Universitatea Națională de Știință și Tehnologie POLITEHNICA București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

## *Site web – bază de date studenți*

###### Proiect Tehnologii de programare în Internet

Student

*Maria-Cătălina Ionescu*

##### Anul

2024

**Cuprins:**

* Introducere
* Studiul și analiza termenilor tehnici
* Proiectare, funcționare, implementare
* Descrierea tehnologiilor utilizate
* Descrierea mediului de lucru
* Concluzii

Introducere

Site-ul web „Bază de date studenți” urmărește implementarea diferitelor limbaje de programe pentru crearea unei interfețe de utilizator prietenoase în care utilizatorul are posibilitatea de a accesa o bază de studenți. Acesta este destinat gestionării informațiilor despre studenți, oferind o platformă centralizată pentru administrarea datelor acestora. De la înregistrarea și editarea informațiilor personale ale studenților, până la vizualizarea detaliilor acestora și gestionarea fișierelor atașate, platforma oferă o gamă largă de funcționalități.

Pentru dezvoltarea și stilizarea acestui site au fost folosite mai multe limbaje de programare precum: Python Flask, HTML/CSS, JavaScript, Ajax și SQL, dar și platforme de design precum Canva, Color Pallete Chooser.

Studiul și analiza termenilor tehnici

**Python Flask** este un micro-framework web care permite dezvoltatorilor să creeze aplicații web rapid și eficient. Este ușor de învățat, flexibil și oferă un set de instrumente esențiale pentru dezvoltarea aplicațiilor web. Iată o prezentare generală despre Python Flask și cum l-am utilizat în site-ul dvs. Flask este denumit un micro-framework deoarece oferă doar componentele esențiale pentru construirea unei aplicații web. Nu include un ORM (Object-Relational Mapping) sau unelte de validare a formularelor, spre deosebire de framework-uri mai complexe. Permite adăugarea de extensii și pachete pentru a adăuga funcționalități specifice (autentificare, bază de date, formulare etc.) în funcție de nevoile proiectului. Flask promovează un cod simplu și clar, ceea ce face ușoară înțelegerea și modificarea aplicațiilor.

**AJAX** este o abreviere pentru "Asynchronous JavaScript and XML" (JavaScript și XML asincron) și reprezintă o tehnologie folosită în dezvoltarea aplicațiilor web pentru a permite comunicația asincronă între client (browser) și server, fără a reincărca întreaga pagină web. Această capacitate de a face solicitări către server și de a primi răspunsuri în fundal, în timp ce utilizatorul interacționează cu pagina, este esențială pentru crearea de interfețe web interactive și dinamice. Componentele principale ale AJAX sunt:

1. **JavaScript**: AJAX folosește JavaScript pentru a iniția și a gestiona solicitările către server și pentru a procesa răspunsurile primite. **JavaScript e**ste un limbaj de programare utilizat în principal pentru a adăuga interactivitate paginilor web. Rulează în browser-ul utilizatorului, ceea ce îl face ideal pentru manipularea elementelor DOM (Document Object Model) și gestionarea evenimentelor (click, hover, submit etc.). Permite dezvoltatorilor să creeze efecte vizuale, să valideze datele introduse de utilizatori și să efectueze modificări dinamice ale conținutului paginii.
2. **XMLHttpRequest (XHR)**: A fost prima metodă de bază utilizată pentru a efectua solicitări HTTP asincrone către server din JavaScript. Deși numele său include "XML", XHR poate manipula și alte tipuri de date, nu doar XML. Este cunoscută pentru a permite încărcarea datelor de pe server fără a reîmprospăta pagina complet.

**HTML și CSS** sunt două componente fundamentale în dezvoltarea web și sunt folosite pentru a crea și stiliza pagini web.

1. **HTML (HyperText Markup Language)**:
   * Este limbajul de marcă pentru structurarea și organizarea conținutului unei pagini web.
   * Defineste elementele de bază ale unei pagini web, cum ar fi titluri, paragrafe, imagini, link-uri etc.
   * Oferă structura semantică a paginii, permițând browserelor și altor agenți să înțeleagă cum să afișeze și să interpreteze conținutul.
2. **CSS (Cascading Style Sheets)**:
   * Este limbajul folosit pentru a stiliza și a formata elementele HTML pentru a le face atrăgătoare și ușor de citit pentru utilizatori.
   * Permite controlul asupra aspectului vizual al paginii web, inclusiv a culorilor, fonturilor, marginilor, paddingului, layout-ului și multe altele.
   * Utilizează reguli și selectori pentru a aplica stiluri diferite elementelor în funcție de nevoi.

**SQL (Structured Query Language)** este un limbaj de programare utilizat pentru gestionarea și manipularea bazelor de date relaționale. SQLite3 este o implementare ușoară și simplă a SQL, care este adesea utilizată în aplicații de dimensiuni mici și mijlocii, precum și în situații în care nu este necesar un server de bază de date separat. SQL este un limbaj standardizat pentru gestionarea bazelor de date relaționale. Este utilizat pentru a defini structura datelor (tabele, coloane, chei străine), pentru a insera, a actualiza și a șterge date, precum și pentru a interoga datele pentru a obține informații specifice. SQL include comenzi fundamentale precum SELECT (pentru a extrage date dintr-o bază de date), INSERT (pentru a adăuga date noi), UPDATE (pentru a actualiza date existente) și DELETE (pentru a șterge date). Aceste comenzi sunt esențiale pentru manipularea eficientă a datelor. SQL este ideal pentru bazele de date relaționale, în care datele sunt organizate în tabele cu relații definite între ele. Acest lucru permite gestionarea și interogarea eficientă a datelor structurate.

**Jinja2** este un motor de șabloane (template engine) pentru limbajul de programare Python. A fost creat pentru a gestiona șabloanele într-un mod eficient și flexibil în aplicațiile web și nu numai. Jinja2 folosește o sintaxă ușor de înțeles și de folosit pentru a defini și manipula șabloanele. Aceasta include utilizarea marcatorelor {{ }} pentru variabile, {% %} pentru blocuri de control (cum ar fi if, for, while) și {# #} pentru comentarii. De asemenea este integrat strâns cu Python, ceea ce permite utilizarea directă a codului Python în șabloanele Jinja2. În contextul framework-ului web Flask, Jinja2 este folosit pentru a genera pagini HTML dinamice, bazate pe datele procesate de server. Suportă moștenirea șabloanelor, o funcționalitate care permite definirea unui șablon de bază (sau șablon părinte) și extinderea acestuia în alte șabloane (sau șabloane copil). Acest lucru facilitează structurarea și reutilizarea codului HTML în aplicații mari și complexe.

**Bootstrap** este un framework front-end open-source pentru dezvoltarea de interfețe web și aplicații mobile. Bootstrap oferă un set de fișiere CSS pre-stilizate și responsabile, care permit dezvoltatorilor să creeze rapid interfețe web atractive și responsive. Aceste stiluri includ grid-uri responsive, componente UI (butoane, formulare, meniuri dropdown, alerte etc.) și stiluri de tipografie. Pe lângă stilurile CSS, Bootstrap include și o colecție de componente JavaScript interactivi, cum ar fi carusele, modale, tabele interactive, și multe altele. Aceste componente sunt gata de utilizare și pot fi integrate ușor în site-ul tău.

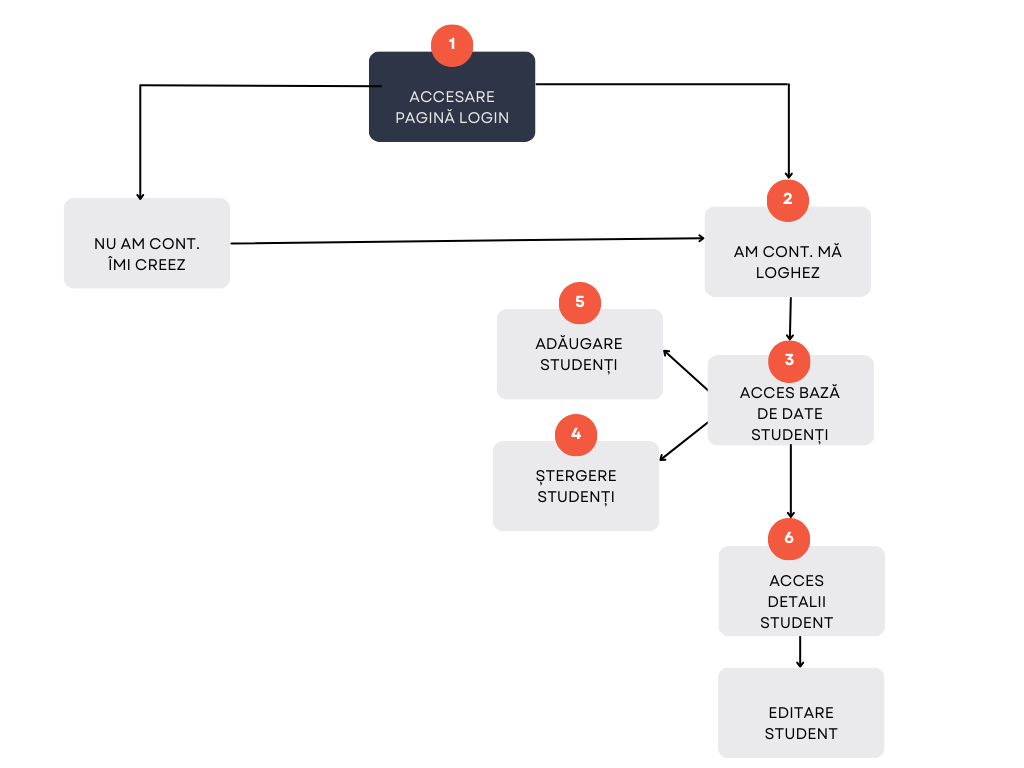
În codul Python Flask au fost implementate anumite componente și biblioteci. Aceste componente și biblioteci sunt fundamentale în dezvoltarea aplicațiilor web cu Flask și Python, oferind funcționalități esențiale pentru gestionarea rutei, a sesiunilor, a bazelor de date, a fișierelor încărcate și a altor aspecte cheie ale dezvoltării web.

1. **Flask**: Este un framework web pentru Python care facilitează dezvoltarea rapidă a aplicațiilor web. Folosește conceptul de rutare pentru a mapa URL-urile către funcțiile Python (rutele) care le gestionează și permite utilizarea de șabloane Jinja2 pentru generarea paginilor web.
2. **render\_template**: Funcția din Flask folosită pentru a afișa șabloane HTML. Permite pasarea datelor din Python (prin intermediul dicționarelor) către șabloane pentru a le afișa dinamic.
3. **request**: Obiectul care conține informații despre cererea HTTP curentă (cum ar fi datele formularului trimise prin POST).
4. **redirect**: Funcție pentru a redirecționa cererea către altă rută sau URL.
5. **url\_for**: Funcție care generează un URL pentru o anumită rută în funcție de numele acesteia.
6. **session**: Obiect care stochează datele sesiunii utilizatorului între cereri. Utilizat pentru gestionarea sesiunilor utilizatorilor autentificați.
7. **jsonify**: Funcție pentru a serializa obiecte Python în format JSON. Utilizată adesea în aplicațiile web pentru a răspunde la cereri AJAX cu date JSON.
8. **flash**: Mecanism pentru a afișa mesaje flash în aplicație. Mesajele flash sunt folosite pentru a transmite feedback utilizatorilor (de exemplu, mesaje de succes sau erori) și sunt afișate în mod tipic în următoarea pagină generată.
9. **os**: Modul Python pentru a interacționa cu sistemul de operare subiacent. Folosit în exemplul dat pentru a crea directoare pentru încărcarea fișierelor și pentru a asigura existența acestora.
10. **sqlite3**: Modul Python pentru a interacționa cu baza de date SQLite. Folosit pentru a crea, gestiona și interoga baza de date SQLite folosită în aplicație.
11. **generate\_password\_hash și check\_password\_hash**: Funcții din Werkzeug pentru a genera și verifica hash-uri securizate ale parolelor. Utilizate pentru a stoca parolele într-un mod sigur în baza de date.
12. **secure\_filename**: Funcție din Werkzeug utilizată pentru a obține un nume de fișier sigur și corect formatat pentru fișierele încărcate de utilizatori.
13. **wraps**: Decorator utilizat pentru a copia metadatele funcției wrapper în funcția decorată. Utilizat în exemplul dat pentru a menține metadatele funcțiilor decorate, cum ar fi numele și docstring-urile.
14. **re**: Modul Python pentru lucrul cu expresii regulate. Folosit în exemplul dat pentru a valida adresele de email folosind expresii regulate.

Proiectare și funcționare

În contextul unui site web, proiectarea și funcționarea sunt două aspecte fundamentale care se concentrează pe modul în care site-ul este conceput și implementat pentru a îndeplini obiectivele sale și pentru a oferi o experiență plăcută utilizatorilor.

**Schemă de funcționare site web**

**

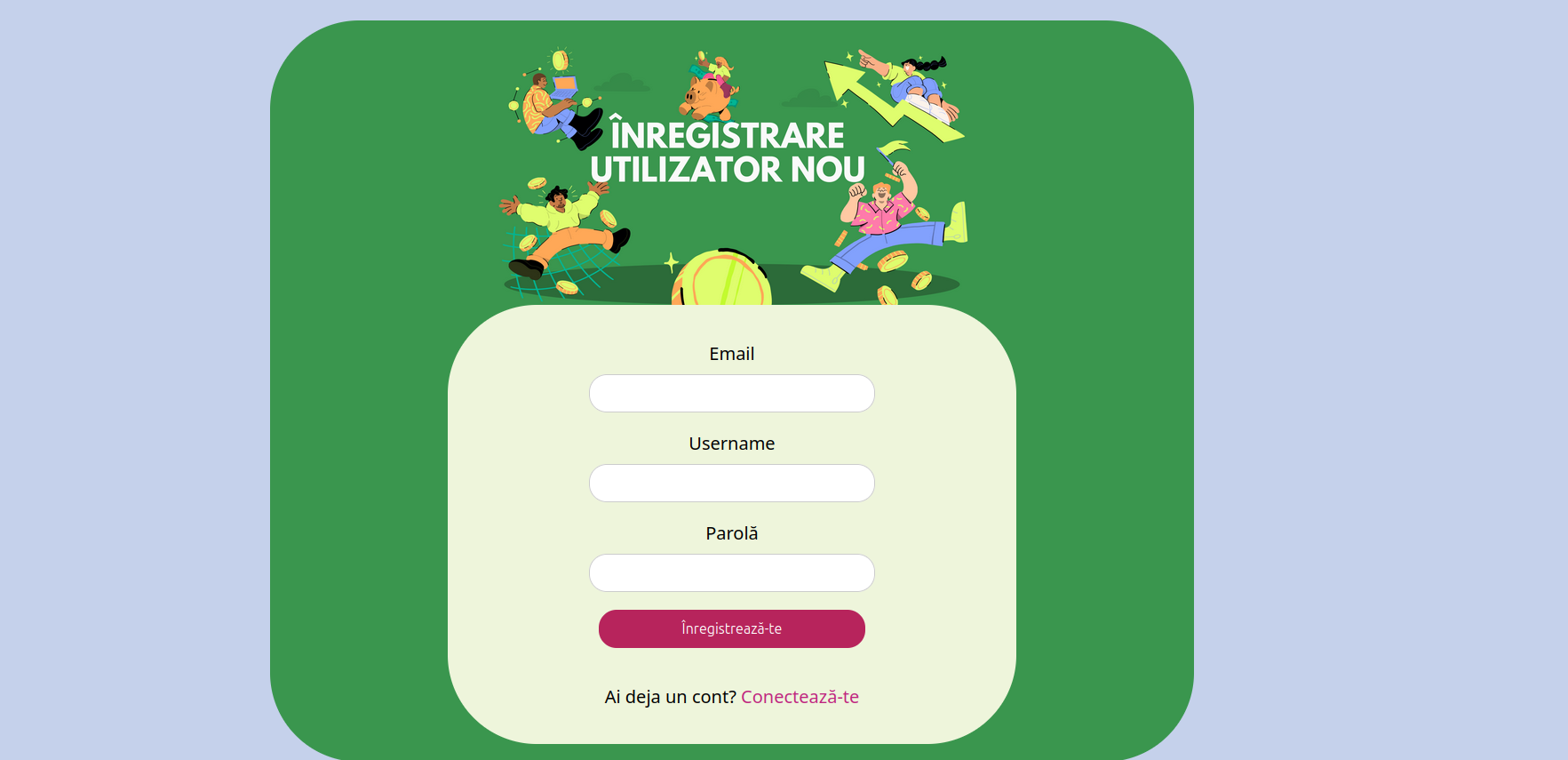
**Proiectarea site-ului web**

Site-ul web a fost proiectat pentru folosirea unei baze de date cu studenți de către un utilizator pentru un management mai ușor al acestora prin intermediul unei interfețe de utilizator. S-au folosit două baze de date diferite, una se ocupă de înregistrarea utilizatorilor, iar cealaltă de cea a studenților.

Odată ce utilizatorul accesează pagina web este întâmpinat de formularul de login care dă accesul la conținutul site-ului odată ce utilizatorul se conectează cu un username și o parolă înregistrate. După ce se va loga cu succes, va fi redirecționat la pagina cu studenți.

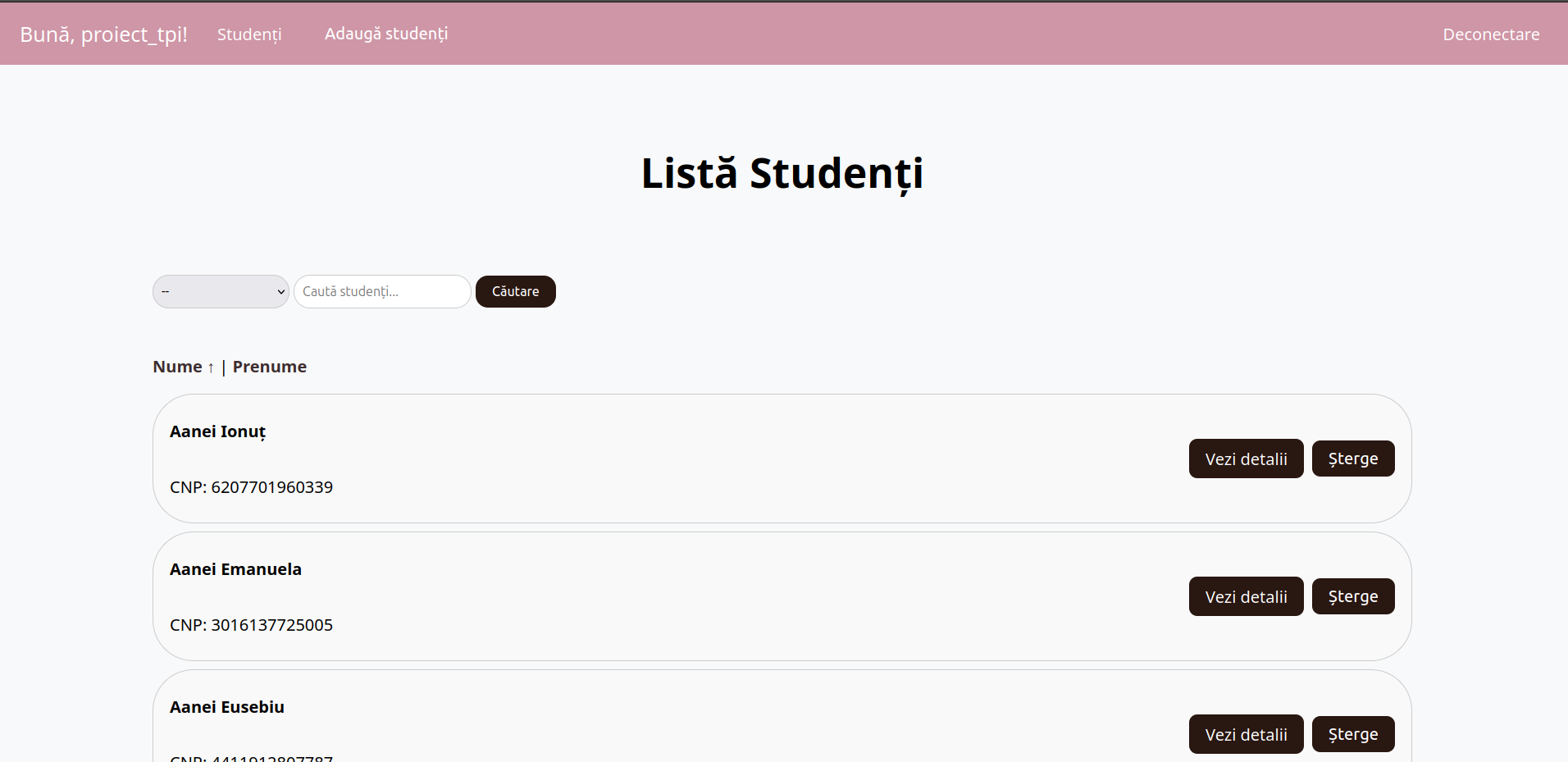


Dacă utilizatorul nu are un cont cu care să acceseze site-ul acesta are posibilitatea de a-și crea propriul cont. Pentru înregistrare acesta trebuie să furnizeze un username, o parolă și un email. Username-ul și email-ul trebuie să fie unice (niciun utilizator nu va putea să se înregistreze cu același username sau email), altfel un mesaj de eroare îi va apărea pe ecran. Parola trebuie obligatoriu să conțină minim 8 caractere, o literă mare și o cifră.



După ce s-a înregistrat cu succes, utilizatorul este redirecționat pe pagina de login unde se va putea conecta.

Odată ajuns pe pagina cu studenții, utilizatorului i se va afișa pe ecran o listă cu studenți și o bară de navigație.

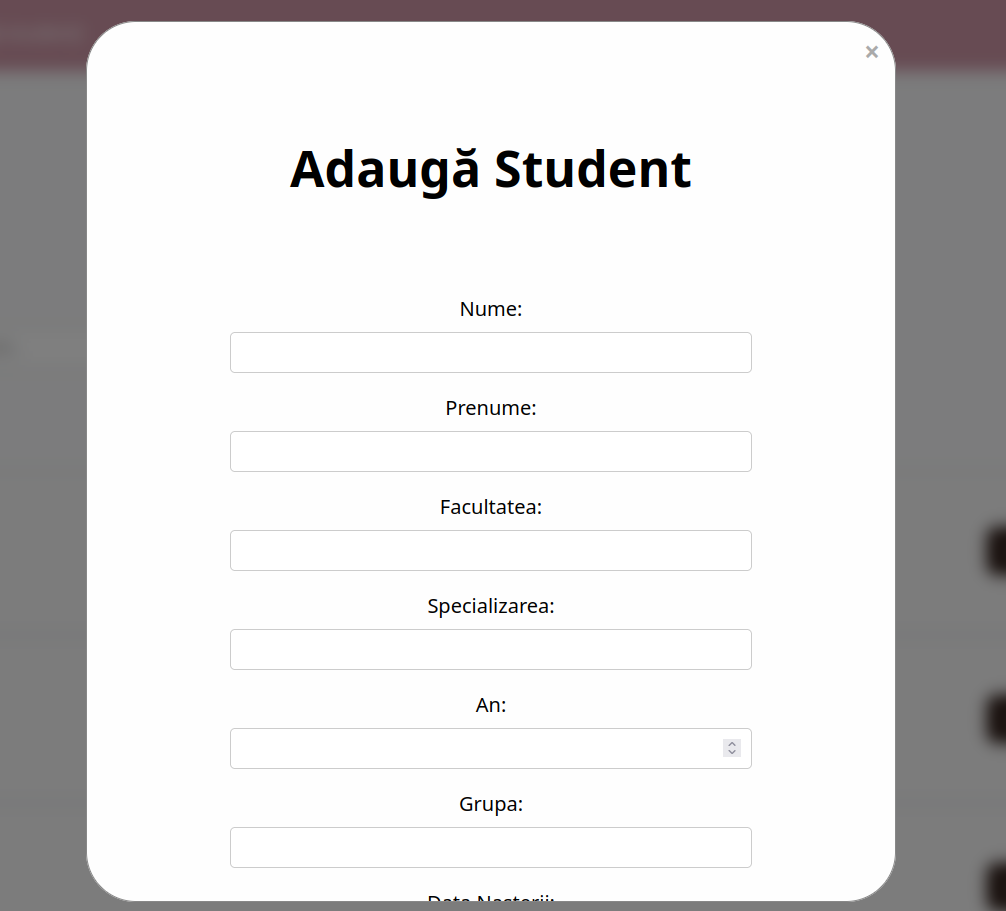


Bara de navigație (sau meniul de navigare) este o componentă esențială a designului site-ului și servește mai multor scopuri importante:

1. **Facilitarea navigării**: Bara de navigație oferă utilizatorilor o metodă simplă și clară de a accesa diferitele secțiuni ale site-ului. Aceasta permite utilizatorilor să se orienteze rapid și să găsească informațiile sau funcționalitățile dorite fără a se pierde în pagini sau secțiuni neclare.
2. **Structurarea informației**: Meniul de navigare ajută la organizarea și structurarea conținutului site-ului într-un mod logic și ușor de urmărit. Acest lucru contribuie la o experiență mai plăcută și intuitivă pentru utilizatori.
3. **Consistență**: Bara de navigație contribuie la consistența designului site-ului. Utilizatorii pot reveni ușor la meniu pentru a naviga în alte secțiuni, indiferent de pagina pe care se află în acel moment.

### Funcționalitatea butoanelor din bara de navigație:

1. **Mesaj de întâmpinare**: Acest mesaj întâmpină utilizatorul cu mesajul Bună, nume utilizator!
2. **Studenți:** Acest buton ar duce utilizatorii la o secțiune dedicată gestionării și vizualizării informațiilor despre studenți. Aici utilizatorii pot vedea o listă de studenți, detalii despre fiecare student sau șterge informații despre studenți.
3. **Adăugare student:** Acest buton ar permite utilizatorilor să adăuge un nou student în baza de date. Acest lucru facilitează înscrierea și actualizarea datelor studenților în mod eficient. Studenții pot fi adăugați în baza de date după următoarele criterii:
   * + - nume
       - prenume
       - facultate
       - specializare
       - an
       - grupă
       - data nașterii
       - CNP - unic
       - număr de telefon - unic
       - email - unic
       - adresă
       - cetățenie
       - etnie
       - poză

 Adăugarea de date care sunt unice încă odată nu va fi posibilă.

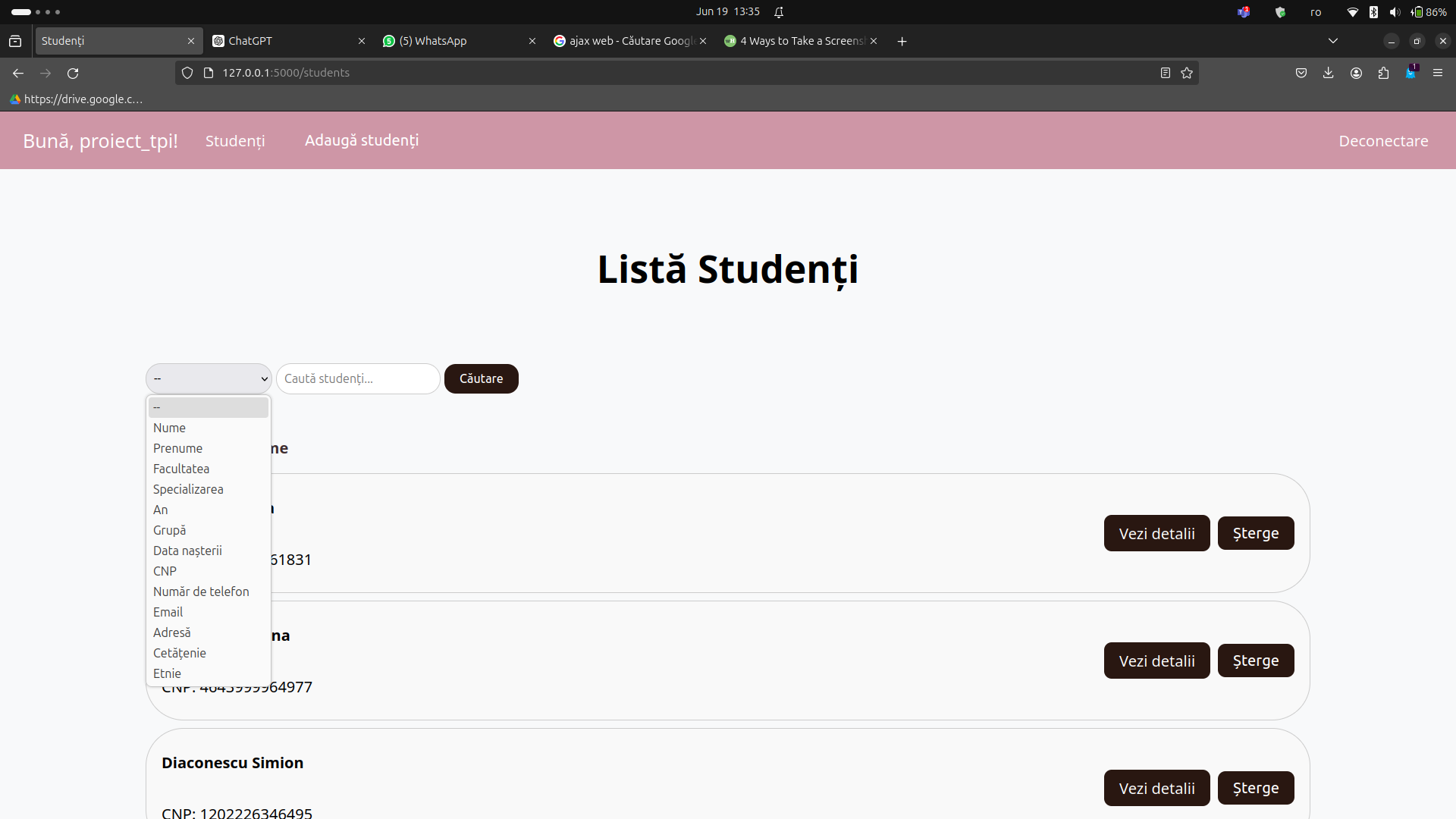
1. **Deconectare**: Acest buton oferă utilizatorilor posibilitatea de a se deloga din contul lor curent. Acest lucru este crucial pentru securitatea utilizatorilor și pentru a permite accesul la site doar utilizatorilor autorizați.

Conținutul paginii cu studenți conține un meniu de search și sortare și lista cu studenții.

Meniul de search și sortare permite utilizatorilor să căută și să sorteze studenții în funcție de anumite câmpuri.

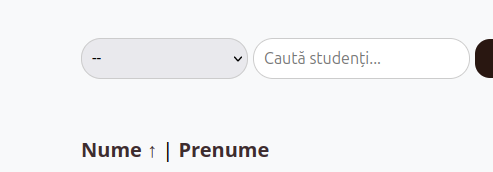
**Meniul de search:**

* + Utilizatorii pot căuta studenți după diferite criterii, selectate dintr-un meniu dropdown și introduse într-un câmp de text.
  + Criteriile de căutare includ: nume, prenume, facultatea, specializarea, an, grupa, data\_nasterii, cnp, numar\_de\_telefon, email, adresa, cetatenie, etnie.

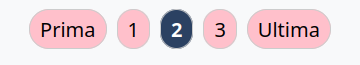


**Opțiuni de sortare:**

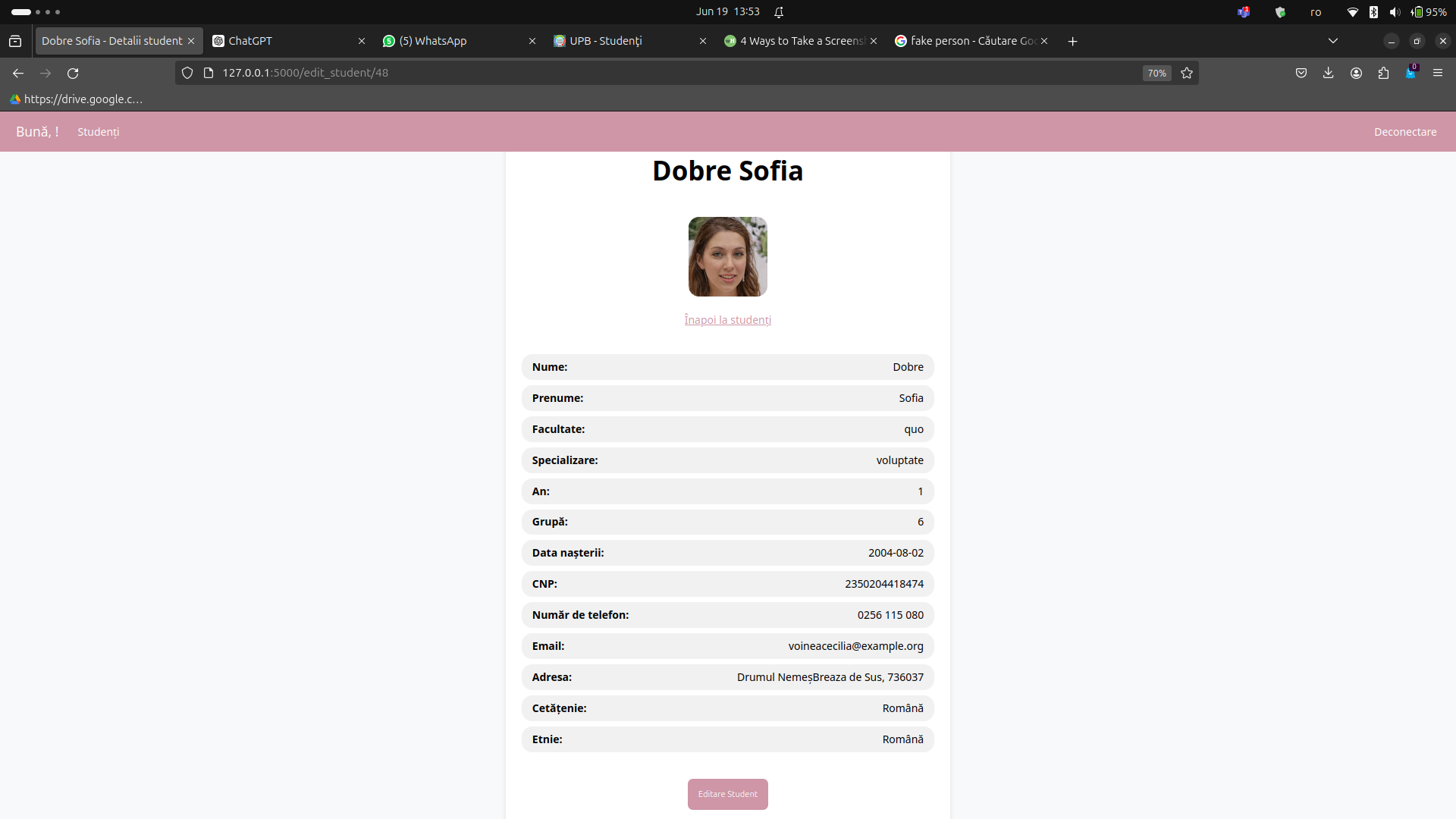
* + Utilizatorii pot sorta lista de studenți în funcție de două câmpuri: nume și prenume.
  + Sortarea poate fi efectuată în ordine crescătoare (asc) sau descrescătoare (desc).



Lista cu studenți afișează studenții după nume, prenume și CNP. În dreptul fiecărui student există posibilitatea de a vizualiza detalii student sau de a șterge. Pe o pagină se pot afișa doar 10 studenți, iar în cazul în care există mai mulți studenți, aceștia se vor situa pe pagini diferite. Paginile pot fi accesate cu ajutorul meniului de sub lista de studenți. Butonul prima redirecționează utilizatorul la prima pagină, ultima la ultima pagină. Butonul din mijloc va fi mereu pagina pe care ne aflăm, iar în stânga/dreapta vor fi pagina anterioară/următoare paginii accesate.



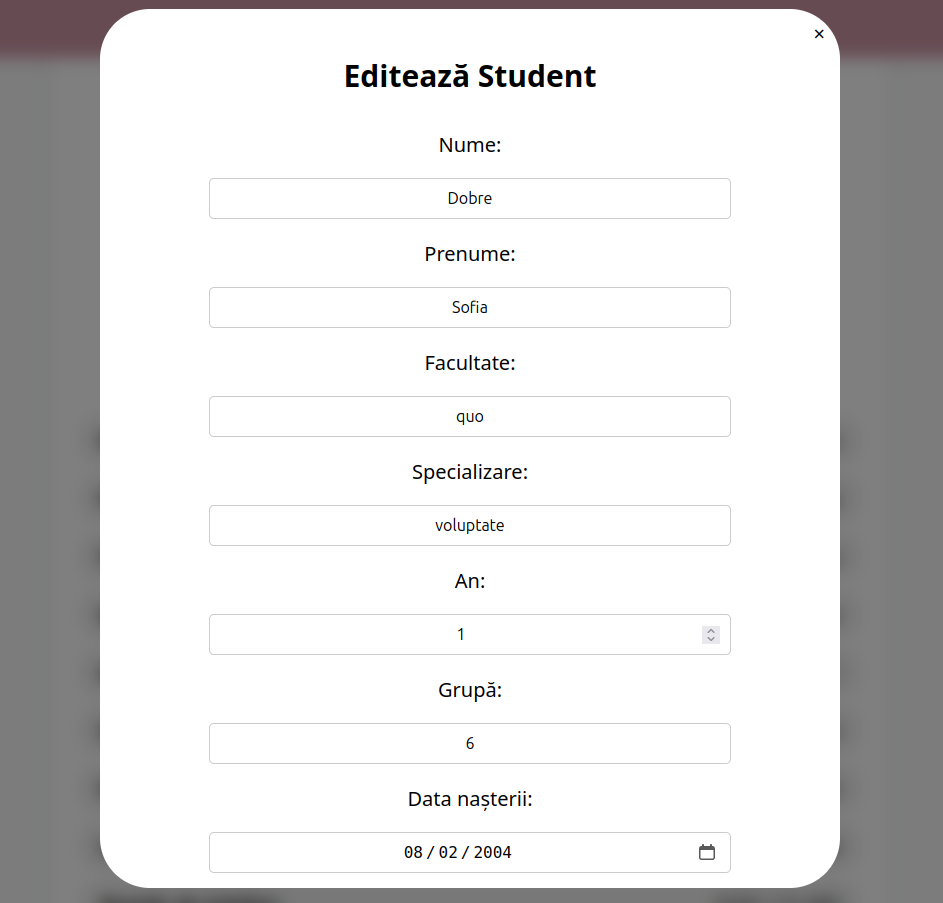
Pagina „Vizualizează student” afișează toate detaliile studentului ales.



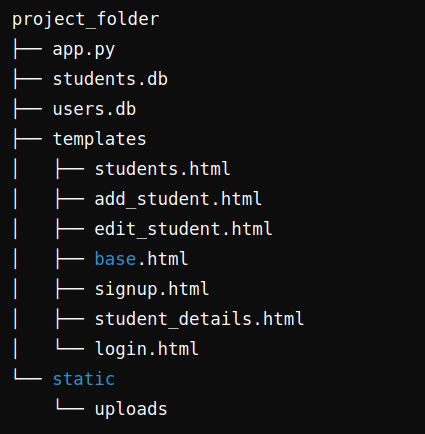
Bara de navigație s-a păstrat și ea, dar lipsește butonul de adăugare al elevilor.

Butonul „Înapoi la studenți” redirecționează utilizatorul la pagina principală.

Butonul „Editare student” permite editarea acestuia, indiferent de criteriu. Acest lucru se realizează în mod asemănător adăugării.



**Implementarea**

 Structură folder proiect

Fișierele HTML din folderul templates:

1. **students.html**:
   * Acest fișier este responsabil pentru afișarea unei liste de studenți sau a rezultatelor unei căutări de studenți.
   * Prin intermediul acestuia, utilizatorul poate naviga printre pagini de studenți, poate sorta și poate filtra rezultatele.
2. **add\_student.html**:
   * Aici utilizatorul poate introduce datele unui student nou.
   * Formularul validează datele introduse, asigurând că acestea respectă cerințele precum lungimea CNP-ului sau tipul de fișier al pozei de profil.
3. **edit\_student.html**:
   * Afișează un formular pre-populat cu datele unui student existent.
   * Permite utilizatorului să modifice detaliile studentului, inclusiv să încarce o imagine de profil nouă.
4. **base.html**:
   * Este șablonul de bază pentru toate paginile HTML ale aplicației tale.
   * Include antete, bare de navigație, sau alte elemente comune la toate paginile.
   * Asigură consistență și un aspect uniform al întregului site.
5. **signup.html**:
   * Acesta este formularul prin intermediul căruia utilizatorii pot crea conturi noi.
   * Solicită informații precum numele de utilizator, parola și adresa de email pentru a crea un nou cont în baza de date.
6. **student\_details.html**:
   * Afișează detalii complete despre un student specific.
   * Include informații detaliate despre nume, prenume, date de contact, detalii academice și poza de profil a studentului.
7. **login.html**:
   * Acesta este formularul folosit pentru autentificarea utilizatorilor.
   * Solicită numele de utilizator și parola pentru a permite accesul în contul utilizatorului.

Fișierul Python app.py:

* **app.py**:
  + Este fișierul principal al aplicației tale Flask.
  + Definește rutele (URL-urile) accesibile de către utilizatori și implementează logica necesară pentru fiecare rută.
  + Folosește SQLAlchemy pentru a interacționa cu bazele de date students.db și users.db, gestionând operațiuni precum adăugarea, modificarea și ștergerea datelor.
  + Implementează sesiuni pentru a gestiona starea autentificată a utilizatorilor în timpul sesiunilor de navigare.
  + Include funcții de validare a datelor, cum ar fi verificarea formatului email-ului sau validarea CNP-ului înainte de a salva datele în baza de date.
  + Utilizează decoratori Flask, cum ar fi @login\_required, pentru a restricționa accesul la anumite rute doar utilizatorilor autentificați.
  + Asigură funcționalități de încărcare a fișierelor de imagine pentru pozele de profil ale studenților în directorul static/uploads.

Descrierea tehnologiilor utilizate

Aplicația web a folosit mai multe limbaje de programare precum:

* **Python**: versiunea 3.11.7
* **HTML și Jinja2**:
* **CSS și Bootstrap**
* **JavaScript și jQuery**
* **SQL și SQLite3**

În pagina, Python Flask a fost utilizat pentru a gestiona și a livra funcționalitățile principale ale aplicației, cum ar fi gestionarea datelor despre studenți, autentificarea utilizatorilor și rutele principale ale aplicației. Modurile în care a fost folosit Python Flask:

1. **Routing și Gestionarea Cererilor HTTP**:
   * Flask utilizează decoratori Python pentru a mapa URL-uri la funcții Python, ceea ce permite definirea rutei și a logicii asociate cu aceasta.
   * De exemplu, rute pentru afișarea listei de studenți (/students), adăugarea unui nou student (/add\_student), editarea datelor unui student (/edit\_student/<id>), ștergerea unui student (/delete\_student/<id>), etc.
2. **Interacțiunea cu Baza de Date**:
   * S-a folosit Flask pentru a conecta aplicația la o bază de date SQLite, gestionând tabelele și interogările prin intermediul modulului sqlite3.
   * S-au definit schemele tabelelor (cum ar fi tabela students și users) și s-a utilizat SQL pentru a realiza operațiuni CRUD (Create, Read, Update, Delete) asupra acestora.
3. **Ședințe și Autentificare**:
   * Pentru a gestiona autentificarea utilizatorilor, s-au utilizat ședințe în Flask (session). Acest lucru permite stocarea și gestionarea stării de autentificare a utilizatorului pe parcursul sesiunii.
   * S-au implementat rute și logici asociate pentru autentificarea (/login), înregistrarea (/signup), verificarea stării sesiunii (/session\_status) și deconectarea utilizatorului (/logout).
4. **Manipularea Fișierelor și Încărcarea de Imagini**:
   * S-au implementat funcționalități pentru încărcarea și stocarea fișierelor (cum ar fi pozele de profil ale studenților) utilizând Flask.
   * Folosind configurarea pentru încărcarea fișierelor (UPLOAD\_FOLDER) și funcții precum secure\_filename, s-a asigurat o manipulare sigură a fișierelor încărcate de către utilizatori.
5. **Răspunsuri JSON și AJAX**:
   * Pentru a facilita interacțiunea asincronă cu aplicația, s-au implementat răspunsuri JSON pentru diverse acțiuni, cum ar fi adăugarea unui student sau editarea datelor acestuia.
   * S-au utilizat tehnologii precum AJAX și librăria jQuery pentru a efectua solicitări asincrone către server și pentru a gestiona răspunsurile.

JavaScript a fost utilizat pentru a realiza următoarele operații:

1. **Manipularea DOM și Evenimente**:
   * S-a folosit JavaScript pentru a manipula elementele DOM (cum ar fi formularul de adăugare a unui student sau de editare a datelor acestuia).
   * S-au gestionat evenimente precum click-ul pe butoanele de adăugare, editare sau ștergere, afișarea și ascunderea modalurilor (pop-up) pentru interacțiuni cu utilizatorul.
2. **Validarea Formularelor**:
   * S-au implementat validări în JavaScript pentru a asigura că datele introduse în formulare sunt corecte înainte de a fi trimise către server.
   * S-a verificat lungimea CNP-ului introdus, tipul de fișier pentru încărcarea imaginilor, tipul de dată pentru an.
3. **Comunicarea Asincronă cu Serverul (AJAX)**:
   * S-a utilizat AJAX pentru a efectua solicitări către server pentru diverse acțiuni, cum ar fi adăugarea unui student, editarea sau ștergerea acestuia, fără a reincărca pagina întreagă.
   * Prin intermediul AJAX, s-a actualizat dinamic conținutul paginii în funcție de acțiunile utilizatorului, cum ar fi afișarea listei de studenți după o căutare sau salvarea datelor modificare a unui student

**SQLite3 a fost folosit pentru a gestiona datele într-o bază de date relațională direct din Python.**

* ****Conectarea la baza de date:** S-a utilizat** sqlite3.connect() **pentru a stabili o conexiune la baza de date SQLite3. În acest caz, calea către fișierul bazei de date este (**STUDENT\_DB\_PATH **și** USERS\_DB\_PATH**).**
* ****Crearea sau inițializarea tabelelor**:** S-au definit funcții cum ar fi init\_db() și init\_users\_db() pentru a crea tabelele necesare în baza de date la începutul execuției aplicației sau la prima cerere (@app.before\_first\_request). Aceste funcții folosesc conn.execute() pentru a executa comenzi SQL de creare a tabelelor.
* **Interogarea și manipularea datelor**: S-au utilizat cursori (conn.cursor()) pentru a executa interogări SQL, cum ar fi SELECT, INSERT, UPDATE și DELETE.
* **Gestionarea tranzacțiilor și commit**: Pentru a asigura coerența datelor, s-au folosit conn.commit() pentru a efectua commit la tranzacțiile SQL, asigurându-te că modificările sunt salvate permanent în baza de date.
* **Închiderea conexiunii la baza de date**: După ce s-a terminat de lucrat cu baza de date, se folosește conn.close() pentru a închide conexiunea, eliberând resursele asociate.

Bootstrap a fost folosit astfel:

* **Integrarea în proiect**: S-au inclus fișierele CSS și JS ale framework-ului în structura proiectului.
* **Utilizarea grid-ului**: S-a folosit sistemul de grid Bootstrap pentru a structura layout-ul paginilor. Acest lucru include împărțirea paginii în rânduri (rows) și coloane (columns), permițând aranjarea flexibilă a elementelor pe ecran în funcție de nevoile de design.
* **Componente UI gata de utilizare**: Au fost integrate diverse componente UI din Bootstrap în site, cum ar fi butoanele stilizate, formularele cu validare integrată, meniurile responsive, alertele și altele. Aceste componente au redus timpul de dezvoltare și au asigurat un aspect uniform și profesional al site-ului.
* **Adaptabilitatea și responsivitatea**: Folosind clasele de responsivitate din Bootstrap, s-a asigurat faptul că site-ul este optimizat pentru afișare pe diferite dispozitive și dimensiuni de ecrane. Acest lucru include utilizarea clasei .container pentru a limita lățimea conținutului și clasele .col-\* pentru a controla dimensiunile coloanelor pe diferite break-points.

Și nu în ultimul rând am folosit jinja2 astfel:

1. **Integrarea cu Flask**: Jinja2 este integrat direct cu Flask, permițând definirea de șabloane HTML în fișiere separate și rendarea dinamică în rutele definite în Flask.
2. **Șabloane HTML**: S-a folosit sintaxa specifică Jinja2 pentru a include variabile din Python, a itera prin liste de date și a aplica condiții.
   * **Variabile**: Au fost incluse variabile din Python în HTML folosind duble acolade {{ }}. De exemplu, {{ student.nume }} ar fi putut fi utilizat pentru a afișa numele unui student într-un șablon HTML.
   * **Structuri de control**: S-au utilizat blocuri de control Jinja2 pentru a itera prin liste de studenți, cum ar fi {% for student in students %} ... {% endfor %} pentru a afișa o listă de studenți într-o tabelă HTML.
   * **Încorporarea de cod Python**: S-au folosit blocuri Jinja2 precum {% if condition %} ... {% endif %} pentru a aplica condiții în șabloanele HTML, cum ar fi afișarea diferitelor elemente în funcție de starea lor.

Descrierea mediului de lucru

Am optat să folosesc Linux datorită simplității mediului de lucru de instalare a tuturor package-urilor necesare pentru aplicația în Python Flask. Mediul de lucru pentru care am optat pentru dezvoltarea aplicației a fost Spyder3 care rulează versiunea 3.11 de Python. Am folosit Spyder și pentru scrierea modulelor HTML/CSS, acesta fiind compatibil și cu aceste limbaje.

Spyder (acronim pentru Scientific PYthon Development EnviRonment) este un mediu de dezvoltare integrat (IDE) open-source pentru programarea în limbajul Python. Spyder este proiectat special pentru lucrul cu analize de date și dezvoltare științifică în Python și este cunoscut pentru integrarea puternică cu librării populare din ecosistemul Python, cum ar fi NumPy, Pandas, Matplotlib, SciPy și altele.

### Caracteristici principale ale Spyder:

1. **Editor de cod avansat:** Spyder oferă un editor puternic de cod Python cu funcții precum evidențierea sintaxei, indentarea automată, completarea automată a codului și navigarea rapidă prin cod.
2. **Explorer de variabile:** Permite vizualizarea și explorarea variabilelor în timpul rulării codului, facilitând depanarea și analiza datelor.
3. **Consolă IPython integrată:** Spyder include o consolă IPython interactivă, care permite rularea de cod Python în timp real și interacțiune directă cu obiectele și datele într-un mod interactiv.
4. **Integrare cu librării științifice:** Spyder este optimizat pentru lucrul cu librării precum NumPy, SciPy, Pandas și Matplotlib, facilitând analiza și vizualizarea datelor.
5. **Gestionarea proiectelor:** Permite organizarea proiectelor Python într-un mod structurat, cu suport pentru gestionarea fișierelor, structuri de directoare și setări personalizate ale proiectului.
6. **Extensibilitate:** Spyder este extensibil prin intermediul plugin-urilor, permițând utilizatorilor să adauge funcționalități suplimentare și să personalizeze mediul de dezvoltare conform nevoilor lor.
7. **Interfață utilizator grafică (GUI):** Are o interfață intuitivă și prietenoasă pentru utilizator, care facilitează navigarea și utilizarea caracteristicilor avansate ale IDE-ului.

Conda3 a fost interpreter-ul codului. Acesta este un sistem de gestionare a pachetelor, un manager de medii de programare și un mediu de gestionare a mediilor de programare care combină pachete și dependențe de la orice sursă.

Concluzii

În cadrul proiectului, aplicația web implementată utilizează Flask pentru dezvoltarea sa. Structura principală a proiectului include un fișier central app.py, responsabil pentru gestionarea logicii aplicației. Acesta conține rutele principale și funcționalitățile necesare pentru operarea corectă a aplicației.

Bazele de date students.db și users.db sunt utilizate pentru stocarea datelor despre studenți și pentru gestiunea utilizatorilor, respectiv. Structura de bază a proiectului include și directoarele templates și static. În templates, sunt plasate toate fișierele HTML care definesc aspectul și interfața utilizator pentru diferitele funcționalități ale aplicației.

În ceea ce privește funcționalitățile implementate:

* **Autentificare și Înregistrare**: Utilizatorii pot crea noi conturi și se pot autentifica folosind sesiuni gestionate de Flask. Parolele sunt stocate în baza de date după criptarea lor cu generate\_password\_hash și verificate cu check\_password\_hash.
* **Gestionarea Studenților**: Proiectul permite adăugarea, editarea și ștergerea datelor despre studenți. Aceste operații sunt gestionate prin interfețe dedicate în cadrul aplicației web.
* **Interacțiunea cu Baza de Date**: Modulele sqlite3 sunt folosite pentru a realiza operațiunile CRUD (Create, Read, Update, Delete) asupra datelor stocate în bazele de date. Aceste operațiuni sunt esențiale pentru manipularea și afișarea corectă a informațiilor despre studenți.

Proiectul utilizează, de asemenea, tehnici de validare a datelor și de gestionare a fișierelor încărcate, toate acestea fiind integrate într-o interfață web prietenoasă utilizatorului, realizată prin intermediul HTML, CSS și Bootstrap.

Această structură și funcționalitate asigură o experiență coerentă și eficientă pentru utilizatorii aplicației, facilitând administrarea și gestionarea informațiilor legate de studenți într-un mod sigur și organizat.