

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE ADMINISTRAȚIE ȘI AFACERI
SPECIALIZAREA CIBERNETICĂ ECONOMICĂ

LUCRARE DE LICENȚĂ

COORDONATOR ȘTIINȚIFIC

Profesor Universitar Doctor OANCEA BOGDAN

ABSOLVENT(Ă),
TAȘCĂ LAURA-ELENA

BUCUREȘTI

2024

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE ADMINISTRAȚIE ȘI AFACERI
SPECIALIZAREA CIBERNETICĂ ECONOMICĂ

**APLICAȚIE WEB PENTRU GESTIONAREA
ORELOR DE CURS ȘI SEMINAR ȘI
INDICAȚII DE ORIENTARE ÎN
FACULTATEA DE MATEMATICĂ ȘI
INFORMATICĂ**

COORDONATOR ȘTIINȚIFIC

Profesor Universitar Doctor OANCEA BOGDAN

ABSOLVENT(Ă),
TAȘCĂ LAURA-ELENA

BUCUREȘTI

2024

Cuprins

1. Introducere	5
2. Descrierea tehnologiilor folosite.....	7
2.1 Tehnologii de back-end	7
2.1.1 Python	7
2.1.2 Django	7
2.1.3 Baze de date și MySQL.....	9
2.1.4 Docker	10
2.2 Tehnologii de front-end.....	11
2.2.1 HTML	11
2.2.2 CSS	11
2.2.3 JavaScript	11
2.2.4 Bootstrap.....	12
2.3 Algoritmul “Breadth First Search”	13
2.4 Testul de orientare.....	14
2.5 Descrierea arhitecturii client-server.....	16
2.5.1 Backend-ul (Serverul).....	16
2.5.2 Frontend-ul (Clientul).....	16
2.5.3 Comunicarea Client-Server.....	17
3. Partea practică	18
3.1 Arhitectura aplicației.....	18
3.1.1 Model.....	18
3.1.2 View	22
3.1.3 Template.....	30
3.2 Rolurile utilizatorilor din aplicație	32
3.2.1 Utilizator neautentificat.....	32
3.2.2 Utilizator autentificat de tip student	33
3.2.3 Utilizator autentificat de tip profesor	33
3.2.4 Administratorul.....	34
3.3 Determinarea indicațiilor de orientare.....	34
3.3.1 Structura Fișierului.....	35
3.3.2 Descrierea clasei Graph și funcționalitățile sale.....	36
3.4 Pașii pentru configurarea inițială a aplicației	41
3.5 Fluxul de acțiuni al aplicației	43
3.5.1 Pagina de înregistrare.....	43
3.5.2 Pagina de login	44

3.5.3	Pagina de home.....	45
3.5.4	Pagina Directions	46
3.5.5	Pagina Activity.....	47
3.5.6	Pagina Grades	49
4.	Concluzie.....	51
5.	Bibliografie	52

1. Introducere

Proiectul meu de licență constă într-o aplicație web denumită MathInfo Academy destinată Facultății de Matematică și Informatică a Universității din București, având scopul de a îmbunătăți considerabil experiența studenților și profesorilor prin facilitarea gestionării materialelor didactice, notelor și orientării în clădirea facultății.

Motivația principală care a stat la baza alegerii acestei teme a fost combinația între nevoia personală și observațiile asupra dificultăților întâmpinate de alți studenți. Aceasta a fost întărită de discuțiile avute cu prietenii mei de la Facultatea de Matematică și Informatică (FMI) care mi-au povestit adesea despre dezorganizarea care există în modul în care profesorii încarcă materialele și anunță notele, folosind platforme diferite și metode variate de comunicare, ceea ce complică semnificativ procesul de accesare a acestor resurse, creând confuzie pentru studenți.

Pe de altă parte, de-a lungul anilor de studii, atât eu, cât și colegii mei de la Facultatea de Administrație și Afaceri am întâmpinat dificultăți considerabile în găsirea sălilor de curs, mai ales din cauza lucrărilor de renovare asupra clădirilor diferitelor facultăți. Aceste lucrări au determinat Facultatea de Matematică și Informatică să găzduiască cursurile altor facultăți, inclusiv majoritatea cursurilor specializării mele. În cei trei ani de studiu, eu alături de colegii mei, dar și profesorii de la Cibernetică Economică, am întâmpinat frecvent dificultăți în găsirea sălilor din FMI, ceea ce a dus la întârzieri și scurtarea timpului de prezentare a materiei, necesară evaluării. Chiar și studenții de la FMI aveau probleme în găsirea sălilor, mai ales cei din primul an, ceea ce subliniază necesitatea unei soluții eficiente de orientare în facultate.

Aplicația dezvoltată își propune să ofere soluții inovatoare care să rezolve dificultățile întâmpinate, atât pentru orientarea în clădirea Facultății de Matematică și Informatică a oricărui membru sau vizitator al Universității din București, cât și pentru accesul la materialele de curs și la notele destinate studenților din FMI. O astfel de aplicație ar aduce beneficii semnificative nu doar studenților, cât și profesorilor, centralizând informațiile și simplificând procesele de învățare și de organizare academică.

Lucrarea este structurată în două părți principale: partea teoretică și partea practică, fiecare organizată în capitole și subcapitole ce descriu în detaliu aspectele esențiale ale dezvoltării și funcționalităților aplicației web.

Partea teoretică este compusă din mai multe subcapitole, incluzând **Tehnologii de Backend, Tehnologii de Frontend și Arhitectura Client-Server**. În subcapitolul **Tehnologii de Backend**, sunt prezentate limbajele de programare și serviciile utilizate pentru dezvoltarea părții de server a aplicației. Subcapitolul **Tehnologii de Frontend** descrie instrumentele și limbajele utilizate pentru dezvoltarea interfeței utilizatorului, explicând modul în care acestea oferă o experiență intuitivă și atractivă pentru utilizatori. Subcapitolul **Arhitectura Client-Server** oferă o perspectivă detaliată asupra modului în care componentele client și server interacționează pentru a asigura o funcționalitate coerentă și eficientă a aplicației.

Partea practică se concentrează pe implementarea concretă a aplicației și este împărțită în mai multe subcapitole. Subcapitolul **Arhitectura aplicației MVT** descrie structura de bază a aplicației, explicând modul în care sunt organizate modelele, vizualizările și șabloanele pentru a asigura o separare clară a responsabilităților și o dezvoltare facilă. În subcapitolul **Rolurile utilizatorilor**, sunt prezentate diferitele roluri pe care utilizatorii aplicației le pot avea și modul în care aceste roluri influențează accesul și interacțiunea cu aplicația. Subcapitolul **Determinarea indicațiilor de orientare** explică metodele folosite pentru a oferi indicații precise de orientare în facultate, utilizând structuri de date și algoritmi specifici. Subcapitolul **Pașii de configurare a aplicației** oferă detalii clare despre procesul de instalare și configurare a aplicației. În subcapitolul **Flow-ul aplicației**, sunt descrise în detaliu fluxurile de utilizare și interacțiune, ilustrând modul în care utilizatorii navighează și utilizează diferitele funcționalități ale aplicației.

2. Descrierea tehnologiilor folosite

2.1 Tehnologii de back-end

2.1.1 Python

Python este numele unuia dintre cele mai populare limbaje de programare de nivel înalt, orientat pe obiecte, codul specific acestuia fiind interpretat, nu compilat direct în cod mașină. Acesta a fost lansat de Guido van Rossum în anul 1991, drept un înlocuitor al limbajului de programare ABC (Stan, 2022, p.19). Este renumit pentru faptul că prezintă o sintaxă simplă și ușor de învățat, punând accent pe lizibilitate și reducând astfel costul de întreținere a programelor. Limbajul suportă module și pachete, încurajând modularitatea programului și reutilizarea codului. Este, de asemenea, cunoscut pentru biblioteca standard cuprinzătoare cu care vine la pachet. Pe lângă aceasta, folosind diverse managere de pachete (package managers), precum PIP, se pot instala diverse alte pachete, dintre care unele oferă suport pentru dezvoltarea aplicațiilor web. Flask și Django sunt cele mai populare astfel de unelte ce facilitează dezvoltarea aplicațiilor web folosind limbajul de programare Python (Stan, 2022, p.21).

2.1.2 Django

Django este un framework destinat construirii aplicațiilor web, având la bază limbajul Python, care încurajează dezvoltarea rapidă și design-ul curat, pragmatic. Acest framework a fost creat în anul 2003 într-o agenție de presă din Lawrence, Kansas. Scopul său este de a crea site-uri web dinamice, folosind Python. Codul sursă al acestui framework a fost publicat sub agenție BDS în anul 2005, iar în anul 2008 a fost creată Django Software Foundation pentru a susține și avansa dezvoltarea acestui framework, urmând ca versiunea 1.00 a framework-ului să fie publicată câteva luni mai târziu. Acest framework adoptă paradigma MVT (Model-View-Template), ceea ce permite dezvoltatorilor să construiască o arhitectură coerentă. Până în anul 2013, Django era compatibil doar cu versiunile Python 2.x, dar lansarea versiunii 1.5 a framework-ului, pe 26 februarie 2013, a marcat începutul compatibilității cu Python 3. În ziua de astăzi, organizații mari precum Instagram, Mozilla.org și Openstack.org folosesc Django pentru dezvoltarea site-urilor lor. (Dauzon, Bendoraitis & Ravindran, 2016, p. 5)

Framework-ul Django prezintă o mulțime de avantaje, dintre care cele mai notabile sunt următoarele:

- **Arhitectura Model-View-Template (MVT):** Django adoptă o abordare de tip Model-View-Template (MVT). Aceasta constă în trei componente esențiale: Model, View și Template. Modelul se ocupă de manipularea datelor în baza de date, View-ul gestionează logica aplicației și interacționează cu modelul, iar Template-ul are rolul de prezentare a interfeței utilizatorilor. Această separare a responsabilităților face codul mai clar și ușor de întreținut.
- **Securitate ridicată:** Django oferă funcționalități de securitate avansate, cum ar fi protecția împotriva atacurilor de tip SQL injection și cross-site scripting (XSS). De asemenea, are o comunitate activă care monitorizează și rezolvă rapid problemele de securitate.
- **Suport pentru mai multe baze de date:** Django suportă oficial patru baze de date: PostgreSQL, MySQL, SQLite și Oracle. Aceasta oferă dezvoltatorilor libertatea de a alege tipul de bază de date ce consideră că se potrivește mai bine aplicației construite.
- **Object Relational Mapper (ORM) inclus:** Framework-ul folosește un ORM pentru a abstractiza interacțiunile cu baza de date utilizată. Acest lucru îl scutește pe programator de la a scrie direct cod SQL pentru a manipula baza de date, oferind modalități alternative pre-implementate în limbajul Python.
- **Panoul de administrare implicit:** Django vine cu un panou de administrare predefinit, care permite gestionarea ușoară a datelor din baza de date. Acesta poate fi personalizat pentru a se potrivi nevoilor fiecărui proiect.
- **Biblioteci bogate:** Django include o serie de biblioteci utile pentru diverse sarcini, cum ar fi manipularea imaginilor, gestionarea sesiunilor, autentificarea utilizatorilor și multe altele.
- **Dezvoltare rapidă:** Datorită funcționalităților predefinite și a arhitecturii bine gândite, dezvoltarea în Django este rapidă și eficientă.
- **Comunitate activă:** Există o comunitate mare și dedicată de dezvoltatori Django, care oferă suport, tutoriale și exemple de cod.

Desigur, ca orice unealtă tehnologică, Django are și dezavantaje, cum ar fi o perioadă inițială de învățare, modificări de sintaxă între versiuni sau structura impusă de către framework ce poate nu se potrivește anumitor proiecte. Cu toate acestea, documentația bogată și sprijinul comunității fac ca aceste inconveniențe să fie ușor de depășit. (Dauzon, Bendoraitis & Ravindran, 2016, pp. 7-8)

2.1.3 Baze de date și MySQL

Conceptul de bază de date se referă la o colecție organizată și structurată de informații esențiale pentru gestionarea eficientă a datelor în era digitală. Aceste date pot fi structurate în diferite moduri, dar scopul principal este de a facilita gestionarea, accesul și procesarea datelor într-un mod eficient și ordonat. În zilele noastre, bazele de date sunt esențiale pentru stocarea și manipularea cantităților mari de informații într-un mod care permite accesul eficient și precis la datele dorite. Acestea sunt utilizate în aproape toate domeniile, de la afaceri și guvernare, până la știință și tehnologie. (Documentația oficială MySQL)

MySQL este un exemplu de sistem de gestionare a bazelor de date (SGBD), lansat în anul 1995, care are un rol important atât în utilizarea independentă dar și în integrarea cu alte aplicații. Scopul inițial al acestui SGBD a fost acela de a gestiona baze de date de dimensiuni semnificative mult mai eficient decât alte variante existente pe piață la momentul respectiv, dovedindu-se a fi un succes în acest sens. MySQL Server funcționează în sisteme client/server sau încorporate. Într-un sistem client/server, serverul realizează diferite funcționalități, iar pentru persistența informațiilor sunt utilizate bazele de date. (Documentația oficială MySQL)

Bazele de date MySQL sunt relaționale, ceea ce înseamnă că datele sunt stocate în tabele separate. Organizarea datelor în fișiere fizice este optimizată pentru o viteză cât mai mare la interogări. SQL (Structured Query Language) este cel mai cunoscut pentru interogarea bazelor de date, fiind un standard cu mai multe implementări, inclusiv pentru MySQL. Software-ul MySQL este open-source, ceea ce înseamnă că oricine are posibilitatea de a vizualiza, utiliza și chiar modifica codul produsului software în mod gratuit, fără a încălca vreun drept de autor. (Documentația oficială MySQL)

Bazele de date MySQL pot fi utilizate într-o varietate de scenarii, iar aici sunt câteva dintre cele mai notabile sunt aplicațiile web și cele la nivel de întreprindere. MySQL este frecvent folosit ca bază de date pe partea de backend a aplicațiilor web, inclusiv a sistemelor de gestionare a conținutului (CMS), platformelor de comerț electronic și rețelelor sociale. De

asemenea, utilizarea acestuia în aplicațiilor la nivel de întreprindere este favorizată de suportul direct oferit, dar și de faptul că setul său de caracteristici acoperă majoritatea nevoilor unei astfel de aplicații (Suehring, 2002, p.18).

Printre numeroasele avantaje pe care acest sistem de gestiune a bazelor de date le prezintă, cele mai importante sunt următoarele (Suehring, 2002, p.18):

Suport open-source: MySQL este open-source, ceea ce înseamnă că oricine poate descărca și extinde codul pentru a satisface propriile nevoi.

Încărcare redusă: MySQL poate rula confortabil pe sisteme cu resurse limitate, cum ar fi un computer Intel Pentium cu 32 MB de RAM. De menționat este faptul că în cazul aplicațiilor complexe este recomandat să se folosească sisteme cu specificații mai avansate de atât.

Dimensiune mare disponibilă pentru tabel: Tabelele MySQL pot crește considerabil, iar unele arhitecturi pot găzdui până la 8 terabytes per tabel folosind MySQL.

Stabilitate: MySQL este în general considerat un sistem stabil, pe care programatorii se pot baza în construirea aplicațiilor software, dar este de menționat că unele caracteristici proaspăt introduse pot conduce la situații neașteptate. Din acest motiv este necesară alegerea atentă a versiunilor folosite, de preferat unele testate riguros de către echipa de dezvoltare a MySQL. (Suehring, 2002, p.18)

2.1.4 Docker

“Docker este o platformă open-source care ajută la dezvoltarea, livrarea și rularea aplicațiilor software. Prin folosirea Docker, se pot separa aplicațiile de infrastructură, ceea ce permite livrarea rapidă a software-ului.” (Documentația oficială Docker)

Docker permite împachetarea și rularea de aplicații în containere izolate. Aceste containere pot fi executate pe aceeași mașină gazdă, beneficiind de izolare și securitate. Un avantaj major este că aceste containere sunt ușoare și includ tot ce este necesar pentru a rula aplicația, fără a depinde de configurația gazdei. De asemenea, containerele pot fi partajate între dezvoltatori, fapt ce garantează că toți programatorii ce lucrează la aceeași aplicație pot beneficia de configurații de sistem identice. În plus, structura bazată pe containere pe care docker o adoptă este ideală pentru realizarea unui flux de lucru cu integrare și livrare continuă (CI/CD). (Documentația oficială Docker)

2.2 Tehnologii de front-end

2.2.1 HTML

HyperText Markup Language, cunoscut mai frecvent drept HTML, este limbajul de marcare folosit pentru definirea structurii a paginilor web. Prin intermediul etichetelor, HTML specifică cum trebuie să fie organizate textul, imaginile, link-urile și alte elemente într-un document web. De exemplu, pentru definirea unui paragraf se poate folosi eticheta <p>, iar pentru inserarea unei imagini eticheta . HTML face posibilă crearea de site-uri web complexe și interconectate prin utilizarea etichetelor de link, cum ar fi <a>. Astfel, navigarea între pagini și crearea de conținut online devin posibile. (Stefanescu, 2023)

2.2.2 CSS

CSS, sau Cascading Style Sheets, este un limbaj de stilizare care definește aspectul documentelor HTML. Acesta a fost publicat de către World Wide Web Consortium (W3C) la mijlocul anilor 1990 pentru a oferi un standard de stilizare al fișierelor HTML. CSS oferă posibilitatea programatorilor de a controla aspectul unei pagini web, inclusiv culorile, dimensiunile, aliniamentul și alte astfel de elemente. De exemplu, schimbarea familiei de fonturi (font-family) în toate paginile HTML ale proiectului se poate face rapid și ușor folosind CSS, în locul modificării fiecărei pagini în parte. CSS poate fi, de asemenea, utilizat pentru a crea tranziții și animații ale elementelor pe o pagină web. (Stefanescu, 2023)

2.2.3 JavaScript

JavaScript este un limbaj de programare folosit în paginile web pentru extinderea posibilităților de dinamism și interactivitate. Limbajul acesta este unul de scripting, semnificând faptul că secvențele de cod JavaScript sunt definite într-un fișier independent, ce este mai apoi inclus într-un document HTML pentru a fi interpretat de către browser, fără nevoia de a fi compilat în cod mașină. Această abordare are drept avantaje viteza de scriere și testare a codului, însă aduce de la sine eficiență scăzută la rulare, comparativ cu limbajele compilate. (Stefanescu, 2023)

2.2.4 Bootstrap

Bootstrap, inițial cunoscut sub numele de Twitter Blueprint, reprezintă un framework open-source pentru CSS utilizat pentru dezvoltarea web pe front-end. A fost creat de Mark Otto și Jacob Thornton, angajați la Twitter. Înainte de Bootstrap, utilizarea diverselor biblioteci pentru dezvoltarea interfețelor ducea la inconsistente și o sarcină mare de întreținere. Bootstrap a fost conceput pentru a standardiza instrumentele front-end utilizate în cadrul companiei la care cei doi dezvoltatori lucrau. (Spurlock, 2013, p.1)

Acest framework oferă dezvoltatorilor componente de interfață cu utilizatorul (UI) bazate pe HTML, CSS și JavaScript din categorii precum butoane, bari de navigare, formulare, dar și casete de dialog, precum modalele. Toate aceste componente reutilizabile pot fi personalizate în continuare de către programatori pentru îndeplinirea cerințelor aplicației pe care o implementează. Interfețele create cu ajutorul Bootstrap pot fi construite cu ușurință pentru a se adapta dispozitivului pe care este vizualizat website-ul și pentru a oferi o experiență plăcută utilizatorilor. (Documentația Bootstrap W3Schools)

Multiple versiuni ale framework-ului Bootstrap au fost publicate de la lansarea sa inițială din august 2011, când acesta era concentrat pe utilizarea de CSS. De atunci s-a ales integrarea anumitor plugin-uri de JavaScript pentru o interactivitate mai plăcută cu aplicația a utilizatorului final, ajungându-se la versiunea de Bootstrap 5 în prezent. (Spurlock, 2013, p.1)

Pentru integrarea framework-ului Bootstrap într-o aplicație web se pot alege mai multe abordări, însă cea mai ușoară este includerea fișierelor CSS și JavaScript în paginile HTML ale aplicației prin folosirea tag-urilor `<link>`, respectiv `<script>`. Este de menționat că pentru a efectua integrarea în acest mod este necesară conexiunea la Internet. (Spurlock, 2013, p.1)

Printre avantajele prezentate de către acest framework de interfață cu utilizatorul, cele mai importante ar fi ușurința de integrare și utilizare a acestuia în aplicații web, chiar în unele deja existente, adaptabilitatea componentelor definite în acest framework în funcție de proporțiile ecranului dispozitivului ce accesează aplicația, dar și faptul că cea mai recentă versiune a acestuia este compatibilă cu majoritatea browser-elor moderne și populare, precum Google Chrome, Microsoft Edge, Mozilla Firefox, Safari, Opera, iar pentru browser-e mai vechi se pot utiliza versiuni mai vechi precum Bootstrap 3 sau 4. (Documentația Bootstrap W3Schools)

2.3 Algoritmul “Breadth First Search”

```
Procedure BFS( $G, s; \lambda$ )  
  
1  empty  $Q$  and put  $s$  in  $Q$   
2   $\lambda(s) \leftarrow 0$   
3   $T \leftarrow \{s\}$   
4  while  $Q \neq \emptyset$  do  
5      remove the first vertex,  $u$ , from  $Q$   
6      for every edge  $u - v$ , if  $v \notin T$ , do  
7           $T \leftarrow T \cup \{v\}$   
8           $\lambda(v) \leftarrow \lambda(u) + 1$   
9          put  $v$  in  $Q$ 
```

Figura 2.1 Pseudocod algoritm BFS, preluat din (Even, 2012, p. 12)

- **Q:** Coada Q este utilizată pentru a ține evidența nodurilor care trebuie să fie explorate în continuare. Aceasta funcționează pe principiul “primul intrat, primul ieșit” (FIFO), unde noile noduri adânci în arborele de căutare sunt adăugate la sfârșitul cozii. Vechile noduri, care sunt mai aproape de rădăcină, sunt expandate mai întâi.
- **s:** Reprezintă nodul de start (sau sursă) de la care începe parcurgerea BFS.
- **λ :** Este o funcție ce reprezintă distanța minimă calculată de către algoritmul BFS de la nodul de start până la toate nodurile grafului. De exemplu, $\lambda(x)$ va fi valoarea distanței de la nodul de start la nodul x .
- **T:** Setul T este inițializat ca fiind gol și este populat pe parcursul execuției BFS. Acesta conține nodurile care au fost deja vizitate.
- **u și v:** Acestea sunt variabile care reprezintă nodurile în graf. u este nodul extras din coadă, iar v este unul dintre vecinii lui u .

Inițializare:

Algoritmul BFS primește ca parametri graful(G), nodul de start (s) și lista pentru stocarea distanțelor de la nodul sursă la celelalte noduri.

- Coada Q este inițializată ca fiind goală: $Q \leftarrow \emptyset$.
- Nodul de start (sau sursă) este pus în coadă: $Q(s)$.

- Se marchează nodul de start ca vizitat (aceasta reprezentând distanța de la nodul s la el însuși): $\lambda(s) \leftarrow 0$.

Explorare:

- Cat timp coada Q nu este goală:
 - Se extrage un nod din coadă: $u \leftarrow Q()$.
 - Se adaugă nodul în lista de noduri vizitate: $T \leftarrow T \cup \{u\}$.
 - Pentru fiecare vecin v al lui u care nu a fost încă vizitat:
 - Dacă v nu este în setul T , continuăm:
 - Adăugăm v în coadă: $Q(v)$.
 - Actualizăm distanța pentru v : $\lambda(v) \leftarrow \lambda(u) + 1$.

Terminare:

- Procesul continuă până când coada Q devine goală.

Algoritmul BFS explorează graful pe niveluri, începând de la nodul de start. Coada Q asigură că nodurile sunt vizitate în ordinea în care au fost adăugate. Setul T conține nodurile deja vizitate, iar funcția λ urmărește distanțele sau nivelurile în arborele de căutare.

Beneficii și Eficiență:

- **Găsirea Drumului Minim:** Deoarece BFS explorează nodurile nivel după nivel, garantează găsirea celui mai scurt drum într-un graf neponderat.
- **Complexitate:** Timpul de rulare al algoritmului este $O(V + E)$, unde V este numărul de noduri și E este numărul de muchii, făcându-l eficient pentru grafuri mari.

2.4 Testul de orientare

Testul de orientare reprezintă un instrument matematic esențial, utilizat pe scară largă pentru a determina poziția relativă a unui punct față de o dreaptă orientată. Acest test se bazează pe conceptul de determinant al unei matrice și are aplicații diverse în domenii precum geometria computațională și grafica pe calculator.

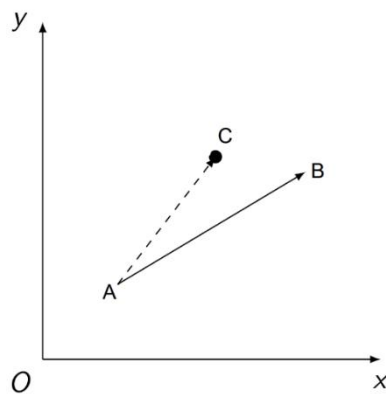


Figura 2.2 Poziția relativă a unui punct față de un vector, adaptată după (Sacristan, n.d., p.4)

Pentru a efectua acest test de orientare, luăm în considerare trei puncte distincte din plan, pe care le notăm cu A, B și C. Fiecare dintre aceste puncte are coordonate specifice: $A(x_A, y_A)$, $B(x_B, y_B)$ și $C(x_C, y_C)$. Utilizând coordonatele acestor puncte, putem construi o matrice 3x3. Fiecare rând al acestei matrice corespunde unui punct, iar prima coloană este completată cu 1, pentru a facilita calculul determinantului. Matricea astfel obținută este (Sacristan, n.d., p.5-6):

$$\begin{vmatrix} 1 & 1 & 1 \\ x_A & x_B & x_C \\ y_A & y_B & y_C \end{vmatrix}$$

Determinantul acestei matrice este o valoare scalară care ne oferă informații importante despre poziția punctului C în raport cu dreapta formată de punctele A și B. Calculul determinantului se face folosind formula standard pentru determinantul unei matrice 3x3, care implică o serie de produse și sume ale elementelor matricei.

Semnificația valorii determinantului poate fi interpretată în moduri diferite, în funcție de semnul său (Sacristan, n.d., p.5-6):

- Dacă determinantul este pozitiv, aceasta indică faptul că punctul C se află la stânga dreptei AB, atunci când sensul de parcurgere este de la A la B.
- Dacă determinantul este negativ, punctul C se situează la dreapta segmentului orientat AB.
- Dacă determinantul este zero, concluzia este că punctele A, B și C sunt coliniare.

Testul de orientare ne permite să înțelegem relațiile geometrice dintre puncte într-un plan și să rezolvăm probleme complexe de geometrie printr-o metodă sistematică și matematic riguroasă. Aceasta este o unealtă fundamentală pentru oricine lucrează în domenii care implică

geometria și grafica pe calculator, oferind o bază solidă pentru analiza și soluționarea problemelor geometrice.

2.5 Descrierea arhitecturii client-server

În cele ce urmează voi descrie elementele componente atât ale părții de server (backend), cât și ale celei de client (frontend).

2.5.1 Backend-ul (Serverul)

Backend-ul aplicației web este responsabil pentru gestionarea logică a datelor și a funcționalităților. Python, împreună cu framework-ul Django, oferă un mediu robust și complet pentru dezvoltarea acestui backend, bazat pe următoarele elemente:

- **Modelul de date (Model):** În Django, modelele de date reprezintă structura și logica de stocare a informațiilor în baza de date. Acestea sunt definite în module Python și pot include diferite câmpuri și metode pentru a manipula datele, fiind gestionate de către ORM.
- **Vizualizarile (Views):** Vizualizarile în Django sunt funcții Python care preiau cererile HTTP, procesează datele și returnează răspunsuri corespunzătoare. Acestea utilizează modelele pentru accesarea și manipularea datelor necesare și pot randa, de asemenea, șabloane HTML pentru a afișa informațiile utilizatorului.
- **Rutele URL (URL Routes):** Rutele URL în Django sunt utilizate pentru a asocia cererile HTTP, în funcție de URL-ul pe care acestea îl prezintă, cu vizualizările corespunzătoare. Acestea sunt definite în fișierele de configurare a rutelor și asigură navigarea corectă în cadrul aplicației.

2.5.2 Frontend-ul (Clientul)

Frontend-ul aplicației web este responsabil pentru prezentarea informațiilor și interacțiunea cu utilizatorul, sarcină ce revine unui browser web. HTML este utilizat pentru structura paginilor web, CSS pentru stilizarea aspectului vizual, iar JavaScript pentru interactivitate și manipularea paginilor.

- **Șabloane HTML (Templates):** În Django, șabloanele HTML sunt utilizate pentru a genera interfața utilizatorului în mod dinamic și personalizat, în funcție de necesitățile aplicației, combinând datele procesate de backend cu structura definită prin HTML. Datele sunt furnizate de către vizualizările ce randează aceste template-uri, iar interpolarea lor în șabloane se realizează folosind o sintaxă specială, numită Jinja.
- **Stilizare (CSS):** Fișierele CSS sunt utilizate pentru a stiliza elementele HTML și pentru a crea aspectul vizual al aplicației. Acestea pot fi utilizate pentru a defini culori, fonturi, dimensiuni și alte proprietăți de design.
- **Interactivitate (JavaScript):** JavaScript este utilizat pentru a adăuga interactivitate paginilor web, cum ar fi validarea formularelor, efecte de animație complexe și actualizarea datelor în timp real. Acesta permite crearea unei experiențe dinamice și interactive pentru utilizatori.

2.5.3 Comunicarea Client-Server

Comunicarea dintre backend și frontend se va realiza în modul următor: aplicația client (browser-ul) va iniția cereri HTTP către server pentru a cere sau trimite informații, iar server-ul, având rutele URL și view-urile configurate, pune la dispoziție unele endpoint-uri ce vor primi aceste cereri, le vor procesa, folosind eventual datele din modelele de date, pentru ca în final să întoarcă un răspuns aplicației client, ce poate include și un template randat.

Prin integrarea acestor componente esențiale, o aplicație web bazată pe arhitectura client-server în Python, cu Django și alte tehnologii precum HTML, CSS și JavaScript, poate oferi o experiență web puternică și interactivă pentru utilizatori. Aceasta permite dezvoltatorilor să creeze aplicații web complexe și scalabile, care pot îndeplini o varietate de cerințe și nevoi ale utilizatorilor.

3. Partea practică

3.1 Arhitectura aplicației

Pentru a obține o organizare a codului cât mai eficientă și pentru a facilita întreținerea și dezvoltarea ulterioară a aplicației mele, am ales să adopt principiul separării conform structurii Model-View-Template (MVT). Această abordare asigură o delimitare clară între diferitele componente ale aplicației, simplificând astfel gestionarea și dezvoltarea codului.

În procesul de dezvoltare a scheletului aplicației, m-am inspirat din videoclipul de pe canalul de Youtube Telusko.

Framework-ul Django, ales pentru dezvoltarea back-end-ului aplicației, se potrivește perfect cu această abordare, oferind suport robust pentru fiecare componentă MVT:

3.1.1 Model

Django oferă un ORM (Object-Relational Mapping) puternic care simplifică definirea și gestionarea modelelor. ORM-ul Django permite interacțiunea cu baza de date într-un mod intuitiv, eliminând necesitatea scrierii de cod SQL explicit și simplificând foarte mult operațiile de creare, citire, actualizare și ștergere (CRUD) a datelor.

Astfel, am definit modelele pentru a reprezenta structura datelor și relațiile dintre ele. Pentru aplicația mea de gestionare a cursurilor din Facultatea de Matematică și Informatică, am creat modele pentru utilizatori (User), materii (Subject) activități ale profesorilor (Teacher_Activity), materiale (Materials) și activități ale studenților (Student_Activity). Pe lângă aceste modele am definit și o enumerare ce conține valori pentru fiecare tip de activitate didactică, denumită Activity_Type.

Fiecare activitate desfășurată de un profesor, definită în modelul Teacher_Activity, este direct asociată cu un profesor specific prin intermediu teacher_id, care face referință la modelul User, stabilind astfel legătura între activitatea didactică și utilizatorul care o conduce. De asemenea, aceste activități sunt legate de materiile corespunzătoare prin course_id, creând o asociere directă cu modelul Subject.

Materialele educaționale, gestionate prin modelul Material, sunt conectate la activitățile specifice ale profesorilor prin intermediul teacher_activity_id, ceea ce înseamnă că fiecare

resursă este direct asociată cu o activitate definită anterior în Teacher_Activity, facilitând astfel organizarea și accesul la resursele necesare pentru cursuri, seminare și laboratoare. Pe de altă parte, participarea studenților la aceste activități este monitorizată prin modelul Students_Activity, unde student_id face referință la modelul User, legând fiecare înregistrare de un student specific, în timp ce teacher_activity_id asigură corelarea cu activitatea didactică relevantă.

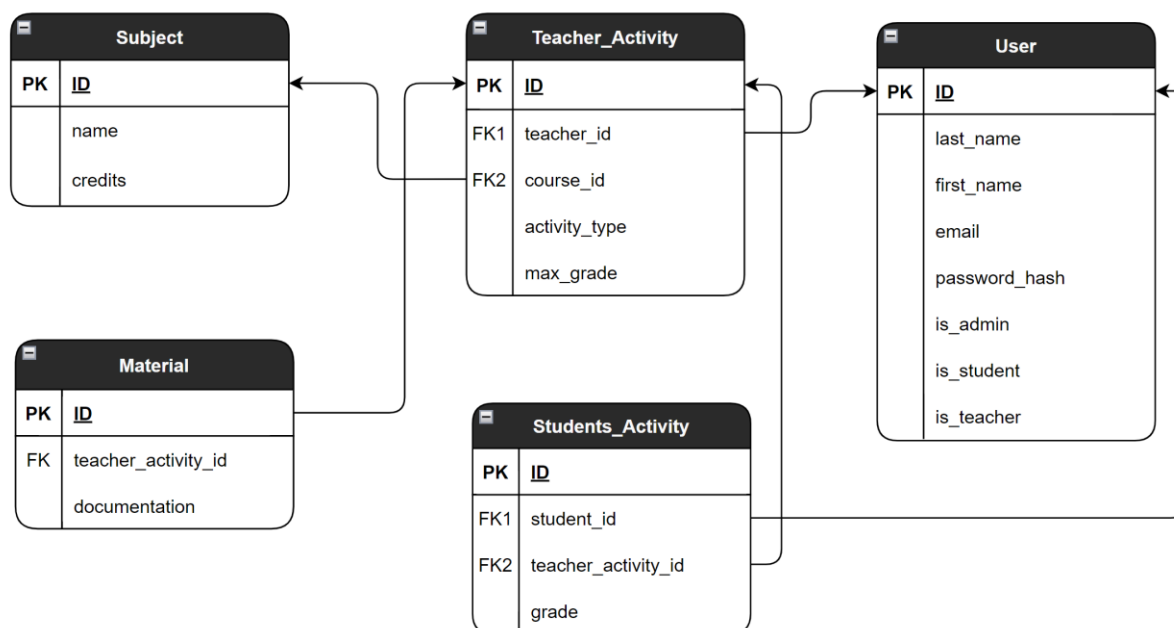


Figura 3.1 Diagrama bazei de date

Modelul User

Modelul User extinde modelul AbstractUser, deja implementat în Django, pentru a adăuga câmpuri suplimentare specifice aplicației mele cum ar fi **is_student** și **is_teacher**. Aceasta permite diferențierea utilizatorilor pe baza rolurilor lor în sistem:

Atribute:

- **email**: Un câmp de tip EmailField care stochează adresa de email a utilizatorului limitat la 50 de caractere pentru a menține consistența și a preveni input-uri invalide. EmailField este un câmp specific în Django utilizat pentru a stoca și valida adrese de email în modelele bazei de date. Acest câmp oferă validări implicite pentru a asigura că datele introduse respectă formatul unei adrese de email valide. De exemplu, se verifică dacă există un simbol '@' și un domeniu valid după acesta.
- **first_name**: Un câmp de tip CharField destinat stocării prenumelui utilizatorului.

- **last_name:** Similar cu first_name, acest câmp stochează numele de familie al utilizatorului. Acest câmp împreună cu first_name sunt esențiale pentru identificarea clară a utilizatorului curent.
- **is_student:** Un câmp de tip BooleanField care indică dacă utilizatorul are rolul de student. Setat la False implicit, filtrează accesul utilizatorilor la funcționalitățile destinate studenților.
- **is_teacher:** Un alt câmp de tip BooleanField care indică dacă utilizatorul are rolul de profesor. De asemenea, implicit False, este folosit pentru gestionarea diferitelor permisiuni și funcționalități specifice profesorilor.

Activity_Type

Activity_Type este o clasă de alegeri (TextChoices) care definește tipurile de activități academice:

Valori:

- **LECTURE:** Reprezintă o activitate de tip prelegere, utilizată pentru cursurile teoretice.
- **SEMINAR:** Reprezintă un seminar, folosit pentru activități interactive și discuții.
- **LABORATORY:** Reprezintă un laborator, destinat activităților practice și experimentale.

Modelul Subjects

Modelul Subject reprezintă cursurile disponibile în sistem.

Atribute:

- **id:** Un câmp de tip AutoField care generează automat un identificator unic pentru fiecare curs. Asigură unicitatea fiecărui subiect în baza de date.
- **name:** Un câmp de tip CharField cu o lungime maximă de 50 de caractere, care stochează numele cursului. Numele materiilor sunt unice pentru că nu ar avea sens să existe două materii cu același nume.

- **credits:** Un câmp de tip IntegerField care stochează numărul de credite alocat cursului. Acesta este folosit pentru a evalua importanța și volumul de muncă necesar pentru curs.

Modelul Teacher_Activity

Modelul Teacher_Activity leagă profesorii de activitățile specifice pe care le desfășoară.

Atribute:

- **id:** Un câmp de tip AutoField care generează automat un identificator unic pentru fiecare activitate a profesorului.
- **activity_type:** Un câmp de tip CharField care stochează tipul de activitate. Acesta folosește valorile definite în Activity_Type și are o lungime maximă de 10 caractere.
- **max_grade:** Un câmp de tip FloatField care stochează nota maximă posibilă pentru activitate.
- **teacher:** Un câmp de tip ForeignKey care face referință la modelul User. Acesta stochează profesorul asociat activității și are setat on_delete=models.CASCADE, ceea ce înseamnă că activitatea este ștearsă dacă profesorul este ștears.
- **course:** Un alt câmp de tip ForeignKey, care face referință la modelul Subject. Acesta stochează cursul asociat activității și are on_delete=models.CASCADE.

Modelul Materials

Modelul Materials reprezintă materialele didactice asociate activităților profesorilor.

Atribute:

- **id:** Un câmp de tip AutoField care generează automat un identificator unic pentru fiecare material. Acest lucru asigură gestionarea unică și organizată a materialelor.
- **documentation:** Un câmp de tip FileField care permite încărcarea documentelor. Utilizează funcția documentation_upload_path pentru a determina calea la care vor fi încărcate fișierele și le validează pentru a accepta doar extensii pdf, asigurând astfel un standard de formatare.
- **teacher_activity:** Un câmp de tip ForeignKey care face referință la modelul Teacher_Activity. Stochează activitatea profesorului asociată materialului și are

`on_delete=models.CASCADE`. Acest câmp este setat la `null=True` pentru a permite inițializarea fără date complete.

Modelul `Students_Activity`

Modelul `Students_Activity` reprezintă legătura dintre studenți și activitățile desfășurate de profesori.

Atribute:

- **id**: Un câmp de tip `AutoField` care generează automat un identificator unic pentru fiecare activitate a studentului. Acesta asigură că fiecare înregistrare este unică și poate fi identificată separat.
- **student**: Un câmp de tip `ForeignKey` care face referință la modelul `User`. Acesta stochează studentul asociat activității și are `on_delete=models.CASCADE`.
- **teacher_activity**: Un câmp de tip `ForeignKey` care face referință la modelul `Teacher_Activity`. Acesta stochează activitatea profesorului asociată studentului și are `on_delete=models.CASCADE`.
- **grade**: Un câmp de tip `FloatField` care stochează nota obținută de student în cadrul acelei activități, permițând evaluarea și urmărirea performanței academice.

3.1.2 View

În cadrul Django, view-urile sunt responsabile pentru logica de procesare a cererilor. View-urile preiau cererile HTTP, interacționează cu modelele pentru a accesa sau modifica datele necesare și returnează răspunsuri adecvate. Atunci când server-ul primește o cerere de la un client, acesta verifică URL-ul cererii respective, iar dacă se găsește o potrivire în lista sistemului de URL routing este apelat view-ul asociat șablonului URL respectiv. De exemplu, atunci când un utilizator introduce o adresă URL în bara browser-ului sau interacționează cu aplicația, o cerere este trimisă către server. Acolo URL-ul este verificat după cum am menționat anterior, iar mai apoi este apelat view-ul asociat, în cazul unei potriviri. View-ul aplică logica definită în corpul său pentru cererea primită și întoarce un răspuns utilizatorului, fie sub forma unui fișier HTML, a unui obiect JSON sau de informații sub orice alt format. În aplicația mea, am creat view-uri pentru a gestiona diferitele cereri ale utilizatorilor, precum autentificarea,

înregistrarea, afișarea materialelor educaționale, încărcarea și ștergerea de materiale, gestionarea notelor și oferirea de indicații de orientare în facultate.

În Django, funcția `path()` este utilizată pentru a asocia un URL unui view specific în cadrul unei aplicații web. Aceasta face parte din modulul `django.urls`.

Parametrii pe care funcția `path()` îi primește sunt următorii (Documentația oficială Django):

- **URL:** Primul parametru al funcției `path()` este șablonul URL-ului pe care dorim să-l asociem unui view. Acesta poate fi un șir de caractere simplu sau unul formatat într-un mod specific pentru a conține variabile ce pot fi accesate mai departe în view.
- **View:** Al doilea parametru specifică funcția view care va fi apelată atunci când URL-ul specificat este accesat.
- **Name (Nume)** (opțional): Parametrul `name` este un identificator unic pentru URL-ul respectiv. Este util pentru a face referire la URL-ul primit drept argument din alte părți ale codului, precum utilizarea în șabloanele HTML fără a mai fi nevoie să fie scris întreg URL-ul.

View-ul home

Funcția `home` din fișierul `views.py` primește un argument `request`, care este un obiect ce reprezintă cererea HTTP trimisă de client, conținând toate informațiile despre această cerere, inclusiv metoda (GET, POST), datele trimise și informațiile despre utilizatorul autentificat. Funcția utilizează `render` pentru a genera un răspuns HTTP cu conținut HTML, combinând șablonul `home.html` cu un dicționar gol de variabile de context. Astfel, atunci când un utilizator accesează pagina principală, serverul Django mapează URL-ul către funcția `home`, care generează și trimite înapoi răspunsul HTML corespunzător.

View-ul Login

Prin intermediul funcției `user_login` se realizează verificările și acțiunile necesare autentificării utilizatorilor. Aceasta primește un obiect numit `request`, care conține detalii despre cererea HTTP. Dacă utilizatorul este deja autentificat (`request.user.is_authenticated`), acesta este redirecționat către pagina principală (`redirect('/')`). Dacă cererea este de tip POST,

funcția extrage numele de utilizator și parola din formularul trimis și încearcă să autentifice utilizatorul folosind `'auth.authenticate'`. Dacă autentificarea reușește, utilizatorul este conectat cu `'auth.login'` și redirecționat către pagina principală. Dacă autentificarea eșuează, se afișează un mesaj de eroare și utilizatorul este redirecționat înapoi la pagina de login (`redirect('login')`). Pentru cererile de tip GET, se redă pagina de login (`login.html`).

View-ul Logout

View-ul `user_logout` este folosit pentru gestionarea cererii de deconectare a utilizatorului. Se verifică mai întâi dacă utilizatorul este autentificat prin accesarea atributului `is_authenticated` asociat obiectului `user` prezent în obiectul `request`. În caz afirmativ se apelează funcția `auth.logout(request)` pentru ștergerea datelor legate de utilizator din sesiunea gestionată de Django și cookie-urile aferente, urmând ca utilizatorul respectiv să fie redirecționat către pagina principală a aplicației. În cazul în care utilizatorul ce apelează view-ul acesta nu este autentificat, el este redirecționat către pagina login.

View-ul Register

Acest view returnează răspunsul HTTP ce redă pagina `register.html`, împreună cu formularul de înregistrare (`{'form': form}`). Utilizatorul va completa și trimite datele necesare pentru crearea unui cont nou.

Funcția începe prin verificarea dacă utilizatorul este deja autentificat (`request.user.is_authenticated`), în acest caz utilizatorul fiind redirecționat către pagina principală (`/'`), deoarece un utilizator autentificat nu ar trebui să acceseze pagina de înregistrare.

Dacă cererea HTTP este de tip POST, funcția continuă cu următorii pași:

- Creează o instanță a formularului `RegistrationForm` folosind datele trimise în cerere (`request.POST`).
- Verifică dacă formularul este valid (`form.is_valid()`). Acest lucru implică validarea datelor introduse de utilizator (dacă adresa de e-mail este validă și parolele coincid).
- Dacă formularul este valid, datele sunt salvate în baza de date folosind `form.save()`. Astfel, se creează un nou utilizator în sistem.
- Utilizatorul este redirecționat către pagina de login (`'login'`) pentru a se autentifica cu noile date de conectare.

Dacă cererea HTTP nu este de tip POST (de exemplu, este de tip GET), funcția creează o instanță goală a formularului `RegistrationForm`. Acest formular va fi afișat utilizatorului pentru a completa datele de înregistrare.

View-ul activity

Funcția `activity` gestionează afișarea activităților academice atât pentru studenți, cât și pentru profesori, pe baza rolului utilizatorului autentificat. Este decorată cu `@login_required`, ceea ce înseamnă că doar utilizatorii autentificați pot accesa această funcție.

Dacă metoda cererii nu este GET, utilizatorul este redirecționat către pagina principală ('/').

Pentru studenți, funcția colectează toate activitățile asociate cu studentul curent prin modelul `Students_Activity` și creează un dicționar `info_dict` structurat pe cursuri și tipuri de activități. Pentru fiecare activitate, funcția accesează materialele asociate din modelul `Materials` și le adaugă în dicționar cu informații despre URL-ul documentului și numele acestuia. După același principiu se construiește și dicționarul `teachers_names`, destinat reținerii numelor profesorilor ce predau fiecare activitate didactică. În final, pagina `activity.html` este redată cu aceste două dicționare de informații.

În cazul în care utilizatorul autentificat este un profesor, funcția colectează toate activitățile didactice ale acestuia prin modelul `Teacher_Activity`. Asemănător procesului descris în paragraful anterior, funcția construiește un dicționar `info_dict` organizat pe cursuri și tipuri de activități, dar adaugă și un alt dicționar, `teach_act_dict`, care mapează cursurile și tipurile de activități asociate profesorului. Pentru fiecare activitate didactică, funcția obține materialele asociate și le adaugă în dicționar cu informații despre URL, numele documentului și ID-ul materialului, util pentru operațiunile disponibile în template ce privesc aceste materiale. La final, pagina `activity.html` este redată cu ambele dicționare, `info_dict` și `teach_act_dict`, pentru a furniza informații complete despre activitățile și materialele profesorului.

View-ul upload material

Funcția `upload_material` este responsabilă pentru gestionarea procesului de încărcare a materialelor didactice de către profesori.

Inițial se verifică dacă cererea HTTP este de tip POST și dacă utilizatorul curent este un profesor, pentru a permite doar acestuia să încarce materiale. Astfel, se obțin datele despre profesor din cerere trimisă din aplicație (`teacher = request.user`), cât și cursul și activitatea specifică a profesorului. În continuare se verifică dacă nu există un fișier în cererea `POST('documentation' not in request.FILES)` sau dacă fișierul nu este în format PDF (`documentation.name.split('.')[-1] != 'pdf'`), astfel afișând un mesaj de eroare și redirecționând utilizatorul înapoi la pagina de activități. În cazul în care validarea acestor condiții este îndeplinită, se creează un nou obiect de tip `Materials` asociat activității didactice a profesorului în care este încărcată calea fișierului primit în cererea HTTP, fișierul fizic fiind stocat în sistemul de fișiere al computer-ului pe care rulează server-ul, iar în final obiectul este salvat în baza de date. Dacă verificarea inițială nu reușește, metoda cererii fiind alta decât POST sau utilizatorul neavând rol de profesor, acesta este redirecționat către pagina principală a aplicației.

View-ul delete material

Funcția `delete_material` este folosită pentru ștergerea unui material didactic specificat prin `material_id` atât din baza de date, cât și din sistemul de fișiere al calculatorului gazdă.

În primă instanță, aceasta încearcă să găsească materialul specificat prin `material_id` utilizând `Materials.objects.get(id=material_id)`, iar dacă materialul nu este găsit, se tratează excepția `Materials.DoesNotExist` prin afișarea unui mesaj de eroare utilizatorului cu textul “Material not found!”.

Dacă utilizatorul curent este un profesor, se verifică existența fișierului asociat materialului în sistemul de stocare, iar dacă fișierul există, acesta este șters din sistemul de stocare folosind `default_storage.delete(material.documentation.name)`. După ștergerea fișierului din stocare, materialul este eliminat din baza de date cu `material.delete()`, iar utilizatorului `i` se afișează un mesaj de succes cu textul “Material deleted successfully!”. În final, utilizatorul este redirecționat către pagina de activități (`return redirect('/activity')`),

În caz contrar, dacă un utilizator fără rol de profesor încearcă să acceseze funcția, acesta este redirecționat automat la pagina principală fără a efectua nicio acțiune asupra materialului.

View-ul directions

Funcția `directions` gestionează cererile HTTP pentru obținerea indicațiilor de orientare între camere din clădirea facultății, folosind un graf pentru a modela relațiile spațiale. Lista de mai jos conține o descriere mai detaliată a operațiilor efectuate de această funcție:

- Funcția începe prin inițializarea unei instanțe a grafului, variabilelor pentru indicații și mesaje de eroare, și determinarea camerelor de start și de sfârșit, în cazul unei cereri POST.
- Dacă cererea este de tip POST, sunt extrase valorile pentru camerele de start și sfârșit din corpul cererii. Dacă camerele nu sunt găsite în graf, funcția setează un mesaj de eroare corespunzător “Invalid or no room selected! Please, select a valid one!”, altfel se calculează indicațiile de orientare necesare dintre aceste camere cu ajutorul metodei `get_directions` a grafului.
- Este definită funcția `process_node`, ce preia un nod din graf și verifică dacă acesta este valid și dacă conține detalii relevante. Dacă nodul nu există în graf sau nu are detalii, funcția returnează `None`. Dacă nodul reprezintă o intrarea în facultate (marcat prin valoarea atributului `no_room` a nodului care este “0”) sau are detalii specifice precum “Terrace” sau “Toilet”, funcția returnează aceste detalii. În caz contrar, dacă nodul are un număr de cameră valid și nu conține detalii specifice care să excludă tipul de cameră (terasă sau toaletă), funcția returnează un string formatat cu numărul și detaliile camerei.
- Informațiile despre toate camerele sunt obținute din graf, sortate în funcție de numărul camerei. În șablonul `directions.html` se transmit variabilele `rooms` (camerele sortate), `directions` (indicațiile de orientare obținute), `error` (mesajul de eroare, dacă există), și rezultatele procesării nodurilor de start și finish prin funcția `process_node`.

View-ul Grades

Funcția `grades` este responsabilă pentru gestionarea cererilor legate de afișarea și manipularea notelor studenților, în funcție de rolul utilizatorului autentificat. Iată o descriere detaliată a funcției:

- Funcția începe prin verificarea rolului utilizatorului autentificat și a metodei cererii HTTP. Dacă cererea nu este de tip GET, utilizatorul este redirecționat către pagina principală pentru a preveni utilizarea incorectă a funcționalității.
- Dacă utilizatorul autentificat este un student, funcția extrage activitățile la care este înscris și notele acestuia din baza de date și construiește un dicționar `info_dict` care conține informații despre notele fiecărui student pentru fiecare activitate, împreună cu informații suplimentare cum ar fi creditele asociate fiecărei materii și notele maxime posibile pentru fiecare activitate didactică la care este înscris. De asemenea, calculează nota finală pentru fiecare curs și o normalizează la o scară de la 0 la 10.
- În cazul în care utilizatorul autentificat are rol de profesor, funcția extrage activitățile didactice pe care acesta le susține și notele studenților asociate fiecărei astfel de activități din baza de date. Informațiile sunt organizate într-un dicționar `info_dict`, care conține detalii despre studenți și notele acestora structurate în funcție de materie și de activitatea didactică la care participă. De asemenea, sunt definite alte două dicționare, denumite `max_grades` și `teach_act_ids`, ce au o structură asemănătoare cu cea a lui `info_dict` și sunt destinate stocării informațiilor legate de notele maxime posibile și id-urile asociate fiecărei activități susținute de profesorul apelant.

View-ul grade student

Funcția `grade_student` permite unui profesor să acorde sau să actualizeze o notă pentru un anumit student înscris la una dintre activitățile susținute de către el.

- Începe prin a verifica dacă cererea HTTP este de tip POST și dacă utilizatorul autentificat este un profesor, în caz contrar redirecționând utilizatorul la pagina cu notele, fără a face modificări asupra datelor din sistem. În cazul în care condițiile sunt îndeplinite, funcția încearcă să obțină activitatea studentului specificată prin `student_activity_id` din baza de date și preia atât detaliile studentului, cât și ale activității profesorului asociate acesteia.
- Urmează ca funcția să verifice dacă cererea venită de la utilizator conține o valoare în corpul său. Dacă valoarea nu este găsită, mesajul de eroare 'Please, enter a grade!' este setat și funcția se încheie prin redirecționarea către pagina cu note. În caz contrar, se încearcă parsarea unei valori numerice din string-ul extras, iar dacă

aceasta eșuează este setat un mesaj de eroare diferit, și anume 'Invalid grade! Please, enter a number!', utilizatorul fiind din nou redirecționat către pagina cu notele.

- După conversia reușită, funcția verifică dacă nota se încadrează în intervalul permis (între 0 și nota maximă specificată pentru activitatea respectivă). Dacă nota nu se încadrează în acest interval, setează mesajul de eroare adecvat și redirecționează utilizatorul, și de această dată, tot către pagina '/grades'.
- Dacă toate verificările sunt trecute cu succes, setează noua notă și salvează modificările în baza de date. În final, setează un mesaj de succes pentru a informa profesorul că nota a fost actualizată cu succes și redirecționează utilizatorul înapoi la pagina cu notele.

Prin aceste operațiuni, funcția asigură o actualizare corectă și sigură a notelor studenților, validând atent inputul profesorului și asigurând o feedback clar prin mesaje informative.

View-ul modify max grade

Funcția `modify_max_grade` permite profesorilor să modifice nota maximă posibilă asociată unei activități didactice pe care ei o susțin.

- Funcția mai întâi verifică dacă cererea primită este de tip POST și dacă utilizatorul are rol de profesor, iar dacă cel puțin una din condiții nu este adevărată, se setează un mesaj de eroare și utilizatorul este redirecționat către pagina cu notele.
- Funcția extrage noua valoare a notei maxime din corpul cererii și încearcă să o convertească într-un număr cu virgulă mobilă. În cazul în care câmpul este gol sau conversia eșuează, setează câte un mesaj de eroare după caz și redirecționează utilizatorul la pagina cu notele.
- După conversie, funcția verifică dacă nota este un număr pozitiv. Dacă nu este, afișează un mesaj de eroare și redirecționează utilizatorul; altfel, continuă cu actualizarea notei maxime.

- Funcția obține activitatea profesorului specificată prin `teacher_activity_id` din baza de date, setează noua valoare a notei maxime și salvează modificările. În final, afișează un mesaj de succes și redirecționează utilizatorul la pagina cu notele.

3.1.3 Template

Modul de structurare al aplicației web de tip MVT are drept componentă finală pe cea de Template. Componenta aceasta este cea cu care utilizatorul final al aplicației ajunge să interacționeze în mod direct, jucând astfel un rol elementar din punctul său de vedere. De aici se trimit cereri către server, unde sunt apelate view-urile asociate lor, dar tototdată această componentă este responsabilă pentru afișarea informațiilor din model într-o manieră prietenoasă pentru utilizatori.

În cadrul proiectului meu, am utilizat un set divers de template-uri HTML, ce reprezintă componenta responsabilă de definirea aspectului și a interfeței vizuale a aplicației web. În ceea ce privește structura lor, template-urile sunt compuse dintr-o varietate de elemente HTML standard, precum tag-uri `<div>`, ``, `<p>`, etc. Pe lângă acestea, am inclus și limbajul de templating Jinja care permite integrarea dinamică a datelor, precum formulare dinamice, componente interactive și zone de afișare a datelor dinamice, dar și a funcționalităților specifice în cadrul paginilor web.

Am ales să integrez framework-ul Bootstrap în proiectul meu pentru designul template-urilor, ceea ce mi-a permis să folosesc componente predefinite și stiluri adaptive. Cu ajutorul său, am implementat butoane, formulare, bara de navigare și alte elemente UI, simplificând procesul de dezvoltare și reducând timpul necesar pentru crearea unui design coerent și modern.

În spatele fiecărui template se află clase și funcții Python care prelucrează datele, extrag informații din baza de date sau gestionează logica specifică aplicației. Acestea sunt organizate în mod coerent, urmând o structură clară și modulară pentru a asigura un proces de dezvoltare eficient și pentru a scrie cod ușor de întreținut. Astfel, prin intermediul template-urilor HTML și a logicii din spatele lor, se realizează o experiență interactivă și intuitivă pentru utilizatorii finali ai aplicației web.

Pe paginile de înregistrare și autentificare, utilizatorul este întâmpinat de un design intuitiv și prietenos. În ambele pagini, este un formular în care se află câmpurile de completat

informațiile necesare înregistrării, utilizatorul introducând detaliile pentru crearea unui cont nou (Username, Email, First name, Last name, Password și Password confirmation), respectiv autentificării acesta furnizând detaliile unui cont existent pentru a se conecta (Username și Password). După trimiterea formularului, utilizatorul primește confirmarea că datele au fost trimise cu succes și, în caz de erori, îi sunt afișate mesaje clare pentru a le corecta. Pentru crearea acestor template-uri, m-am inspirat din videoclipul de pe Youtube postat pe canalul Ludiflex.

Pagina principală (Home) a aplicației MathInfo Academy include un mesaj de bun venit și descrie cele trei secțiuni importante:

1. Pagina “Directions” pentru orientarea în campus.
2. Pagina “Activity” pentru accesul la materiale educaționale.
3. Pagina “Grades” pentru vizualizarea și gestionarea notelor.

Template-ul de navbar este vizibil în toate celelalte pagini, acesta cuprinzând un meniu de navigare fix în partea de sus, care oferă linkuri către “Home”, “Directions” pentru orice utilizator, iar dacă utilizatorul este autentificat, și către “Activity” și “Grades”. Bara de navigare include și butoane pentru login și register pentru utilizatorii neautentificați sau un mesaj de salut și butonul de logout pentru utilizatorii autentificați. Conținutul specific al celorlalte pagini se încarcă în secțiunea principală definită de `{% block content %}`. În crearea template-ului de navbar, m-am inspirat din videoclipul de pe canalul Code Jungle.

În pagina “Directions” a MathInfo Academy, utilizatorul poate selecta camera de început și destinație folosind câmpuri de căutare interactive. După selectarea camerelor, utilizatorul apasă pe “Get Directions” și obține instrucțiuni pas cu pas afișate într-o secțiune dedicată, cu pictograme pentru orientare vizuală.

Șablonul “Activity” are o structură în format de acordeon (Documentația oficială Bootstrap), unde studenții pot accesa materialele educaționale pentru fiecare materie la care sunt înscriși. Fiecare curs are secțiuni pentru diferite tipuri de activități, precum cursuri și seminare, care pot fi extinse pentru a afișa documentele disponibile. În cazul în care nu sunt disponibile materiale, un mesaj informativ este afișat.

Profesorii au acces la aceeași structură a paginii “Activity”, dar cu opțiuni suplimentare pentru gestionarea materialelor. Pe lângă vizualizarea și descărcarea documentelor de care beneficiază studenții, profesorii pot gestiona materialele prin intermediul unor ferestre modale

(Documentația oficială Bootstrap) asociate fiecărui tip de activitate. Mesajele de feedback, precum erorile sau succesul operațiunilor, sunt afișate în partea de sus a paginii.

În pagina “Grades” studenții pot vedea cursurile la care participă, structurate într-un acordeon, fiecare cu secțiunile destinate activităților didactice la care studentul este înscris. Fiecare curs afișează nota finală și creditele aferente, cât și notele pentru diferitele tipuri de activități dacă se extinde secțiunea activității.

Spre deosebire de studenți, profesorii au câteva funcționalități în plus pentru gestionarea notelor studenților înscriși la activitățile predate de aceștia. În secțiunea fiecărui tip de activitate pot vedea notele fiecărui student și au posibilitatea de a modifica nota maximă pentru activități, respectiv de a edita notele studenților prin intermediul unor ferestre modale dedicate.

3.2 Rolurile utilizatorilor din aplicație

Rolurile multiple în cadrul unei aplicații sunt esențiale pentru gestionarea eficientă a privilegiilor utilizatorilor. Prin definirea de roluri aplicația poate alocă permisiuni specifice fiecărui grup, fără a trebui să seteze manual drepturile pentru fiecare utilizator în parte. Această stratificare a accesului asigură un control precis și securizat, prevenind accesul neautorizat și simplificând administrarea drepturilor utilizatorilor, contribuind la o funcționare fluidă și sigură a aplicației.

În aplicația mea există diverse tipuri de utilizatori, fiecare cu acces specific la anumite funcționalități. Utilizatorii neautentificați pot accesa pagina “Home” și secțiunea de indicații pentru orientare în facultate. Studenții autentificați pot vizualiza materiale din secțiunea “Activity” și notele primite la fiecare disciplină în secțiunea “Grades”. Profesorii autentificați au permisiunea de a edita note pentru studenții alocați activității predate și gestiona materiale aferente fiecărui tip de activitate. Administratorii beneficiază de drepturi speciale, precum gestionarea utilizatorilor și configurarea aplicației. În continuare, voi detalia funcționalitățile accesibile fiecărui tip de utilizator.

3.2.1 Utilizator neautentificat

Utilizatorul neautentificat poate accesa informații despre structura paginilor aplicației pentru a-și face o idee generală asupra acesteia. De asemenea acesta are acces la folosirea

funcționalității de obținerea a indicațiilor de orientare între două săli din facultate. Acest lucru permite oricărei persoane, indiferent dacă deține un cont sau nu, să poată naviga în clădirea facultății, de exemplu dacă este o persoană din afara facultății care este invitată pentru a susține o activitate, aceasta poate accesa aplicația și poate ajunge cu ușurință la sala destinată activității respective.

3.2.2 Utilizator autentificat de tip student

Pentru utilizatorii autentificați de tip student în aplicația web, accesul este extins comparativ cu utilizatorii neautentificați.

Studentii pot accesa, în plus, informații despre materiile la care sunt înscriși împreună cu toate tipurile de activități asociate fiecărei materii, cum ar fi cursuri, laboratoare și seminarii. Pentru fiecare activitate specifică, studenții pot accesa documentația și materialele necesare furnizate de profesori. Totodată aceștia au acces și la notele primite în urma evaluării la fiecare tip de activitate, dar și la calculul notei finale pentru fiecare materie, care include suma tuturor notelor individuale (curs, seminar, laborator), precum și numărul de credite asociat cursului.

3.2.3 Utilizator autentificat de tip profesor

Accesul profesorilor este extins în mod semnificativ față de cel al studenților, permițându-le să interacționeze activ cu materialele specifice activităților didactice și să administreze performanțele academice.

- Profesorii au posibilitatea de a încărca documentația necesară pentru fiecare tip de activitate pe care o predau, fie că este vorba despre cursuri, laboratoare sau seminarii. Aceștia pot adăuga fișiere de tip PDF pentru a sprijini învățarea studenților, dar pot și șterge materialele care nu mai sunt relevante sau necesare.
- Profesorii au acces la lista completă a studenților înscriși la materia pe care o predau. Aceștia pot vedea detalii despre fiecare student și pot monitoriza participarea și progresul acestora în cadrul cursului, acodându-le note în urma evaluării, dar au posibilitatea și de modificare a notelor în caz de erori, sau reevaluări ulterioare.
- De asemenea, profesorii pot ajusta nota maximă care poate fi alocată pentru fiecare tip de activitate, în funcție de complexitatea și importanța acesteia în cadrul materiei.

Aceasta le oferă flexibilitate în stabilirea standardelor de evaluare și în asigurarea unui sistem de notare echitabil.

3.2.4 Administratorul

În aplicația web construită cu Django, utilizatorul de tip administrator are acces complet la gestionarea tuturor aspectelor aplicației. Aceasta se datorează faptului că Django oferă o interfață de administrare integrată, puternică și extensibilă, care permite utilizatorilor privilegiați să administreze eficient conținutul și funcționalitățile aplicației, precum și relațiile dintre diversele sale componente, permițând administratorului să creeze, modifice și elimine elemente după cum este necesar. Această interfață este accesibilă prin ruta “/admin/” și oferă un set robust de instrumente pentru administrarea site-ului.

Administratorul, creat prin comanda “python manage.py createsuperuser”, are un set de permisiuni extinse care îl diferențiază de ceilalți utilizatori autentificați. Cele mai importante permisiuni pe care rolul de administrator le deține sunt:

Capabilitățile adminului din Django includ crearea de noi componente și entități în cadrul aplicației, adăugând funcționalități pentru a satisface cerințele utilizatorilor și nevoile organizaționale. Adminul poate modifica elementele existente pe site, ajustând și actualizând conținutul și configurațiile pentru a îmbunătăți funcționalitatea și eficiența aplicației, precum și ștergerea elementelor neactuale pentru a optimiza performanța.

În cazul concret al aplicației mele, adminul are dreptul de a realiza legături între anumite entități. Astfel acesta poate asocia profesorii cu materiile pe care le predau prin gestionarea activităților profesorului (Teacher Activity) și conectarea studenților cu activitățile la care participă, reflectând participarea lor la cursuri, laboratoare și seminarii prin intermediul activităților studenților (Student Activity).

3.3 Determinarea indicațiilor de orientare

Grafurile sunt reprezentări matematice esențiale pentru modelarea rețelelor de orice tip, inclusiv pentru planurile clădirilor și sunt formate din noduri (reprezentând punctele cheie) conectate prin muchii (ce simbolizează legăturile dintre aceste puncte). Fișierul graph.txt descrie un graf utilizat în aplicație în cadrul secțiunii de orientare în Facultatea de Matematică

și Informatică. Graful este construit astfel încât să reflecte cu precizie dispunerea fizică a sălilor și a holurilor în clădire.

3.3.1 Structura Fișierului

Numărul de Noduri:

- Prima linie a fișierului specifică numărul total de noduri din graf. În acest caz, valoarea “282” indică faptul că există 282 de noduri în graful asociat clădirii facultății.

Detaliile specifice nodurilor:

- Următoarele 282 de linii conțin descrierea fiecărui nod. Fiecare astfel de linie are următoarea structură:
 - **Coordonate:** Primele două numere sunt coordonatele X și Y ale nodului. Acestea sunt calculate astfel încât să corespundă exact poziției fizice a punctelor pe schema clădirii și permit ca indicațiile de orientare generate de aplicație să fie precise și corecte, facilitând navigarea în spațiul fizic.
 - **Eticheta nodului:** Al treilea număr reprezintă ID-ul unic al nodului, utilizat pentru identificarea sa în cadrul grafului.
 - **Tipul de nod:** Acesta este indicat printr-o literă sau grup de litere care specifică natura nodului, cum ar fi:
 - s pentru “nod de sală” (ex. o sală de curs)
 - h pentru “nod de hol” (puncte de trecere în clădire)
 - he pentru “nod de hol-etaj” (puncte de trecere între etaje, de exemplu holul din dreptul scărilor)
 - **Număr de sală (opțional) sau nivelul:** În unele cazuri, nodul include un număr de sală (ex. “321” pentru “Sala 321”). În cazul nodurilor de tip “he”, acest număr reprezintă nivelul (etajul) la care acest nod se află.
 - **Descrierea camerei (Opțional):** Unele noduri pot avea o descriere sub formă de text a locației (ex. “Main Entrance” pentru intrarea principală).

De exemplu, linia “0.5 -0.5 275 he 0 Ground_Floor_Main_Stairs” reprezintă un nod cu coordonatele $X=0.5$, $Y=-0.5$, având ID-ul 275, de tip hol-etaj (he), asociat cu scările principale de la parter.

Descrierea Muchiilor:

- După cele 282 de linii care descriu nodurile, fișierul continuă cu descrierea muchiilor. Fiecare linie în această secțiune a fișierului conține două numere separate prin spațiu ce reprezintă etichetele a două noduri conectate bidirecțional. De exemplu, “275 57” indică o conexiune bidirecțională între nodurile cu ID-urile 275 și 57. Muchiile dintre noduri sunt configurate pentru a reflecta conexiunile reale dintre diferitele puncte din clădire, cum ar fi coridoarele și scările. Aceasta asigură că utilizatorii pot obține instrucțiuni clare și eficiente pentru deplasarea între punctele de interes din facultate.

3.3.2 Descrierea clasei Graph și funcționalitățile sale

Pentru implementarea clasei Graph am ales să definesc alte două clase ajutătoare, în același fișier cu aceasta. Voi descrie mai întâi aceste două clase, Singleton și NodeDetails, după care voi prezenta pe larg clasa Graph.

Descrierea clasei NodeDetails

Clasa NodeDetails este concepută pentru a reprezenta și a stoca detalii despre fiecare nod dintr-un graf, utilizat pentru a modela structura clădirii Facultății de Matematică și Informatică, în cazul meu. Obiecte de tip NodeDetails sunt folosite în clasa Graph pentru a avea o structură mai robustă și lizibilitate a codului ridicată. În fișierul care definește graful, fiecare nod este descris printr-un set de attribute care sunt mapate direct la parametrii constructorului NodeDetails.

Metoda `__init__` din clasa NodeDetails primește șase parametri care definesc proprietățile unui nod: coordonatele x și y , label (eticheta sau id-ul nodului), `node_type` (tipul nodului), `no_room` (numărul camerei), `details` (descrierea camerei) și `floor` (etajul, specific pentru nodurile de tip “he”). Aceste attribute sunt apoi stocate ca variabile de instanță ale obiectului NodeDetails.

1. **Coordonatele x și y:** Acestea sunt stocate ca numere de tip float și reprezintă poziția nodului din planul specific etajului la care se află.
2. **label (eticheta):** Acest atribut de tip int reprezintă un identificator unic pentru nod, utilizat pentru referințierea și accesarea nodului în cadrul grafului.
3. **node_type (tipul nodului):** Acesta este un șir de caractere (str) care specifică tipul nodului, cum ar fi “s” pentru sală, “h” pentru hol, sau “he” pentru hol-etaj.
4. **no_room (numărul camerei):** Disponibil doar în cazul anumitor noduri de tip sală (“s”), acest atribut stochează numărul ei.
5. **details (detalii):** Acest atribut opțional este un șir de caractere care poate conține descrierea camerei sau alte informații suplimentare despre nod. De exemplu, poate specifica dacă nodul este o intrare principală sau o toaletă.
6. **floor (etajul):** Specific pentru nodurile de tip “he”, acest atribut de tip int indică etajul la care se află nodul. Este important pentru navigarea verticală în clădire, cum ar fi urcarea sau coborârea pe scări.

Descrierea clasei Singleton

Clasa Singleton implementează un design pattern cunoscut sub același nume, care asigură că o clasă are o singură instanță în toată aplicația. În metoda `__new__`, verifică dacă `_instance` este `None`. Dacă este, creează o nouă instanță a clasei folosind metoda `super().__new__(cls)`. Dacă `_instance` nu este `None`, returnează instanța existentă, asigurând astfel unicitatea instanței clasei Singleton în aplicație.

Prezentarea clasei Graph

Clasa Graph este o implementare complexă care modelează un graf utilizat pentru a reprezenta structura abstractizată a clădirii Facultății de Matematică și Informatică. Această clasă moștenește din Singleton, ceea ce înseamnă că doar o singură instanță a grafului poate exista în orice moment în aplicație, asigurând astfel un punct central de referință pentru datele structurale.

- **Metoda “ `__init__` ”**

Metoda “ `__init__` ” al clasei Graph este responsabil pentru inițializarea graficului dintr-un fișier specificat sau implicit numit `graph.txt`. Acesta verifică mai întâi dacă atributele

num_nodes, adj, și info sunt deja inițializate pentru instanța curentă. Dacă nu sunt, le inițializează la valori implicite și apoi încarcă graful apelând metoda privată `__load_graph`.

- **Metoda “`__format_string`”**

Metoda “`__format_string`” primește un șir de caractere și înlocuiește toate caracterele de tip underscore (“`_`”) cu spații, făcând astfel textul mai ușor de citit. Aceasta este folosită pentru a formata descrierile și detaliile camerelor sau ale nodurilor din graf.

- **Metoda “`__load_graph`”**

Metoda privată “`__load_graph`” citește structura graficului dintr-un fișier text. Prima linie a fișierului indică numărul total de noduri din graf. Următoarele linii furnizează detalii despre fiecare nod, inclusiv coordonatele sale, eticheta, tipul de nod, și eventual informații adiționale cum ar fi numărul camerei sau detalii despre nodul respectiv. Această metodă parcurge fiecare linie din fișier, analizează informațiile și le stochează în două dicționare: “info”, care asociază fiecare nod cu detaliile sale, și “adj”, care reprezintă conexiunile dintre noduri sub forma listei de adiacență a grafului. Această metodă returnează numărul total de noduri, lista de adiacență, și detaliile nodurilor.

- **Metoda “`find_min_path`”**

Metoda “`find_min_path`” implementează algoritmul BFS (Breadth-First Search) pentru a găsi cel mai scurt drum între două noduri date, “start” și “finish”. Metoda întoarce calea minimă ca o listă de noduri parcurse de la “start” la “finish”. Modul de integrare al algoritmului BFS în funcția “`find_min_path`” este descrisă mai jos.

- **Inițializarea cozii și a dicționarului de vizitare:**

Algoritmul începe prin inițializarea unei cozi (q) pentru a stoca liste de două elemente, ce au pe prima poziție nodul ce urmează să fie explorat, iar pe a doua poziție calea de la nodul de start până la acesta. Acest lucru permite menținerea ordinii în care nodurile sunt procesate, garantând că nodurile sunt vizitate în ordine de la cel mai apropiat la cel mai îndepărtat față de nodul de start, iar calea parcursă este ușor de accesat. Se creează un dicționar viz care conține starea fiecărui nod (vizitat sau

nevizitat), pentru a se evita re-explorarea nodurilor deja procesate. Inițial toate nodurile sunt considerate ca nevizitate (le este asociată valoarea False în dicționarul viz).

- **Explorarea nodurilor din coadă:**

La început o listă formată din nodul de start și calea inițială, care conține doar nodul de start, este adăugată în coadă.

Cât timp coada nu este goală, algoritmul extrage primul element din coadă, și îl destructurează în `curr_node` (nodul curent) și `curr_path` (calea parcursă până la acel nod). Folosind aceste informații, algoritmul efectuează următoarele operații:

- verifică dacă nodul curent a fost deja vizitat. Dacă da, se trece la următorul element din coadă. Dacă nodul nu a fost vizitat anterior, acesta este marcat ca vizitat (`viz[curr_node] = True`).
- urmează să verifice dacă nodul curent este nodul de destinație (finish). Dacă este adevărat, metoda returnează `curr_path`, care reprezintă calea completă de la nodul de start până la nodul de destinație, algoritmul ajungând la final.
- dacă algoritmul nu a găsit încă nodul de final, se continuă prin verificarea dacă următorul vecin (`next_node`) al nodului curent (`curr_node`) a fost vizitat. Dacă vecinul nu a fost vizitat, acesta este adăugat în coadă împreună cu calea extinsă, care include nodul vecin la sfârșit (`curr_path + [next_node]`).
- pașii descriși anterior se repetă până când coada devine goală sau nodul de destinație este găsit.

- **Metoda “get_directions_for_path”**

Metoda “get_directions_for_path” primește un traseu reprezentat ca o listă de etichete de noduri și generează o listă de indicații de orientare pentru parcurgerea acelui traseu. Aceasta folosește un sistem de mapping pentru a traduce virajele în instrucțiuni simple cum ar fi “la stânga” sau “la dreapta”. De asemenea, gestionează schimbările de etaj, instruind utilizatorul să urce sau să coboare scările.

Metoda începe prin inițializarea unei liste goale “directions” care va stoca instrucțiunile de orientare generate pe parcursul traseului. De asemenea, sunt definite trei dicționare: “steer_mapping” pentru direcțiile de virare, “floor_directions” pentru direcțiile de urcare sau

coborâre între etaje și “floor_mapping” pentru traducerea codurilor de etaj în denumiri explicite.

Dacă nodul de start din traseu este de tip “s” (sală), se verifică detaliile asociate acestuia. Dacă detaliile conțin cuvântul “entrance”, se adaugă instrucțiunea “Enter the faculty”, altfel se generează instrucțiunea pentru ieșirea din camera specificată. Această verificare și generare a instrucțiunilor inițiale se face utilizând atributele “node_type”, “no_room” și “details” ale nodului de start.

Se inițializează variabila “changing_floor” cu valoarea False pentru a urmări dacă traseul implică schimbarea etajului. Apoi, metoda parcurge fiecare nod din traseu, începând de la al doilea nod până la penultimul nod. Pentru fiecare nod curent, se identifică nodul precedent și nodul următor. Dacă nodurile precedente, curente și următoare sunt toate de tip “h” (hol), se continuă fără a adăuga instrucțiuni, deoarece nu este necesară nicio schimbare de direcție.

Dacă nodul curent este de tip “he” (hol de etaj) și “changing_floor” este False, se setează “changing_floor” la True pentru a indica faptul că urmează o schimbare de etaj. Dacă nodul curent este tot de tip “he” și nodul următor este de tip “h”, se resetează “changing_floor” la False și se calculează diferența de nivel dintre nodul în care se părăsesc scările și cel în care încep să se folosească scări. Dacă diferența este pozitivă se va afișa o instrucțiune de urcare, altfel se va afișa de coborâre. Instrucțiunea corespunzătoare pentru urcarea sau coborârea scărilor este apoi adăugată la lista de instrucțiuni, specificând etajul la care se ajunge.

Pentru restul nodurilor din traseu, se iau câte trei noduri consecutive și se determină natura virajului în nodul mijlociu folosind metoda auxiliară “__orientation_test”. În lista “directions” este adăugată câte o indicație de orientare în funcție de tipul nodului curent și al nodului următor. Spre exemplu, dacă punctele sunt coliniare și nodul următor este de tip “s” (destinație), se adaugă instrucțiunea “Your destination is straight ahead”, iar dacă punctele sunt coliniare și nodul următor este de tip “he” (hol de etaj), se adaugă instrucțiunea “Go to the stairs in front of you”. Pentru alte cazuri, se generează instrucțiuni în funcție de contextul nodurilor adiacente.

În final, metoda returnează lista de instrucțiuni generate pentru traseul specificat, oferind utilizatorului indicații clare și precise pentru parcurgerea traseului de la nodul de start la nodul de destinație.

- **Metoda “__orientation_test”**

Metoda auxiliară și privată “__orientation_test” calculează direcția în care trebuie să se vireze la o intersecție dată de trei puncte consecutive (noduri) utilizând un test de orientare bazat pe determinant. Acesta returnează -1 pentru viraje la dreapta, 1 pentru viraje la stânga și 0 pentru direcții drepte (coliniare).

- **Metoda get_directions”**

Metoda “get_directions” folosește “find_min_path” pentru a găsi calea minimă între două noduri date și apoi apelează “get_directions_for_path” cu rezultatul metodei anterior menționate drept input pentru a genera indicațiile necesare pentru a urma această cale. Rezultatul este o listă de instrucțiuni de orientare pas cu pas de la nodul de start la cel destinație.

- **Metoda “get_rooms_info”**

Metoda “get_rooms_info” returnează o listă de detalii despre toate nodurile care reprezintă camere (au valoarea “s” în atributul “node_type”) din graf. Utilitatea acestei metode se remarcă în momentul în care sunt necesare informații specifice despre sălile clădirii între care se poate naviga, spre exemplu pentru afișarea în front-end a listei de săli pentru pornire sau destinație.

3.4 Pașii pentru configurarea inițială a aplicației

Pașii următori descriu instalarea și configurarea aplicației pentru Windows 11. Asigurați-vă că urmați fiecare pas în ordinea specificată pentru a evita problemele de configurare.

Pentru a începe, este necesar să aveți instalate Docker Desktop (v4.30), Python (v3.12) și Git (acesta trebuie configurat în prealabil).

Primul pas este să creați un nou folder în calculatorul dumneavoastră și deschideți terminalul în acesta, prin tastarea secvenței ”cmd” în bara de navigare a ferestrei Explorer și apăsați tasta Enter. Aceasta va deschide un terminal CMD direct în locația folderului unde doriți să clonați proiectul.

Următorul pas este să clonați repo-ul proiectului în acest folder. În terminalul CMD, rulați comanda: `git clone https://github.com/LauraElena16/MathInfoAcademy.git`. Aceasta va descărca toate fișierele din repo-ul GitHub în folderul dvs. local. După ce clonarea este completă, intrați în repo-ul clonat folosind comanda: `cd MathInfoAcademy`.

Următorul pas este să creați un volum Docker. Deschideți Docker Desktop și navigați la secțiunea “Volumes”. Creați un nou volum cu numele `proiect_app_licenta_vol` făcând click pe “Create Volume”. Apoi, importați arhiva `proiect_app_licenta_vol.tar.gz`, care se găsește în `MathInfoAcademy/Proiect`, în volumul creat anterior din Docker Desktop.

În folderul `MathInfoAcademy/Proiect`, adăugați un fișier `.env`, necesar pentru specificarea unor variabile de mediu pentru containerul Docker. Deschideți fișierul creat într-un editor de text (cum ar fi Notepad) și adăugați următorul conținut: `MYSQL_ROOT_PASSWORD=“Super_Secret_Password”`. Acesta va seta parola utilizatorului root pentru serverul MySQL ce va rula în Docker și va fi utilizat de aplicație.

În continuare, în folderul `MathInfoAcademy/Proiect/laura/laura`, adăugați un alt fișier `.env` cu următorul conținut:

```
DB_NAME=app_licenta
DB_USER=user_licenta
DB_PASSWORD=d6jQw3lSU7
DB_HOST=localhost
DB_PORT=3307
```

Pentru a crea un mediu virtual în cadrul proiectului, reveniți la terminalul CMD deschis anterior și utilizați comanda: `python -m venv venv`. După ce mediul virtual este creat, activați-l folosind comanda: `.\venv\Scripts\activate`. În terminal, pentru a intra în folderul `Proiect`, folosiți comanda: `cd Proiect`. Odată ce vă aflați în directorul `Proiect` instalați toate dependențele necesare rulând comanda: `pip install -r requirements.txt`. Aceasta va citi fișierul `requirements.txt` și va instala toate pachetele necesare pentru proiect. Este important să aveți mediul virtual activat în acest moment pentru ca toate pachetele să fie instalate în mediul virtual și nu în mediul global de Python.

Porniți containerul Docker utilizând comanda: `docker-compose up -d`. În final, pentru a rula serverul, intrați în folderul `laura` cu comanda: `cd laura`. Din acest director, rulați comanda: `python manage.py runserver`. Aceasta va porni serverul de dezvoltare Django, permițându-vă să accesați aplicația la adresa `http://127.0.0.1:8000`.

3.5 Fluxul de acțiuni al aplicației

Pentru a înțelege cum funcționează aplicația de gestionare a cursurilor din perspectiva utilizatorului și a backend-ului, vom parcurge pașii esențiali prin care trece un utilizator, de la navigarea pe site până la înregistrare și gestionarea activităților sale. Acest flux de utilizare este sprijinit de o structură bine definită atât pe partea de frontend, cât și pe partea de backend.

Înainte de a detalia fiecare pagină a aplicației web, aș dori să menționez că fiecare dintre acestea conține o bară de navigare obținută prin moștenirea șabloanelor, disponibilă în Django. Această bară conține logo-ul aplicației, o serie de butoane pentru navigarea la paginile principale ale aplicației și o zonă destinată autentificării utilizatorului, acestea două din urmă modificându-și conținutul dacă utilizatorul este autentificat sau nu. În cazul în care un utilizator este autentificat, bara de navigare conține opțiunile Home și Directions, iar în partea dreaptă are butoanele Log In și Register. În caz contrar, pe lângă cele două opțiuni apar și Activity și Grades, iar butoanele din partea dreapta sunt înlocuite cu un mesaj de salut.

3.5.1 Pagina de înregistrare

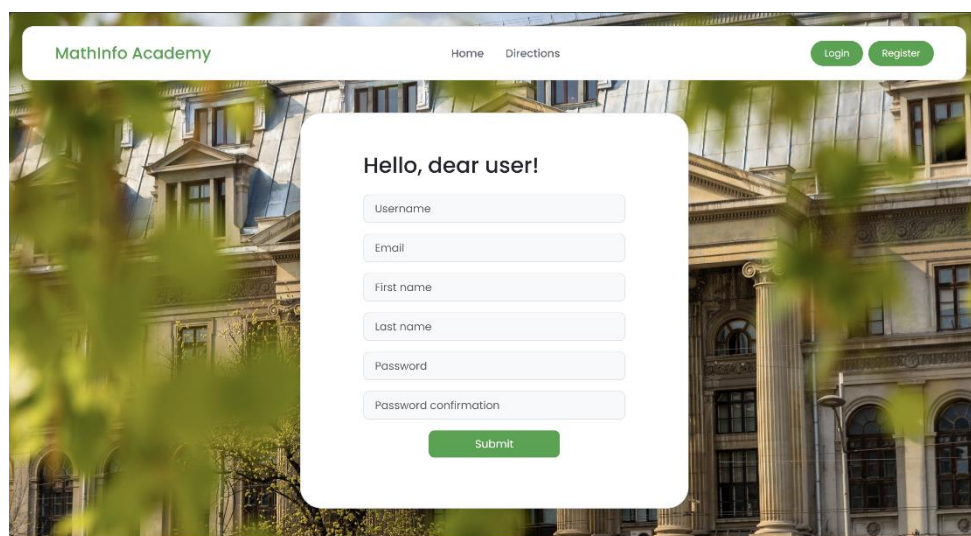


Figura 3.2 Pagina de înregistrare a unui cont nou

Când un utilizator decide să se înregistreze, acesta va da click pe butonul de "Register" din bara de navigare, ceea ce îl va duce la pagina de înregistrare. Această pagină este construită folosind șablonul register.html și conține un formular generat de Django pentru modelul User.

În urma completării și trimiterii formularul de înregistrare, serverul primește o cerere de tip POST către ruta /register/. Funcția register din fișierul views.py procesează această

cerere: dacă utilizatorul este deja autentificat, acesta este redirecționat înapoi la pagina principală (/), iar dacă nu este, funcția verifică validitatea datelor primite prin intermediul formularului RegistrationForm. Dacă datele sunt valide, noul cont este salvat în baza de date, iar utilizatorul este redirecționat către pagina de login.

După înregistrarea utilizatorului, administratorul trebuie să îi atribuie un rol specific, fie de student sau de profesor. Această atribuire este esențială pentru a permite utilizatorului accesul selectiv la anumite funcționalități ale aplicației, după cum au fost descrise în secțiunea 3.2.

3.5.2 Pagina de login

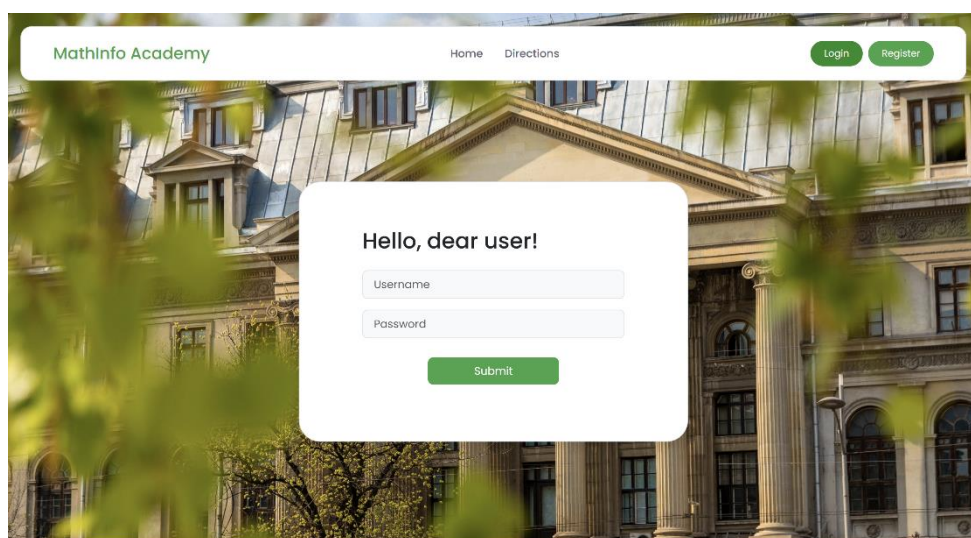


Figura 3.3 Pagina de autentificare a utilizatorului

După înregistrare, utilizatorul este redirecționat automat la pagina de autentificare unde trebuie să își introducă credențialele nou create pentru a se conecta. Alternativ, dacă utilizatorul nu este autentificat și alege să se conecteze mai târziu, poate accesa manual pagina de login prin butonul "Login" din bara de navigare.

Pagina de login (login.html) prezintă un formular de autentificare simplu, cerând utilizatorului să introducă numele de utilizator și parola.

Odată ce utilizatorul completează formularul și apasă butonul de login, datele sunt trimise către server printr-o cerere HTTP POST către ruta /login/. În backend, funcția user_login din views.py preia cererea și continua prin validarea credențialelor introduse utilizând auth.authenticate, iar dacă aceste date sunt corecte, sesiunea utilizatorului este inițiată folosind auth.login și acesta este redirecționat către pagina principală. Dacă, totuși, autentificarea

eșuează, utilizatorul este notificat despre eroare prin mesajul „Invalid credentials”, care este afișat sub formular, și utilizatorul este redirecționat înapoi la pagina de login pentru a reîncerca.

Odată autentificat, utilizatorul are acces la funcționalitățile complete ale aplicației, care sunt reflectate în bara de navigare prin apariția opțiunilor de navigare "Activity" și "Grades".

3.5.3 Pagina de home

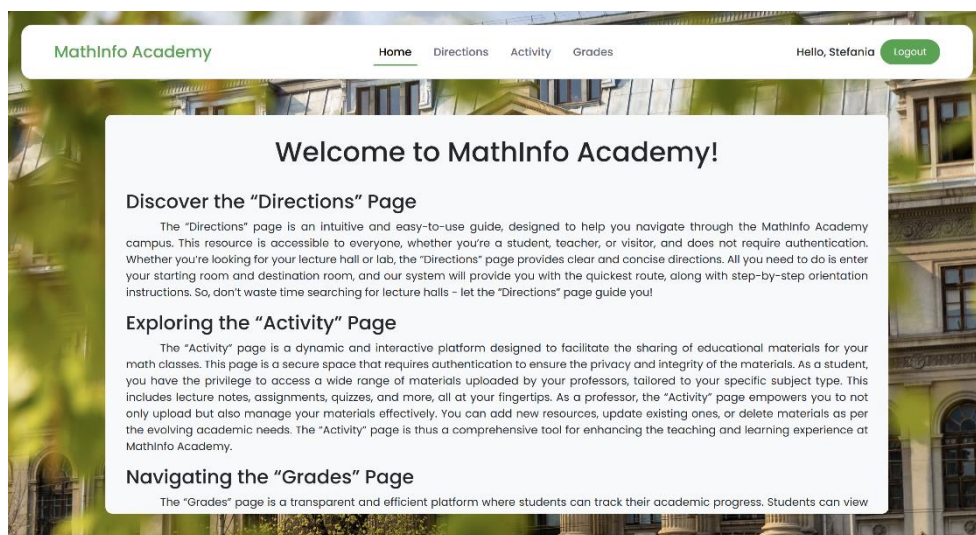


Figura 3.4 Pagina Home

Atunci când utilizatorul introduce adresa URL și solicită accesul la pagina principală a site-ului, browserul său trimite o cerere HTTP GET către serverul MathInfo Academy, ce este interceptată de funcția home din fișierul views.py. Odată ce cererea este procesată, serverul trimite înapoi clientului șablonul home.html, care definește structura și conținutul paginii principale. După încărcarea fișierului HTML în browser, utilizatorul este întâmpinat de un mesaj de bun venit și de câte o descriere pentru fiecare dintre paginile disponibile în bara de navigare.

3.5.4 Pagina Directions

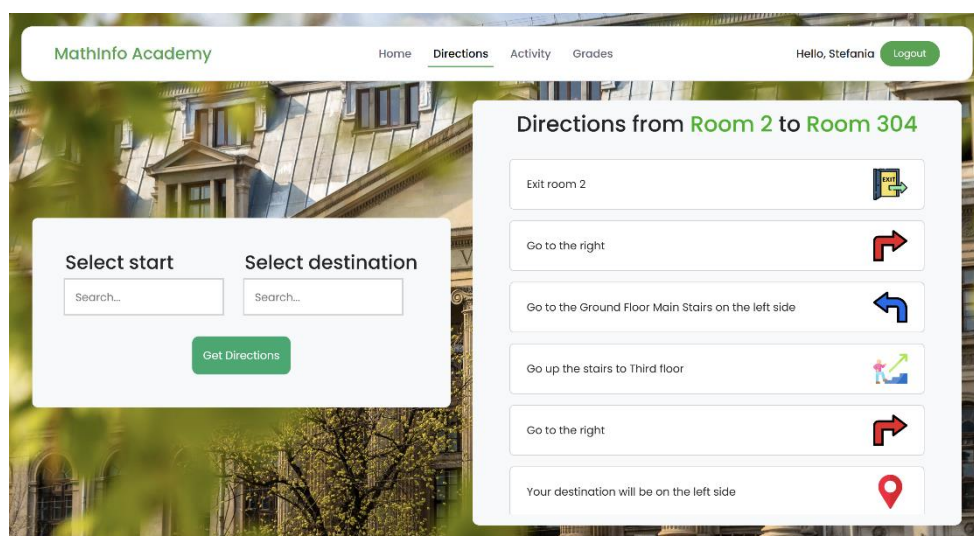


Figura 3.5 Pagina de indicații de orientare în facultate

Când utilizatorul accesează pagina Directions, acesta poate vedea două elemente de tip drop-down în partea stângă pentru selectarea camerei de start și a destinației, iar în partea dreaptă apare o zonă în care vor fi afișate indicațiile de orientare. Mai apoi, acesta are posibilitatea de a căuta și selecta o cameră de start și una drept destinație din listele respective. După selectarea capetelor, prin apăsarea butonului "Get Directions" se trimite o cerere de tip POST ce conține informațiile camerelor selectate, care declanșează procesul de calculare al indicațiilor de orientare.

Serverul încearcă să convertească valorile primite pentru start și finish în numere întregi, și verifică dacă aceste camere sunt valide și existente în graful asociat planului clădirii. În cazul în care una dintre validări eșuează, serverul setează un mesaj de eroare corespunzător pentru a informa utilizatorul despre problema întâmpinată. În schimb, dacă start-ul și finish-ul sunt camere valide, serverul utilizează metoda `get_directions` a clasei `Graph` pentru a genera indicațiile necesare deplasării între cele două camere, proces descris pe larg în secțiunea 3.3.

În urma calculării acestor indicații, server-ul generează răspunsul final prin funcția `render`, care include camerele sortate, indicațiile calculate, eventuale mesaje de eroare, și informațiile detaliate despre camerele de start și finish, în funcție de fiecare caz descris anterior. Utilizatorul primește astfel pagina reîncărcată pentru a afișa rezultatele, indicațiile de navigare între camerele selectate fiind prezentate într-un format clar și detaliat, împreună cu imagini sugestive pentru o mai bună orientare; în cazul în care există vreo eroare, cum ar fi selectarea unor camere invalide, aceasta este afișată pe pagină sub forma unui mesaj de eroare.

3.5.5 Pagina Activity

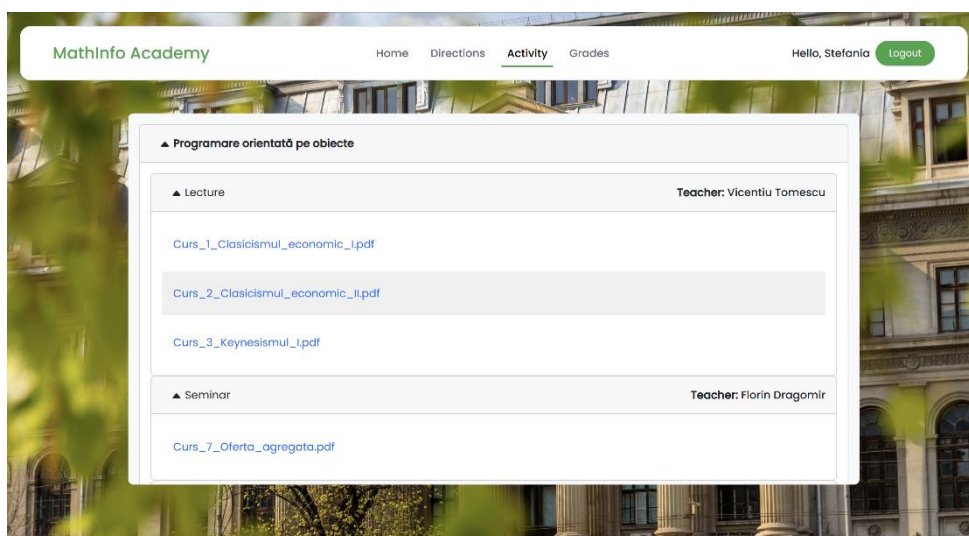


Figura 3.6 Pagina de activități văzută din perspectiva studentului

Când un utilizator accesează pagina "Activity", serverul primește o cerere de tip GET. Funcția activity din fișierul views.py este apelată pentru a procesa această cerere și a pregăti datele necesare pentru afișarea paginii. Înainte de toate, se verifică dacă utilizatorul este autentificat, iar în caz contrar, acesta este redirecționat către pagina principală.

Dacă utilizatorul autentificat este un student, se obține o listă de activități asociate acestuia (Student_Activity), iar pentru fiecare dintre acestea sunt extrase detalii despre activitatea profesorului (Teacher_Activity), inclusiv numele materiei, tipul activității didactice și numele profesorului. Toate aceste date sunt organizate în două structuri de tip dicționar pentru accesul facil la date în template-ul HTML. În una din aceste structuri sunt adăugate informații despre materialele asociate fiecărei activități didactice pentru a fi accesibile studenților. Folosind cele două dicționare, server-ul generează răspunsul prin randarea șablonului activity.html și îl trimite clientului. Acum studentul poate vedea în structurile de tip acordeon din pagină toate materiile și activitățile didactice la care este înscris, alături de materialele în format pdf pentru fiecare astfel de activitate. El poate vizualiza orice material dintre cele listate făcând click pe acesta.

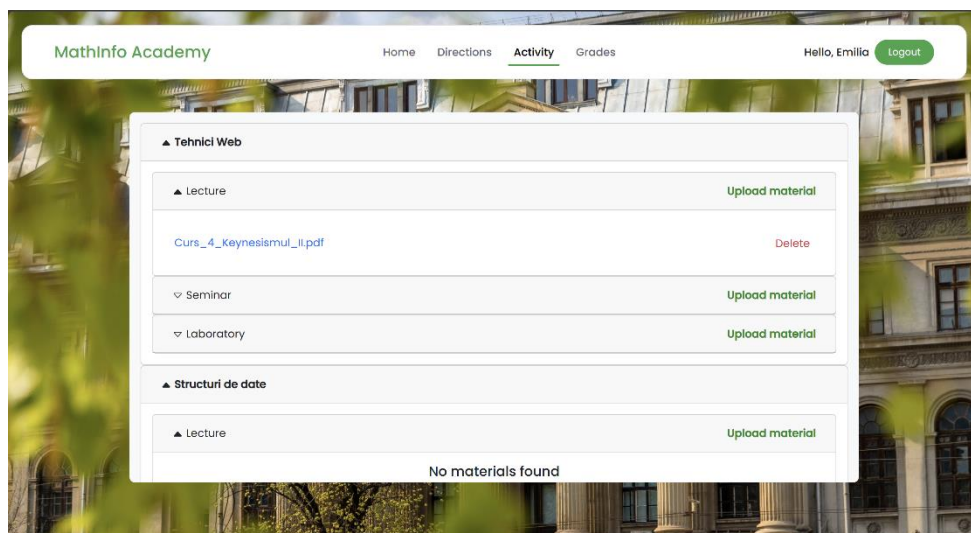


Figura 3.7 Pagina de activități văzută din perspectiva profesorului

În cazul în care un profesor accesează pagina "Activity", aplicația filtrează toate activitățile asociate profesorului autentificat prin intermediul modelului Teacher_Activity, iar aceste activități sunt colectate într-o listă. Aplicația construiește apoi două dicționare: info_dict și teach_act_dict. În primul dicționar, info_dict, aplicația organizează activitățile în funcție de numele cursului și tipul activității, iar în al doilea dicționar, teach_act_dict, stochează activitățile corespunzătoare pentru a facilita gestionarea lor ulterioară. Server-ul construiește răspunsul HTTP și îl întoarce către browser-ul utilizatorului, unde acesta poate vedea, similar ca în cazul studenților, o structură de tip acordeon cu materiile și fiecare activitate didactică susținută.

Spre deosebire de studenți, în cazul profesorilor secțiunile asociate activităților didactice au un buton pentru încărcarea unor noi materiale. La apăsarea acestuia se deschide o fereastră modală în care profesorul poate selecta și încărca un fișier pdf drept material pentru activitatea asociată prin trimiterea unei cereri de tip POST către ruta /upload_material/. De asemenea, în dreptul fiecărui material deja existent este prezent un buton "Delete", ce va iniția procesul de ștergere a materialului respectiv în urma trimiterii unei cereri către ruta /delete_material/.

3.5.6 Pagina Grades

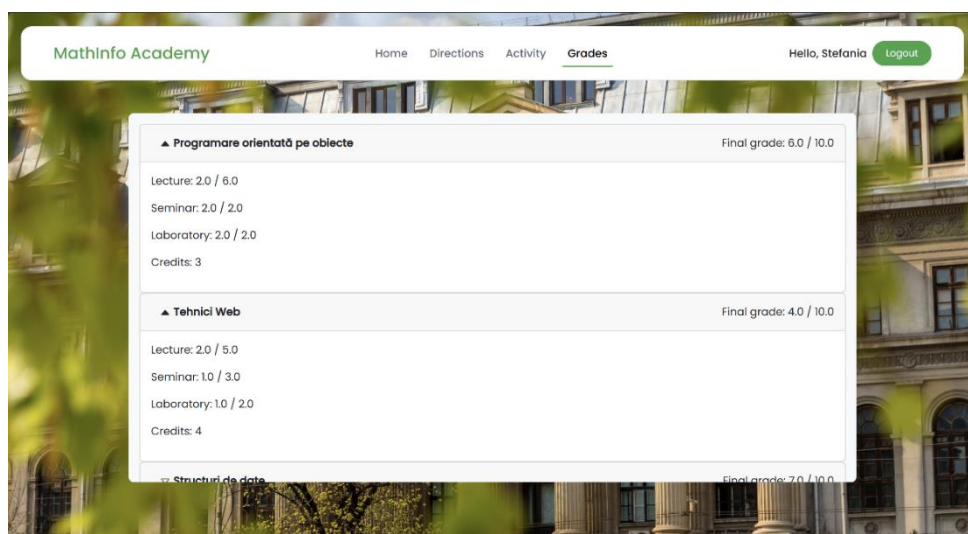


Figura 3.8 Pagina de note văzută din perspectiva studentului

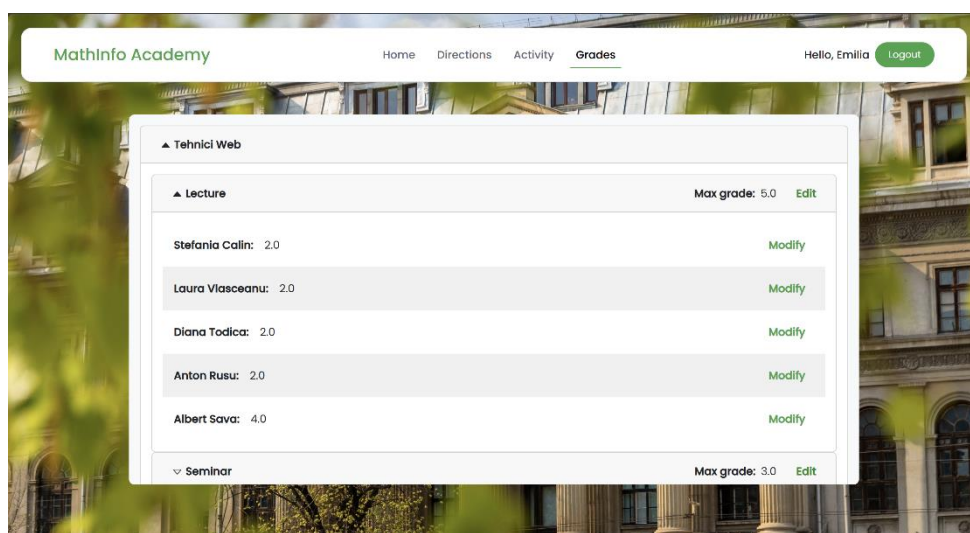


Figura 3.9 Pagina de note văzută din perspectiv profesorului

Când un utilizator accesează pagina "Grades", este verificat faptul că utilizatorul este autentificat, iar în caz negativ, acesta este redirecționat instantaneu către pagina principală pentru a împiedica accesul neautorizat. Dacă utilizatorul este însă autentificat, se execută funcția grades din views.py pentru a gestiona cererea primită.

În cazul unui utilizator de tip student, server-ul colectează toate informațiile necesare legate de materiile și activitățile didactice la care este înscris, inclusiv nota sa exactă și nota maximă de la fiecare activitate, dar și numărul de credite asociat materiei. În plus se calculează și nota finală de la fiecare materie însumând notele de la activitățile asociate și normalizând rezultatul pe o scară de la 0 la 10. Toate aceste informații sunt afișate mai apoi în șablonul grades.html, unde ele sunt aranjate, asemănător paginii "Activity" în structuri de tip acordeon.

De data aceasta fiecare antet asociat unei secțiuni extensibile conține numele materiei în partea stângă, iar în partea dreaptă este afișată nota finală la materia respectivă din maximum de 10. La click pe un astfel de antet este extins conținutul acestuia, unde se pot regăsi notele studentului pentru fiecare activitate, fiind clar specificată și nota maximă asociată acesteia. În plus, în josul acestei secțiuni apare și numărul de credite al materiei, pentru ca studentul să înțeleagă importanța materiei respective în planul său de școlarizare.

Dacă utilizatorul autentificat este de tip profesor și navighează la pagina "Grades", la server sunt căutate informațiile despre activitățile didactice susținute de acesta, incluzând detalii despre materie, studenții înscriși la aceste activități și notele acestora, dar și nota maximă asociată fiecărei activități. În front-end se afișează aceeași structură de tip acordeon, fiecare antet asociat unei activități conținând în partea dreaptă nota maximă specifică și un buton numit "Edit" ce deschide o modală, în urma apăsării, pentru schimbarea notei maxime a acelei activități. În această modală, profesorul poate introduce o nouă notă, iar la apăsarea butonului "Save Changes" este apelată prin metoda POST ruta `/modify_max_grade/` pentru modificarea efectivă a notei. În plus, pe lângă schimbarea notei maxime a unei activități, profesorul poate vedea informații despre toți studenții înscriși la activitățile pe care le predă, prin expandarea structurilor de tip acordeon aferente fiecărei activități, precum numele complet al acestora și nota acordată. Mai mult decât atât, în dreptul fiecărui student apare un buton cu textul "Modify" la apăsarea căruia apare o altă fereastră modală, de data aceasta pentru schimbarea notei studentului la activitatea respectivă. Profesorul poate introduce nota în câmpul vizibil, iar la apăsarea butonului "Save Changes" este trimisă cererea de tip POST către `/grade_student/` pentru acordarea notei studentului.

În ambele ferestre modale apare un mesaj personalizat pentru ca profesorul să știe ce notă editează, iar rezultatul operației de schimbare, atât a notei maxime cât și a notei unui student este anunțat prin mesaje temporare intuitive primite de la server și afișate în partea de sus a ecranului.

4. Concluzie

Aplicația web dezvoltată în cadrul acestui proiect reprezintă o soluție inovatoare și eficientă pentru problemele curente întâmpinate de studenții și profesorii de la Facultatea de Matematică și Informatică a Universității din București. Prin implementarea acestei aplicații, utilizatorii vor beneficia de o platformă centralizată și intuitivă pentru gestionarea materialelor didactice și a notelor, eliminând astfel nevoia de a naviga prin multiple platforme și surse de informare. Acest lucru va simplifica procesul de învățare și de pregătire academică, economisind timp prețios atât pentru studenți, cât și pentru profesori.

Una dintre cele mai semnificative avantaje ale aplicației este funcționalitatea de orientare în clădire, care va ajuta studenții să găsească rapid și ușor sălile destinate activităților didactice. Aceasta nu doar că va reduce timpul pierdut în căutarea locațiilor, dar va diminua și nivelul de stres asociat cu întârzierea la cursuri.

Pe lângă aceste beneficii evidente, aplicația contribuie și la îmbunătățirea comunicării și colaborării între studenți și profesori. Prin intermediul platformei, profesorii pot distribui rapid și eficient materialele de curs, asigurându-se că toți studenții au acces la informațiile necesare. De asemenea, notele pot fi încărcate și vizualizate într-un mod transparent și organizat, facilitând monitorizarea progresului academic.

Cu toate că aplicația satisface toate cerințele actuale ale utilizatorilor, este inevitabil ca acestea să se modifice cu timpul. Astfel, aplicația prezintă unele limitări ce pot fi reduse prin adăugarea unor funcționalități adiționale în viitor precum: un calendar în care vor fi vizibile activitățile didactice din fiecare săptămână, verificarea sălilor disponibile la o anumită oră dintr-o zi specificată sau ținerea evidenței prezențelor la activitățile de curs direct în aplicație.

În final, dezvoltarea acestei aplicații a reprezentat atât o provocare pentru mine, cât și o oportunitate de învățare foarte valoroasă. Prin implementarea acestei platforme online am dobândit o imagine clară asupra tuturor pașilor necesari construirii unei aplicații web folosind tehnologii populare și sigure, dar și a componentelor ce interacționează pentru a o face să funcționeze.

5. Bibliografie

- Dauzon, S., Bendoraitis, A., Ravindran, A. (2016). *Django: Web Development with Python*. Packt Publishin. Accesat la data 12.06.2024 la adresa:
[https://books.google.ro/books?hl=ro&lr=&id=vKjWDQAAQBAJ&oi=fnd&pg=PP1&dq=%09Dauzon,+S.,+Bendoraitis,+A.,+Ravindran,+A.+\(2016\).+Django:+Web+Development+with+Python.+Packt+Publishin&ots=2oOlwCn_sM&sig=gIQlevaI39ao5PDU6nf8roctsc&redir_esc=y#v=onepage&q=%09Dauzon%2C%20S.%2C%20Bendoraitis%2C%20A.%2C%20Ravindran%2C%20A.%20\(2016\).%20Django%3A%20Web%20Development%20with%20Python.%20Packt%20Publishin&f=false](https://books.google.ro/books?hl=ro&lr=&id=vKjWDQAAQBAJ&oi=fnd&pg=PP1&dq=%09Dauzon,+S.,+Bendoraitis,+A.,+Ravindran,+A.+(2016).+Django:+Web+Development+with+Python.+Packt+Publishin&ots=2oOlwCn_sM&sig=gIQlevaI39ao5PDU6nf8roctsc&redir_esc=y#v=onepage&q=%09Dauzon%2C%20S.%2C%20Bendoraitis%2C%20A.%2C%20Ravindran%2C%20A.%20(2016).%20Django%3A%20Web%20Development%20with%20Python.%20Packt%20Publishin&f=false)
- Even, S., Even, G. (2012). *Graph Algorithms*. Cambridge University Press. Accesat la data 12.06.2024 la adresa:
https://books.google.ro/books?hl=ro&lr=&id=m3QTSMYm5rkC&oi=fnd&pg=PR5&dq=graph+algorithms&ots=ItzdmkA_Qm&sig=1_HXaHwwnS1OYlf0oOgaYJmOtK8&redir_esc=y#v=onepage&q&f=false
- Sacristan, V. (n.d). *Basic tool: orientation tests*. Discrete and Algorithmic Geometry Facultat de Matematiques i Estadistica Universitat Politecnica de Catalunya. Accesat la data 12.06.2024 la adresa:
<https://dccg.upc.edu/people/vera/wp-content/uploads/2012/10/DAG-OrientationTests.pdf>
- Spurlock, J. (2013). *Bootstrap: Responsive Web Development*. O'Reilly Media, Inc. Accesat la data 12.06.2024 la adresa:
https://books.google.ro/books?hl=ro&lr=&id=LZm7Cxgi3aQC&oi=fnd&pg=PR2&dq=bootstrap+framework&ots=eYYGzxHnHG&sig=cbeMxQdkEvYYyWkR78hZBvT_3Vs&redir_esc=y#v=onepage&q=bootstrap%20framework&f=false
- Stan, A. (2022). *Introducere în python folosind google colab*. UTPRESS. Accesat la data 12.06.2024 la adresa:
<https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/593-0.pdf>
- Stefanescu, M. (2023). *Tutorial HTML pentru începători. Istoria limbajului și exemple practice*. Accesat la data de 30.04.2024 la adresa:
<https://code-it.ro/tutorial-htmlpartea-1-ce-este-html/>

- Stuehring, S. (2002). *MySQL Bible*. Wiley Publishing, Inc. Accesat la data 12.06.2024 la adresa:
http://box.cs.istu.ru/public/docs/other/_New/Books/Data/DB/MySQL/MySQL%20Bible.pdf
- Documentația oficială Docker. Accesat la data 12.06.2024 la adresa:
<https://docs.docker.com/get-started/overview/>
- Documentația oficială MySQL. Accesat la data 12.06.2025 la adresa:
<https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html>
- Documentația Bootstrap de pe website-ul W3Schools. Accesat la data 12.06.2024 la adresa:
https://www.w3schools.com/bootstrap5/bootstrap_get_started.php
- Documentația oficială Django. Accesat la data 12.06.2024 la adresa:
<https://docs.djangoproject.com/en/5.0/ref/urls/>
- Code Jungle, *Bootstrap 5 | How to create a Responsive Navbar | Step by Step Tutorial*, canalul de Youtube Code Jungle, publicat în octombrie 2023. Accesat la data 12.06.2024 la adresa:
<https://www.youtube.com/watch?v=h5apE3E72wY&list=PLHzLPEvuUJ2rFBNbyWTckmqOcizxicseq>
- Ludiflex, *Responsive login page created with Bootstrap 5 and HTML & CSS*, canalul de Youtube Ludiflex, publicat în martie 2023. Accesat la data 12.06.2024 la adresa:
<https://www.youtube.com/watch?v=oF28ns9eVdc&list=PLHzLPEvuUJ2rFBNbyWTckmqOcizxicseq&index=3>
- Documentația oficială Bootstrap. Accesat la data de 12.06.2024 la adresa:
<https://getbootstrap.com/docs/5.0/components/accordion/>
- Documentația oficială Bootstrap. Accesat la data de 12.06.2024 la adresa:
<https://getbootstrap.com/docs/5.0/components/modal/>
- Reddy, N., *Django Tutorial for Beginners | Full Course*, canalul de Youtube Telusko, publicat în iunie 2019. Accesat la data 12.06.2024 la adresa:
<https://www.youtube.com/watch?v=OTmQOjsl0eg&list=PLHzLPEvuUJ2rFBNbyWTckmqOcizxicseq&index=4>