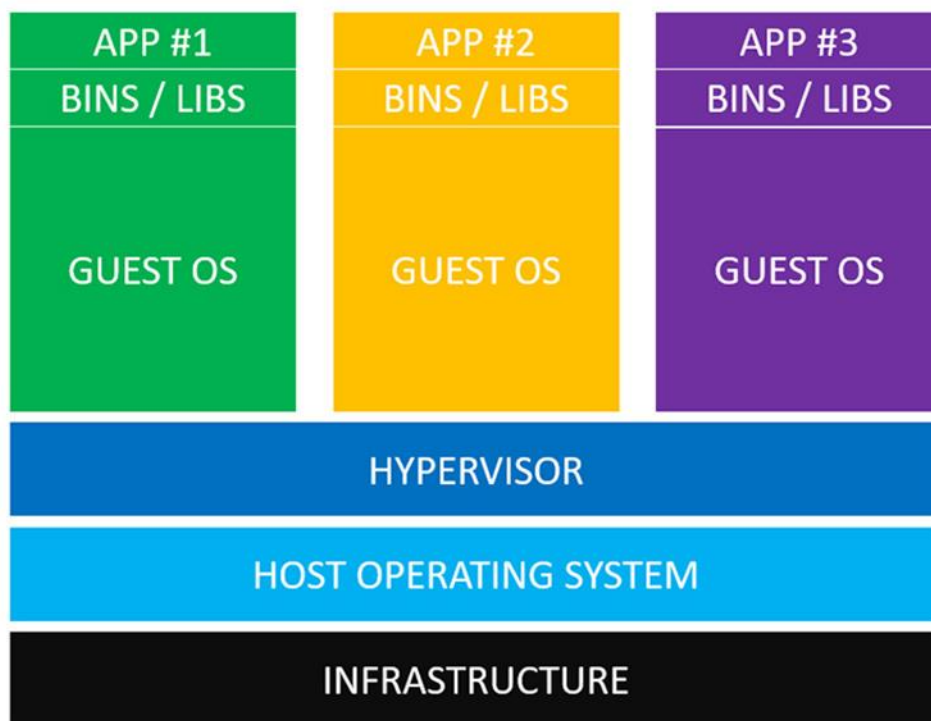
### III.8.2 Containere versus mașini virtuale

Deși din punct de vedere al funcționalității și al problemei pe care o rezolvă am putea spune că mașinile virtuale sunt asemănătoare cu soluția oferită de docker, mai precis containerizarea, aceste două unități virtuale diferă prin arhitectură. Containerele excelează când vine vorba de pornirea/oprirea unei unități, modificarea acesteia, salvarea unei stări etc., pe când mașinile virtuale consumă destul de mult timp pentru a executa aceste comenzi.

Pentru a exemplifica diferența de arhitectură dintre cele două soluții, vom considera un exemplu întâlnit în practică: găzduirea unui număr de aplicații, fiecare aflându-se pe o unitate virtuală separată.

- **Arhitectura unei aplicații găzduite pe mașini virtuale**

După cum se poate vedea în figura alăturată (vezi Figura 7), dacă am dori să găzduim aplicațiile pe câte o mașină virtuală separată, ar trebui să avem câte o imagine separată de sistem de operare pentru fiecare mașină, fiecare fiind configurată cu librăriile de care ar avea nevoie aplicația găzduită pe aceasta să ruleze.



23

**Figura 7: Arhitectura aplicațiilor găzduite pe multiple mașini virtuale**

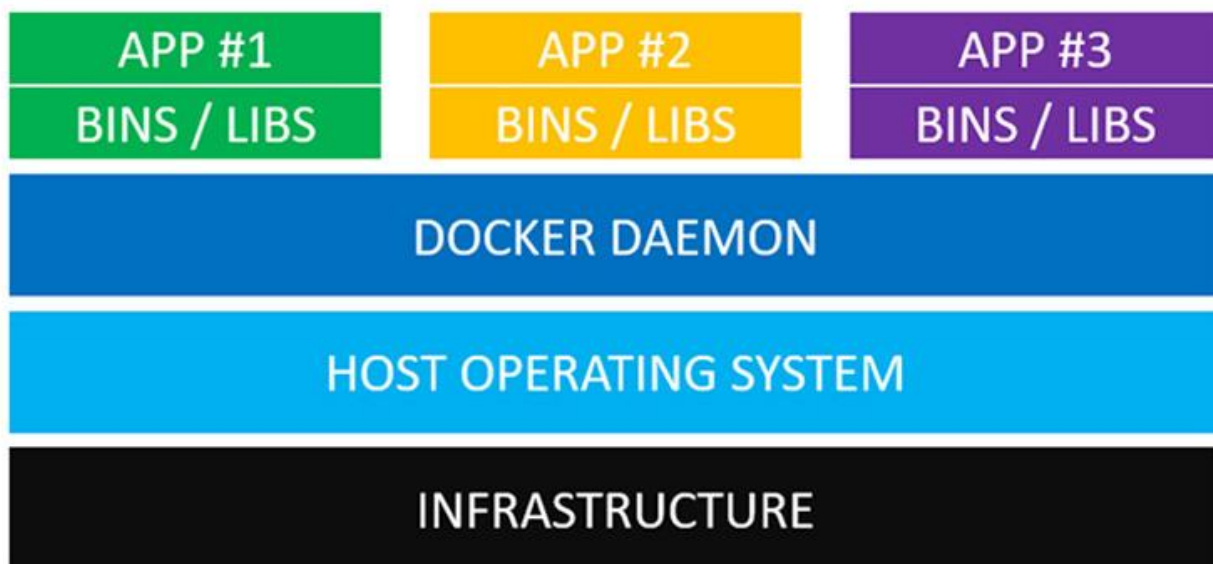
<sup>23</sup> <https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers>

Pentru a putea avea multiple instanțe de mașini virtuale, ar trebui să pornim *Hypervisor*-ul de pe mașina locală (Hyper-V pentru Windows, Hyperkit pentru MacOS și KVM pentru Linux), acest software având responsabilitatea de a distribui fiecărei instanțe de mașină virtuală resursele ei necesare, cum ar fi memorie internă, CPU etc.

- **Arhitectura unei aplicații găzduite pe containere**

Spre deosebire de arhitectura folosind mașini virtuale (vezi Figura 7), arhitectura folosind containerele oferite de docker (vezi Figura 8) diferă prin faptul că în loc de acel *Hypervisor* regăsim *Docker Daemon*, acesta fiind un serviciu ce rulează în background pe sistemul de operare al mașinii și manageriază întregul proces de virtualizare al containerelor și interacțiunile cu acestea.

23




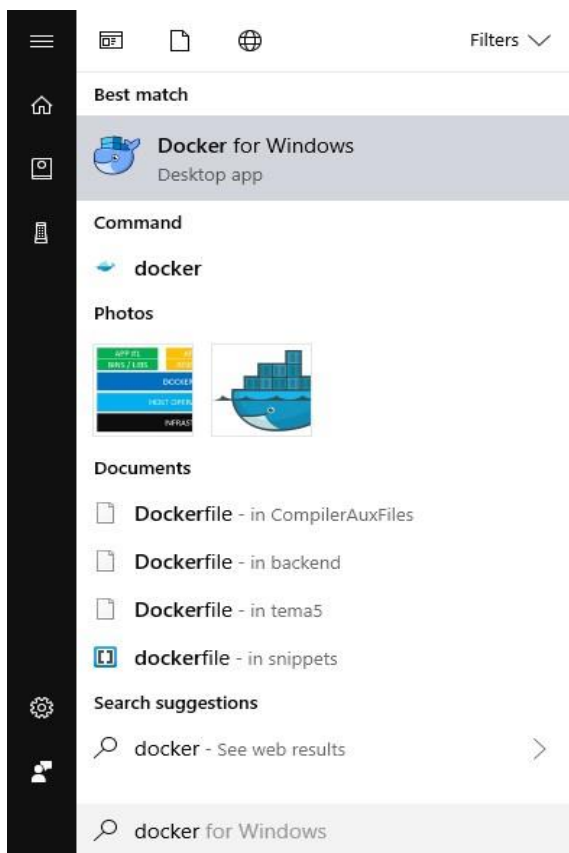
**Figura 8: Arhitectura aplicațiilor găzduite pe containere**

După cum se poate observa, librăriile și fișierele binare de care are nevoie fiecare aplicație nu mai sunt configurate pe un sistem de operare rulat pe o mașină virtuală. Aceste fișiere sunt instalate pe o imagine de docker, în cele din urmă daemon-ul rulând aceste imagini, după cum se poate vedea, aplicațiile rămân în continuare izolate una de cealaltă, fiecare având propria imagine de docker, în cele din urmă imaginea fiind instanțiată într-un container.

### **III.8.3 Instalare pe sistemul de operare Windows**

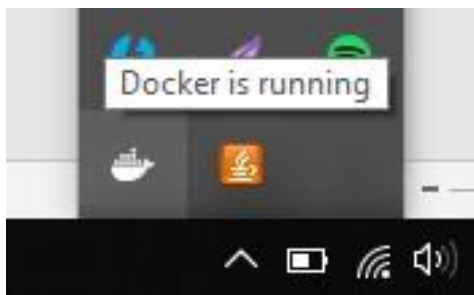
Pentru a începe lucrul cu tehnologia Docker pe sistemul de operare Windows (vom face referire la versiunea 10) o serie de pași sunt necesari de urmat:

1. Se descarcă kit-ul de instalare de la următoarea adresă:  
<https://store.docker.com/editions/community/docker-ce-desktop-windows>
2. Se urmăresc pașii din kit-ul de mai sus și se instalează în directorul dorit.
3. Apăsând pe butonul  și căutând “Docker” va trebui să găsiți executabilul precum în Figura 9, acest lucru însemnând că s-a instalat cu succes.



**Figura 9: Executabilul Docker instalat pe Windows**

4. Dacă în taskbar vă apare o iconiță identică cu cea din Figura 10, atunci docker rulează cu succes pe sistemul dvs. de operare.



**Figura 10: Docker rulează pe sistemul de operare**

### III.8.4 Utilizare Docker pe sistemul de operare local

Dacă ați reușit cu succes să instalați Docker, atunci puteți să-i testați comportamentul și funcționalitățile. De preferat este să executați comenzile specific docker utilizând Windows PowerShell (nu Windows PowerShell ISE), acesta având suport pentru comenzi de tip bash.

În continuare, vom prezenta comenzi uzuale ale docker-ului, pentru a testa unele funcționalități de bază:

- Pentru a verifica versiunea curentă de docker, rulați comanda:

```
$ docker -version
```

- Pentru a rula un container ce folosește o imagine aflată pe repository-ul celor de la Docker, sintaxa este următoarea:

```
$ docker run <numele_imaginii>
```

Așadar dacă vrem să instanțiem un container cu o imagine de ubuntu vom executa comanda:

```
$ docker run ubuntu bash
```

Opțional se vor adăuga atributele *--interactive* *--tty* pentru a rula alte comenzi de bash în interiorul acelei instanțe în manieră interactivă. Altfel, o imagine rulează doar cu un set de comenzi predefinite dinaintea execuției. De exemplu am putea configura o imagine să ruleze una sau mai multe comenzi de bash cum ar fi: ls, echo, whoami etc.

După cum am spus mai devreme, imaginile pentru dockere pot fi luate de pe un repository centralizat, însă putem crea imagini proprii pornind de la alte imagini, sau pornind de la zero, folosind fișiere de tip *Dockerfile*.

Dockerfile este un fișier de configurare ce stabilește dinainte comportamentul unei imagini Docker ce va fi rulată într-un container. Acesta poate fi scris astfel încât imaginea creată de utilizator poate moșteni caracteristici de la o imagine deja existentă, aceasta având posibilitatea de a fi extinsă, adăugând comportamente noi cum ar fi: instalarea unor librării adiționale, rularea unor comenzi specifice, restricționarea accesului unui utilizator etc.

Ca exemplu de fișier Dockerfile vom considera cel folosit în cadrul proiectului pentru a rula fișiere de tip python. Exemplu se poate vedea mai jos:

```
FROM python:3
ADD script.py /
CMD [ "python", "-u", "./script.py" ]
```

Pe scurt, fișierul Dockerfile din cadrul proiectului are următorul comportament:

- Prima comandă moștenește o imagine deja existentă, aceasta fiind un sistem de operare cu distribuție linux, având deja instalat Python versiunea 3.6 pe aceasta.
- Cea de-a doua comandă adaugă un fișier local de tip .py aflat în directorul curent.
- A treia comandă are rolul de a rula fișierul .py adăugat recent la cea de a doua comandă.

După ce am creat Dockerfile-ul, o imagine poate fi construită pe baza acestui fișier folosind următoarea sintaxă:

```
$ docker build -t <numele_imagini> <cale_fisier_Dockerfile>
```

Pentru a rula imaginea creată anterior într-un container, vom apela următoarea comandă:

```
$ docker run <numele_imagini>
```

Comportamentul containerului fiind unul simplu, acesta având rolul de a rula fișierul *script.py* și de a furniza rezultatul la consolă, după cum a fost definit în fișierul Dockerfile.

### III.9 Concluzii

Toate tehnologiile prezentate în acest capitol au avut un rol esențial în cadrul procesului de dezvoltare, fiecare rezolvând câte o subproblemă a problemei mari pe care o rezolvă aplicația. Flask fiind un instrument pe cât de ușor pe atât de folositor când vine vorba de dezvoltarea unei aplicații web îmbinându-se ușor cu alte extensii, iar Docker-ul fiind cea mai bună soluție pentru un mediul virtual lansat rapid pentru a rula cod nesigur provenit din surse străine.

## Capitolul IV: Detalii de implementare

În capitolul curent vor fi prezentate anumite detalii de implementare ce au fost puse în practica pentru a soluționa unele probleme ce au apărut pe parcursul dezvoltării aplicației, acestea fiind de fapt subprobleme ale problemei mari pe care o rezolvă aplicația.

### IV.1 Sistemul de evaluare Elo

Sistemul de evaluare Elo<sup>24</sup> a fost conceput de către Arpad Elo<sup>24</sup>, fost profesor de fizică la Universitatea Marquette din Milwaukee. Fiind unul dintre cei mai buni jucători de șah din Milwaukee în anii 1930, sistemul de evaluare conceput de acesta a fost implementat prima oară în turneele de șah începând cu anii 1970, acesta fiind folosit și la ora actuală de către USCF<sup>25</sup>.

Scopul acestui sistem de evaluare este de a determina un nivel în reprezentare numerică pentru utilizatorii din cadrul aplicației și de a putea fi recalculat pe baza acțiunilor și progresului acestora.

#### IV.1.1 Baza teoretică

Presupunem că avem două entități **A** și **B**, fiecare având un scor definit la un moment dat notate prin  $s(\mathbf{A})$  respectiv  $s(\mathbf{B})$ . Sistemul de evaluare Elo reușește să calculeze noul scor pentru ambele entități,  $s'(\mathbf{A})$  respectiv  $s'(\mathbf{B})$  în urma unei confruntări între cele două, folosindu-se de scorurile anterioare, formulele fiind bazate pe curbe logistice.

Pentru a înțelege mai bine cum se calculează scorul folosind sistemul de evaluare Elo, se va ilustra aceasta printr-un exemplu concret.

Fixăm scorurile entităților ca fiind:

$$s(\mathbf{A}) = 2400$$

$$s(\mathbf{B}) = 2000$$

Intuitiv deducem din aceste valori că entitatea **A** are mai multă experiență față de entitatea **B**, prin urmare ar fi de așteptat ca în urma unei confruntări, entitatea **A** să fie cea victorioasă.

În primă fază, se calculează probabilitatea cu care ar putea câștiga fiecare entitate într-o

---

<sup>24</sup> <https://dotsports.com/general/news/elo-ratings-explained-20565>

<sup>25</sup> [https://en.wikipedia.org/wiki/United\\_States\\_Chess\\_Federation](https://en.wikipedia.org/wiki/United_States_Chess_Federation)

confruntare, fiind notate cu  $e(A)$  respectiv  $e(B)$ , folosind următoarele formule:

$$e(A) = \frac{1}{1 + 10^{\frac{s(A) - s(B)}{400}}}$$

$$e(B) = \frac{1}{1 + 10^{\frac{s(B) - s(A)}{400}}}$$

Prin urmare vor rezulta valorile:

$$e(A) = 0.91$$

$$e(B) = 0.09$$

Este nevoie să definim un eveniment,  $X$  spre exemplu, care ia ca valori entitățile prezente în confruntare și returnează situația finală a confruntării pentru acea entitate.

$$X: \{A, B\} \rightarrow \left\{1, \frac{1}{2}, 0\right\}, \text{ unde: } Y \in \{A, B\} \text{ și:}$$

$$X(Y) = 1, \quad \text{entitatea } Y \text{ câștigă}$$

$$X(Y) = \frac{1}{2}, \quad \text{remiză}$$

$$X(Y) = 0, \quad \text{entitatea } Y \text{ pierde}$$

Presupunem că entitatea  $A$  a ieșit victorioasă din confruntare, adică:

$$X(A) = 1 \text{ și } X(B) = 0$$

Scorurile acestora se vor actualiza folosind următoarea formulă:

$$s'(A) = s(A) + K * (X(A) - e(A)) \quad (1)$$

$$s'(B) = s(B) + K * (X(B) - e(B)) \quad (2)$$

Termenul  $K$  din formula precedentă reprezintă o constantă numită *factorul*  $K$ , ce are un rol esențial pentru a scala impactul unei confruntări între două entități. Cu alte cuvinte, cu cât *factorul*  $K$  este mai mic, cu atât scorurile entităților vor diferenția mai puțin, acestea rămânând mai stabile, iar dacă *factorul*  $K$  are asignată o valoare mare, atunci scorurile se vor schimba drastic de la o



confruntare la alta.

În practică, valoarea  $K = 32$  este folosită de către ICC<sup>26</sup>, însă nu există o valoare general stabilită. În continuare, vom considera în exemplul curent valoarea lui  $K = 32$ .

După aplicarea formulelor (1) și (2) vom deduce noile valori pentru scorul entităților, după o confruntare în care entitatea A a ieșit victorioasă:

$$s'(A) = 2403$$

$$s'(B) = 1997$$

Pornind de la scorurile inițiale, putem observa că valorile noi calculate după o victorie a entității A, nu diferă cu mult, ceea ce semantic înseamnă că era de așteptat ca entitatea A să câștige, așadar nu putem concluziona că a câștigat multă experiență din confruntarea cu entitatea B, aceasta fiind una slabă față de A.

Însă dacă am fi presupus de la început că entitatea B, care este cu mult inferioară entității A din punct de vedere al scorului, ar fi câștigat confruntarea, atunci scorurile finale ar fi arătat astfel:

$$s'(A) = 2371$$

$$s'(B) = 2029$$

Din această situație deducem faptul că entitatea B, care ar fi avut probabilitatea de câștig  $e(B) = 0.09$ , a acumulat multă experiență în urma unei confruntări cu un oponent care era clar favorizat să câștige, fiind o surpriză faptul că aceasta a ieșit victorioasă.

#### IV.1.2 Integrarea sistemului de evaluare în cadrul aplicației

În cadrul aplicației, am abstractizat noțiunea de *confruntare* între două entități, aceasta fiind de fapt încercarea unui utilizator de a rezolva o problemă de programare sugerată de către aplicație. Fiecare entitate (utilizator, lecție, exercițiu) începe cu o valoare prestabilită a scorului, acestea suferind modificări pe parcursul utilizării aplicației în funcție de progresul utilizatorului.

De exemplu, dacă utilizatorul nu reușește să completeze o lecție cu succes, atunci vom stabili că lecția a *câștigat* aceea confruntare, aceștia adăugându-se puncte la scorul său, reprezentând dificultatea lecției pentru acel utilizator. Cu cât aceeași lecție nu este rezolvată cu succes de către utilizator, cu atât devine mai dificilă pentru el, însă datorită sistemului de evaluare

---

<sup>26</sup> <https://www.chessclub.com/>

Elo, dificultatea acesteia nu va crește constant până când va deveni imposibil pentru utilizator să o rezolve, ci va crește treptat și din ce în ce mai puțin.

Prin urmare, în urma execuției unei secvențe de cod într-o unitate virtuală de tip Docker se va primi rezultatul rulării într-un Pipe reprezentat printr-o structură de date de tip tuplă, având doua câmpuri:

- Mesajul din STDOUT
- Mesajul din STDERR

```
def get_result_from_execution(self, source_code):  
    result = self._compiler_repository.evaluate_simple_code_submission(source_code)  
  
    # if the stdout is empty that means the source code submitted has errors  
    if not result[0] or not result[0].strip():  
        return _format_output_error_message(result[1]), 500  
    # else, the code has compiled and executed successfully  
    else:  
        return result[0].decode('utf-8'), 200  
  
def evaluate_submission(self, source_code, source_id):  
  
    result = self.get_result_from_execution(source_code)  
  
    if result[1] == 200:  
        self._elo_rating_repository.user_wins_over_lesson(source_id)  
    else:  
        self._elo_rating_repository.lesson_wins_over_user(source_id, None)  
  
    return result
```

Așadar, dacă mesajul în urma execuției se află pe poziția 1 din tuplă atunci codul nu a rulat cu succes, însemnând că utilizatorul nu a reușit să completeze cu succes lecția, fiind apelate funcții din *EloRatingRepository* pentru a ajusta scorurile entităților.

Odată ce o lecție este completată de către utilizator, atunci nu se vor mai schimba scorurile indiferent de rezultatul unei încercări. Am ales această abordare pentru a evita situația când utilizatorul găsește o soluție pentru o problemă și o va trimite de mai multe ori doar pentru a câștiga experiență.

Pe de altă parte, dacă o lecție devine dificilă pentru utilizator, însă între timp acesta câștigă experiență rezolvând probleme la modulul *Exerciții*, din punct de vedere semantic, deducem că în

urma progresului făcut la exerciții, lecția la care avea probleme s-ar putea să nu mai fie un impediment, așadar, dacă o rezolvă, nu va mai câștiga atât de multă experiență ca în trecut.

#### IV.1.3 Adaptarea la rezolvarea unor funcții

Ce s-a prezentat până acum a fost bazat pe o presupusă confruntare în care o problema avea o singură soluție. De exemplu: dacă o lecție ar fi cerut utilizatorului lucruri cum ar fi afișarea primelor 10 numere pare sau lucruri asemănătoare, se știa că este vorba doar de o soluție, iar dacă secvența de cod trimisă de utilizator nu avea output-ul dorit, atunci confruntarea era câștigată de lecție.

Odată cu implementarea modului *Exerciții*, unde toate soluțiile pentru problemele propuse sunt funcții, a intervenit o problemă în cadrul actualizării scorului. Soluția cerută de la utilizator fiind o funcție, aceasta ar trebui să fie testată folosind mai multe date de test comparând rezultatele funcției trimise de utilizator cu rezultatele funcției corecte, rezultând un factor care reprezintă câte date de test a reușit funcția trimisă de utilizator să îndeplinească.

În cazul în care soluția trimisă de utilizator nu are niciun rezultat corect, atunci este clar că acesta a dat greș în totalitate în rezolvarea problemei, însă dacă acesta ar avea 90% din teste acoperite, nu ar fi corect să actualizăm scorurile ca și cum nu ar fi reușit să rezolve acea problemă, când de fapt aceasta a fost rezolvată aproape în întregime.

Soluția care rezolvă această inconsistență constă în a pondera câștigul (respectiv pierderea) utilizatorului cu raportul de teste rezolvate de acesta.

Spunem că problema care are ca soluție o funcție are în total  $n$  teste de validare. Considerăm variabila  $t$  ca fiind factorul de test cu  $t \in [0,1]$ , prin urmare:

$t = 0 \rightarrow$  *utilizatorul nu a rezolvat deloc problema*

$t \in (0,1) \rightarrow$  *utilizatorul a completat parțial problema*

$t = 1 \rightarrow$  *utilizatorul a completat total problema*

În cazurile în care  $t \in \{0,1\}$  atunci actualizarea scorurilor se va face în modul uzual prezentat până acum.

În cazul în care  $t \in (0,1)$  deși am notat că utilizatorul a completat parțial problema, în sistemul de evaluare Elo nu există stări intermediare, așa că vom considera că utilizatorul este cel care a pierdut confruntarea, iar punctele se vor ajusta folosind următorul algoritm:

**Input:**  $s(w), s(l), t$ , unde:

$s(w) \rightarrow$  scorul actual al câștigătorului

$s(l) \rightarrow$  scorul actual al pierzătorului

$t \rightarrow$  factorul de test

**Output:**  $s'(w), s'(l)$ , unde:

$s'(w) \rightarrow$  noul scor pentru câștigător aplicând ajustarea

$s'(l) \rightarrow$  noul scor pentru pierzător aplicând ajustarea

**begin**

#  $computeElo(scor_{câștigător}, scor_{pierzător})$

# returnează scorurile calculate cu formula normală

$elo_w, elo_l = computeElo(s(w), s(l))$

$diferențăDePuncte = |s(w) - elo_w|$

$s'(w) = s(w) + diferențăDePuncte - diferențăDePuncte * t$

$s'(l) = s(l) - diferențăDePuncte + diferențăDePuncte * t$

**end**

Se poate observa că în funcție de factorul de test  $t$ , scorul se va ajusta atât pentru câștigător cât și pentru pierzător.

Luând un exemplu concret având următoarele valori de intrare:

$$s(w) = 1700$$

$$s(l) = 2000$$

$$t = \frac{4}{5}$$

Valorile calculate folosind algoritmul de ajustare vor avea următoarele valori:

$$s'(w) \approx 1701$$

$$s'(l) \approx 1998$$

Lucru care reflectă mai mult realitatea decât dacă am fi aplicat calculele clasice având valorile:

$$s'(w) \approx 1708$$

$$s'(l) \approx 1991$$

Utilizatorul ieșind într-un dezavantaj destul de mare deși acesta a rezolvat în proporție de 90% problema.

## IV.2 Modulul de exerciții

Modulul de exerciții din cadrul aplicației are rolul de a testa cunoștințele dobândite de utilizator până într-un moment dat pe baza modului în care au fost rezolvate lecțiile din cadrul unui capitol.

### IV.2.1 Structura unui exercițiu

Exercițiile sunt asemănătoare unei lecții, în sensul poziționării chenarelor cu explicații și a editorului de text însă de data aceasta orice rezolvare a unei probleme este reprezentată de o funcție. Utilizatorul va avea completat implicit de către aplicație antetul funcției, iar acesta va trebui să implementeze funcția conform cerinței (vezi Figura 15).

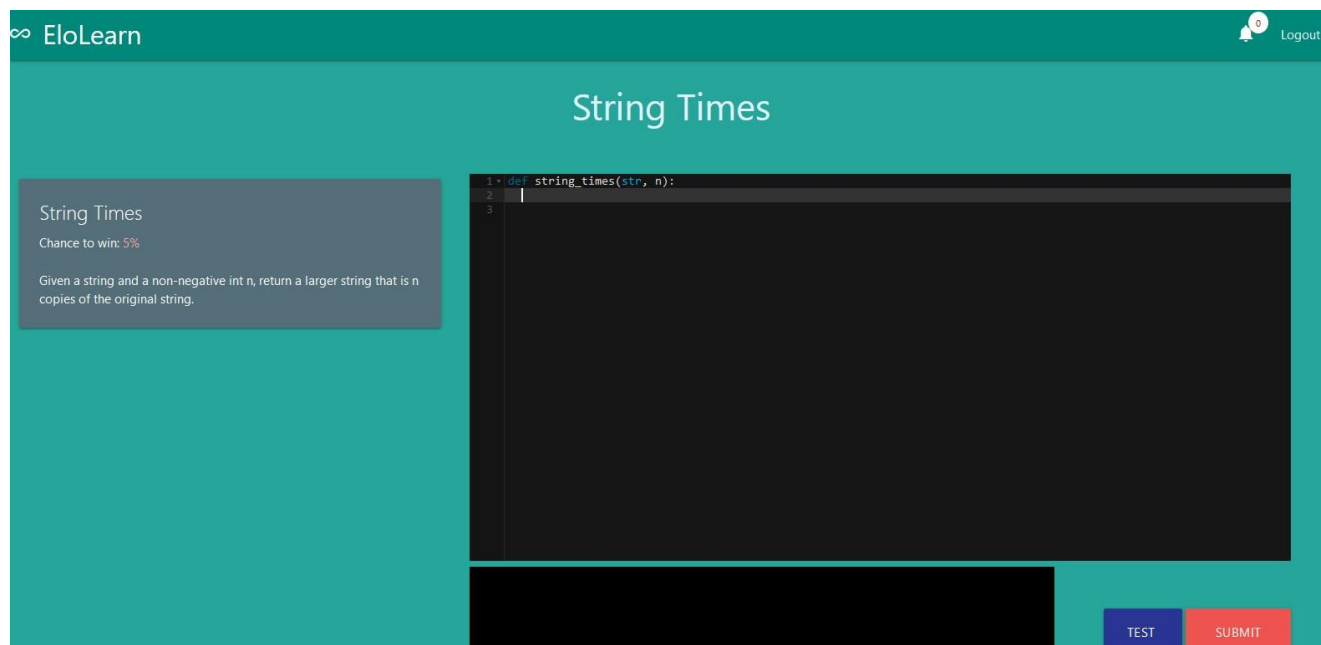


Figura 15: Interfața unui exercițiu în cadrul aplicației

Desigur, acesta are posibilitatea de a-și testa soluția înainte să o trimită, aceasta rămânând stocată în baza de date după apăsarea butonului *Test*. Acest lucru vine în ajutorul utilizatorului atunci când vine cu o soluție parțială ce rezolvă doar o parte din datele de test și dorește să revină în viitor să rezolve total problema.

#### IV.2.3 Recomandarea unui exercițiu

Spre deosebire de modulul *Lecții* unde utilizatorul trebuie să parcurgă iterativ lecțiile din cadrul unui capitol, la modulul *Exerciții* în momentul în care utilizatorul selectează acest modul din dashboard-ul principal al aplicației, pe partea de server se calculează istoricul acestuia la fiecare capitol parcurs de până acum și se recomandă un exercițiu pentru acesta. Exercițiul are legătură doar cu Capitolul, la fel și Lecția (vezi Figura 16).

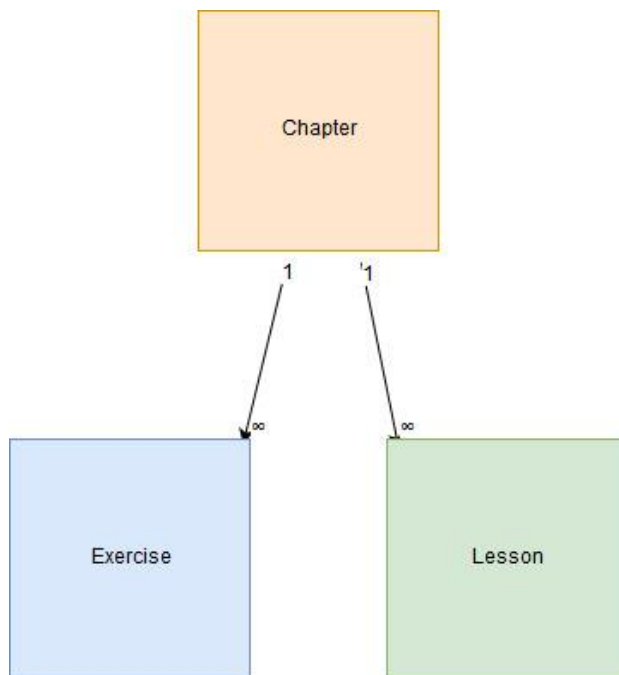


Figura 16: Relațiile între Capitol, Lecție și Exercițiu

Pe scurt acesta recomandă un exercițiu utilizatorului, pe baza unei euristici. Exercițiul din capitolul (completat de către utilizator) care are în medie cele mai multe lecții ce și-au mărit dificultatea pentru utilizator pe parcurs. Practic se vor selecta capitolele în care utilizatorul a întâmpinat dificultăți și se va selecta un exercițiu din cadrul acelu capitol. Un capitol are mai multe exerciții, însă va avea prioritate exercițiul la care utilizatorul ar avea cele mai multe șanse de rezolvare, bazându-ne pe actualul său scor și dificultatea exercițiului.

### **IV.3 Concluzii**

Capitolul curent are rolul de a prezenta detaliile de implementare ce stau în spatele aplicației cum ar fi raționamentul care a fost folosit în unele situații cât și detaliera sistemului de evaluare folosit, ce a avut un impact mare în ideea aplicației și a făcut posibilă măsurarea progresului unui utilizator.

## **Concluziile lucrării și direcții viitoare**

### **Concluzii generale**

Învățarea online a unor limbaje de programare a devenit în ultima vreme destul de comună, o mare parte din lumea dornică să învețe programare apelând la platformele online ce le stau la dispoziție, fiecare având o trăsătură specială și un mod unic de a atrage utilizatori prin interactivitate. După cum am evidențiat și în primul capitolul ce prezintă aplicații similare, multe platforme cer utilizatorilor, pentru o experiență mai plăcută, înregistrarea unui cont de tip premium ce solicită a fi plătit sau oferă cursuri ce necesită în primă fază instalarea și configurarea a unor medii de dezvoltare, kit-uri de dezvoltare etc., lucru care poate fi destul de descurajator la început.

Aplicația dezvoltată are rolul de a oferi utilizatorilor un mediu izolat de sistemul lor de operare unde își pot testa propriul cod, învățând noțiuni de programare într-un mod interactiv prin rezolvarea unor mici cerințe, însă având și posibilitatea de rezolvare a unor exerciții ce sunt recomandate pe baza progresului în cadrul procesului de învățare. Toate acțiunile acestora fiind reprezentate printr-un nivel reprezentat în valoare numerică, acesta fluctuând pe baza progresului din cadrul procesului de învățare și de rezolvare a problemelor. Acest progres este transparent și afișat pe pagina personală a utilizatorului, pentru ca acesta să își facă o idee despre cum se descurcă în cadrul aplicației.

În opinia mea platformele din aceeași categorie au un impact destul de mare asupra utilizatorilor deoarece prin tehnologiile disponibile la ora actuală pot oferi o experiență unică în procesul de învățare al noțiunilor dintr-un domeniu, acestea oferind posibilitatea de a învăța într-un mod interactiv. În plus, aspectul de joc al unei platforme de învățare ajută la atragerea utilizatorului și stârnește interesul acestuia prin feedback-ul constant oferit prin elemente din zona jocurilor cum ar fi: insigne, cuantificarea utilizatorului printr-un nivel, senzația de competiție etc.

### **Îmbunătățiri și direcții viitoare**

Aplicația poate crește în complexitate dacă ar fi implementat un agent inteligent artificial care ar putea sesiza în timp real greșeli de scriere în codul utilizatorului, ba chiar să recomande moduri de scriere a unor secvențe de cod sau folosirea unor soluții ce ar putea fi mai eficiente. În plus ar putea să răspundă la întrebările utilizatorilor, oferind suport non stop pentru aceștia, acesta având un rol de tutore.



O altă îmbunătățire ar fi suportul pentru diferite limbaje de programare, fiecare având probleme specifice de rezolvat.

Prin faptul că aplicația folosește în spate sistemul de evaluare Elo, folosit de asemenea și în sporturi și jocuri video, utilizarea acesteia ar putea fi extinsă în cadrul unui curs din cadrul unei facultăți, această reprezentând una din componentele practice ce necesită a fi promovată din cadrul cursului. Nota de la această componentă reprezentând nivelul la care a ajuns utilizatorul la finalul unei perioade stabilite.

Progresul unui utilizator fiind transparent, acesta ar putea fi revizuit de către titularul de curs, acesta observând unde au fost punctele slabe ale studentului. Titularul de curs ar putea realiza statistici pe baza progresului fiecărui student, văzând ce lecții au fost mai dificile sau ce subiecte au fost prea ușoare.

# Bibliografie

- [1] Runnable, „**Dockerize your Python Application,**” 2018.  
<https://runnable.com/docker/python/dockerize-your-python-application>. [Accesat 2018]
- [2] P. Guo, „**Python Is Now the Most Popular Introductory Teaching Language at Top U.s. Universities,**” 7 Iulie 2014.  
<https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext>. [Accesat 2018]
- [3] R. Yammouni, „**Are there any rating systems (similar to Elo) commonly used in single player games?,**”  
<https://www.quora.com/Are-there-any-rating-systems-similar-to-Elo-commonly-used-in-single-player-games>. [Accesat 2018]
- [4] „**elo 0.1.1,**”  
<https://pypi.org/project/elo/>. [Accesat 2018].
- [5] M. Bekhelouf, „**Login / register material design form,**”  
<https://codepen.io/malikbekhelouf/pen/Kzwgbb>. [Accesat 2018].
- [6] metinmediamath, „**How to calculate the ELO-RATING,**”  
<https://metinmediamath.wordpress.com/2013/11/27/how-to-calculate-the-elo-rating-including-example/>. [Accesat 2018].
- [7] N. Janetakis, „**Comparing Virtual Machines vs Docker Containers,**”  
<https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers>. [Accesat 2018].
- [8] Codingbat,  
<http://codingbat.com/python>. [Accesat 2018].
- [9] C. Swaroop, **A byte of Python.**

- [10] Microsoft, „**TrueSkill**,”  
<http://trueskill.org/>. [Accesat 2018].
- [11] Wikipedia, „**Elo Rating System**,”  
[https://en.wikipedia.org/wiki/Elo\\_rating\\_system](https://en.wikipedia.org/wiki/Elo_rating_system). [Accesat 2018].
- [12] „**Migrate SQLAlchemy Databases with Alembic**,”  
<https://www.pythoncentral.io/migrate-sqlalchemy-databases-alembic/>. [Accesat 2018].
- [13] Ace, „**Embedding Ace in Your Site**,”  
<https://ace.c9.io/#nav=embedding>. [Accesat 2018].
- [14] SQLAlchemy, „**Query API**,”  
<http://docs.sqlalchemy.org/en/latest/orm/query.html>. [Accesat 2018].
- [15] J. Hu, „**100+ Python challenging programming**,”  
<https://github.com/zhiwehu/Python-programming-exercises/blob/master/100%2B%20Python%20challenging%20programming%20exercises.txt>.  
[Accesat 2018].
- [16] Materialize, „**Getting Started**,”  
<https://materializecss.com/getting-started.html>. [Accesat 2018].