Proiect Aplicație Web

Andronic Mihai-Gabriel

Cuprins

[1. Task 3](#_Toc204169248)

[1.1 Task principal 3](#_Toc204169249)

[1.2 Task secundar 4](#_Toc204169250)

[2.Login Register 5](#_Toc204169251)

[3.Meniul aplicației pentru fiecare rol 6](#_Toc204169252)

[4.Meniul de Admin 7](#_Toc204169253)

[4.1.Consum Total/mediu 7](#_Toc204169254)

[4.3 Alerte contor defect 8](#_Toc204169255)

[4.4 Afișare si Modificare a bazei de date 9](#_Toc204169256)

[4.5 Afișarea tabelului de useri 10](#_Toc204169257)

[4.6 Afișare informatii client 11](#_Toc204169258)

[5.Mutarea aplicației pe Azure 12](#_Toc204169259)

[6.Posibile adăugări 13](#_Toc204169260)

[7.Skill-uri dobândite 13](#_Toc204169261)

# 1. Task

## 1.1 Task principal

Crearea unei pagini WEB cu sistem de login/register în care să se folosească baza de date de la task-ul anterior. Pagina WEB trebuie să conțină:

* Diferite roluri de utilizatori
* Meniul aplicației
* Meniu principal pentru admin care să conțină butoane pentru
  + Consum total / mediu
  + Top 5 clienți cu cel mai mare consum
  + Alerte de contor defect
  + Afișarea si modificarea tabelului de consum
  + Afișarea si modificarea rolului utilizatorilor
  + Afișarea unui anumit client în funcție de ClientID

De asemnea aplicația WEB trebuie sa ruleze pe cloud pe Azure la fel și baza de date.

## 1.2 Task secundar

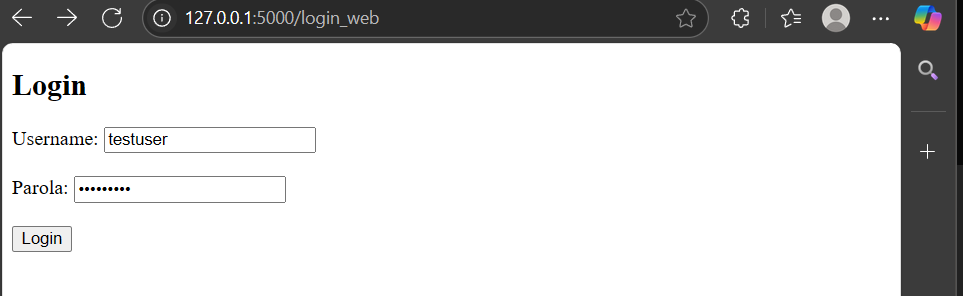
Prelucarea unui fișier CSV de tipul următor, în python astfel încât să îndeplinească cerințele:

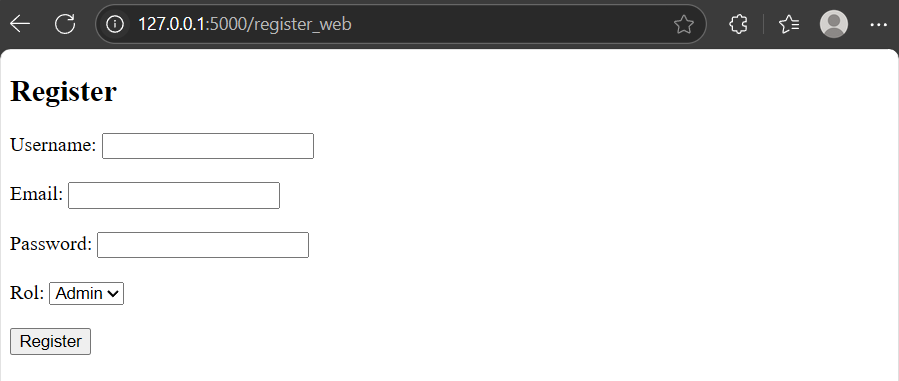
|  |  |
| --- | --- |
| **Coloana** | **Descriere** |
| ClientID | Cod client (poate fi lipsă sau nevalid, ex. "NA") |
| DataCitire | Dată în format mixt (e.g. 2025-05-03 12:15, 03-05-2025, etc.) |
| Consum\_kWh | Consum energetic (poate fi numeric, lipsă sau valori eronate) |
| TipClient | Tipul clientului (Casnic, Industrial) |
| Locatie | Orașul de consum (cu unele inconsistențe, ex. Bucuresti, București) |
| StatusContor | Starea contorului (OK, ok, Defect, DEFECT) |

* Citirea fișierului CSV
* Curățarea acestuia
  + Normalizarea formatelor de dată
  + Conversia la float a valorilor Consum\_kWh
  + Eliminarea / raportarea valorilor invalide
  + Uniformizarea orașelor si a statusurilor (title-case,etc.)
  + Tratarea valorilor lipsă
* Calcularea
  + Consumului mediu pe client / zi
  + Consumul total pe lună
  + Distribuția consumului in funcție de TipClient
* Salvarea datelor într-o bază de date (SQLite/MySQL)

# 2.Login Register

Am creat un sistem simplu de login/register. Tabul de Register salveaza inregistrările într-un tabel din baza de date cu parolele criptate. Tabul de Login verifică daca datele introduse se regăsesc pe vreo linie a tabelului dupa decriptarea parolei.După finalizarea login-ului aplicația te va redirecționa într-un dashboard în funcție de rolul atribuit.





Pașii pe care i-am urmat:

Am folosit framework-ul web principal Flask

Am creat două fișiere HTML simple:

* login.html permite utilizatorului să introducă nume de utilizator și parolă
* register.html include formularul de creare cont cu:
  + username, email, password, rol (admin/user)

Am folosit template-uri Jinja ({{ error }} / {{ message }}) pentru a afișa mesaje dinamice în caz de erori sau succes.

Pentru partea de register:

* Se primește formularul completat prin metoda POST
* Se verifică dacă username și email există deja în baza de date userweb
* Parola este criptată folosind Flask-Bcrypt:
* Se salvează datele în baza de date cu rolul ales (user / admin)
* Se redirecționează utilizatorul spre dashboard-ul principal

Pentru partea de login:

* Se interoghează baza de date pentru utilizatorul introdus
* Se verifică parola cu bcrypt.check\_password\_hash()
* Se creează o sesiune de autentificare cu login\_user() din Flask-Login
* În funcție de rolul din cont, utilizatorul este redirecționat spre dashboard-ul corect:
  + /dashboard pentru admin
  + /dashboard\_user pentru user

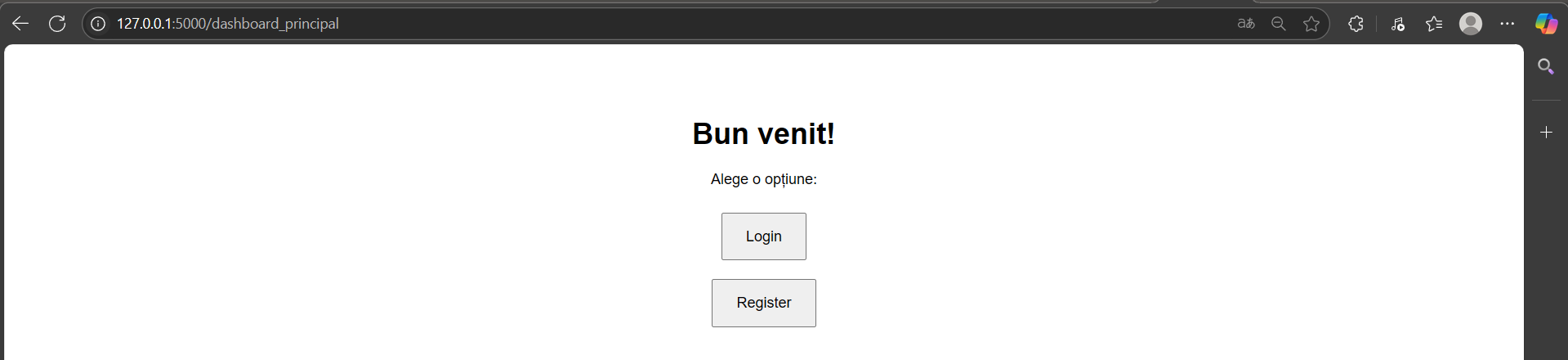
# 

# 3.Meniul aplicației pentru fiecare rol

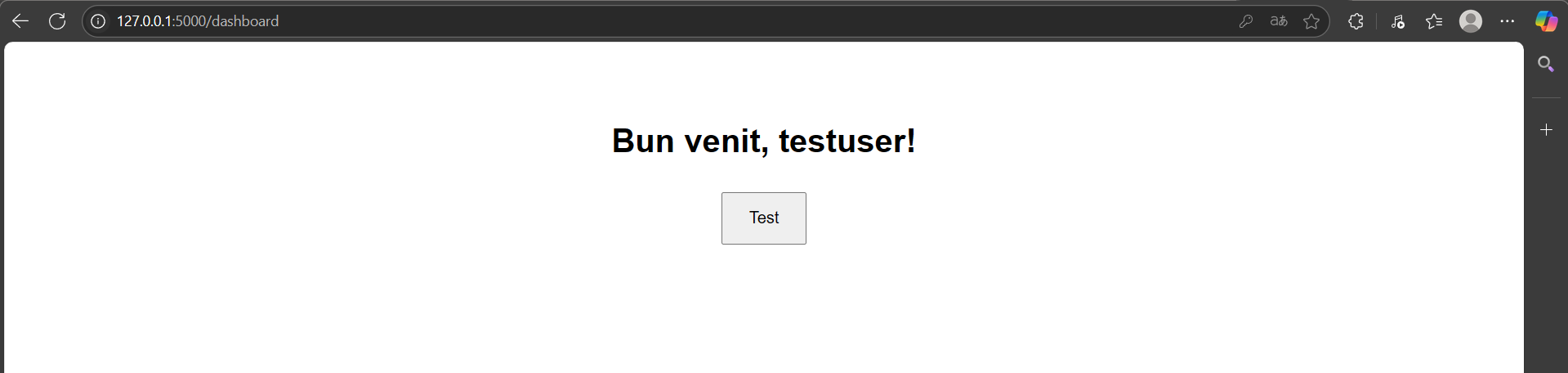
Aplicația oferă interfețe diferite pentru utilizatori, în funcție de rolul atribuit la înregistrare (user sau admin). După autentificare, utilizatorii sunt redirecționați spre dashboard-ul specific rolului lor. În ruta „/dashboard” am folosit un if care verifică rolul utilizatorului conectat și îl redirecționează în funcție de acesta.

Meniul Principal (dashboard\_principal.html)

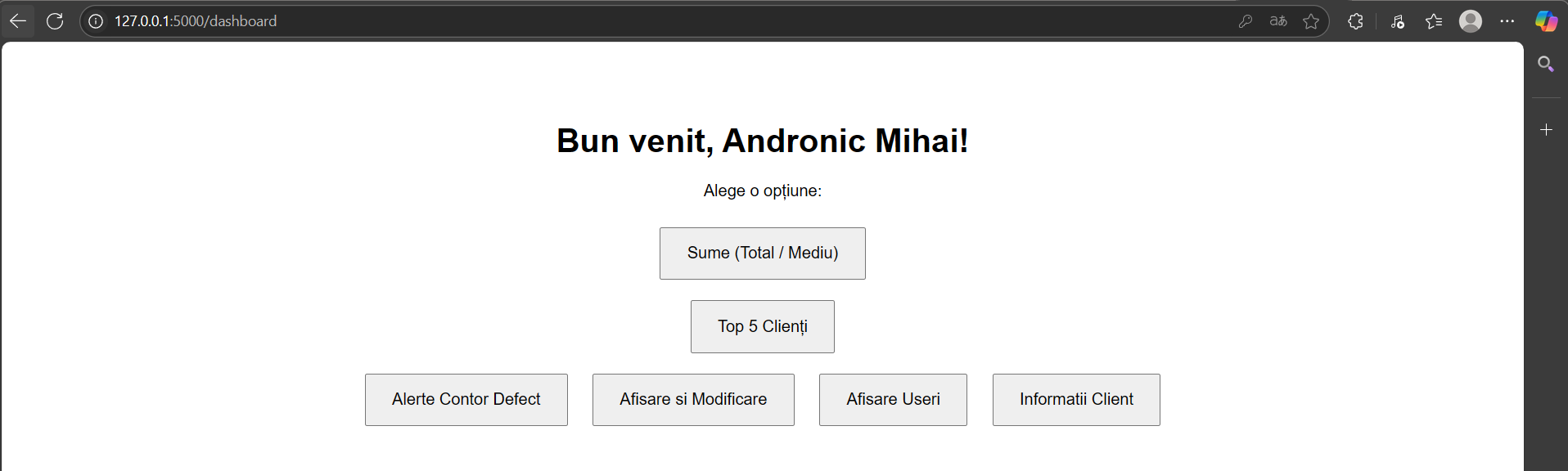
* Este ecranul de pornire care oferă acces către:
  + Pagina de login (/login\_web)
  + Pagina de înregistrare (/register\_web)



Meniul de User (încă în construcții) care conține doar un buton „Test”



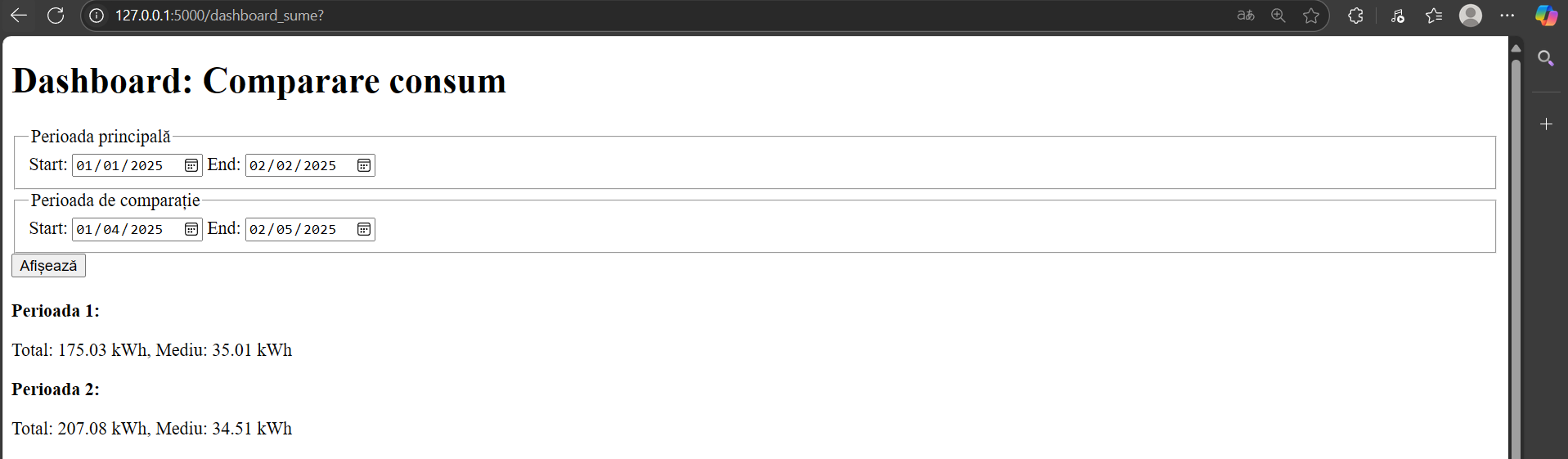
Meniul de Admin care conține 6 butoane denumite: Sume(Total/mediu),Top 5 Clienți, Alerte Contor Defect, Afișare și Modificare, Afișare Useri și Informații Client

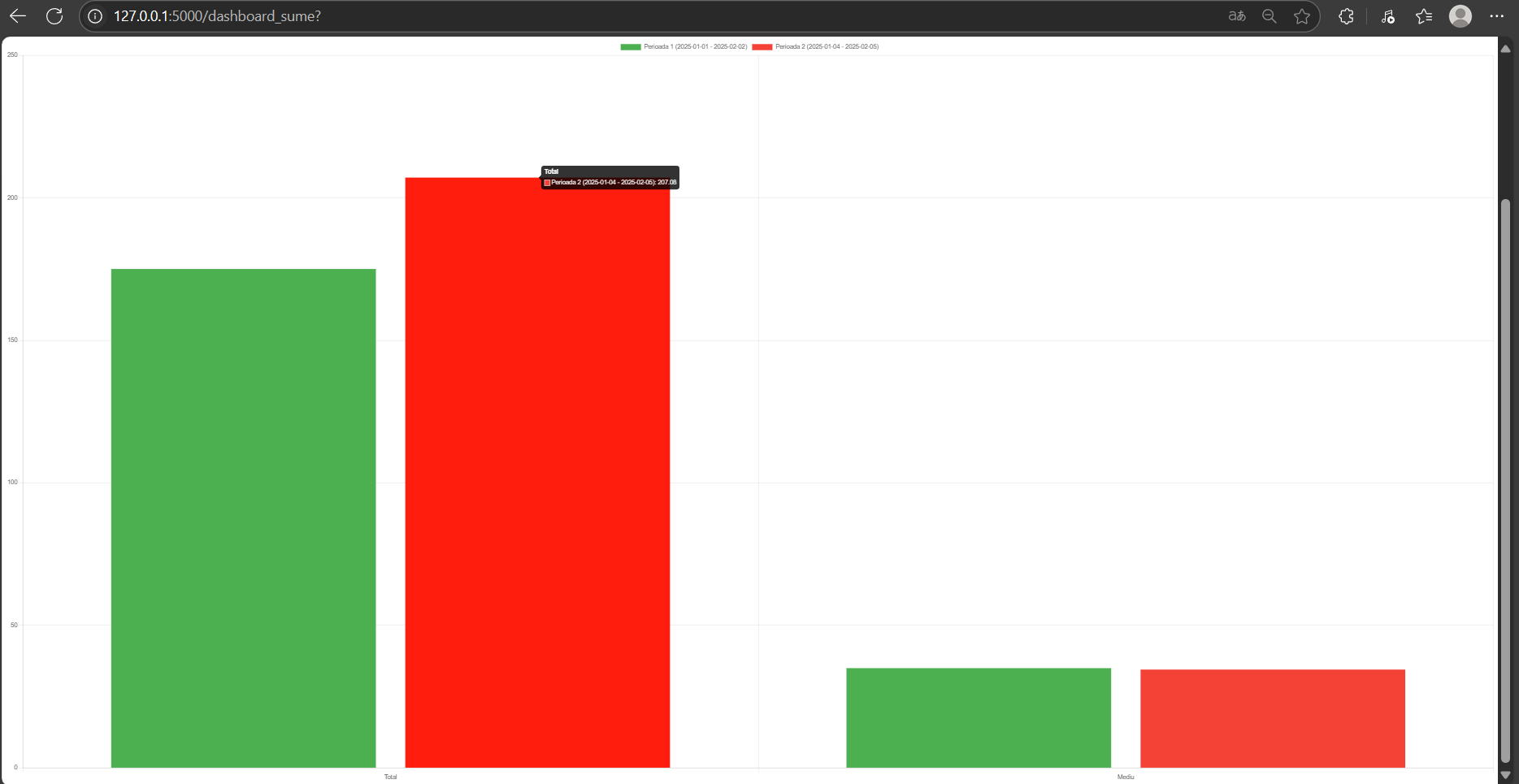


# 4.Meniul de Admin

## 4.1. Consum Total/mediu

Butonul de Sume(Total/Mediu) după ce este apăsat te redirecționează pe o pagină în care poți afla consumul total al unei perioade. După selectarea perioadei dorite și apăsarea butonului de „Afișează” se va afișa consumul total al și consumul mediu al perioadei. De asemenea se pot selecta 2 perioade în cazul în care se dorește compararea acestora. Totodată, se vor afișa si 2 grafice interactive, unul pentru consumul total și unul pentru consumul mediu, pentru o comparație mai bună.

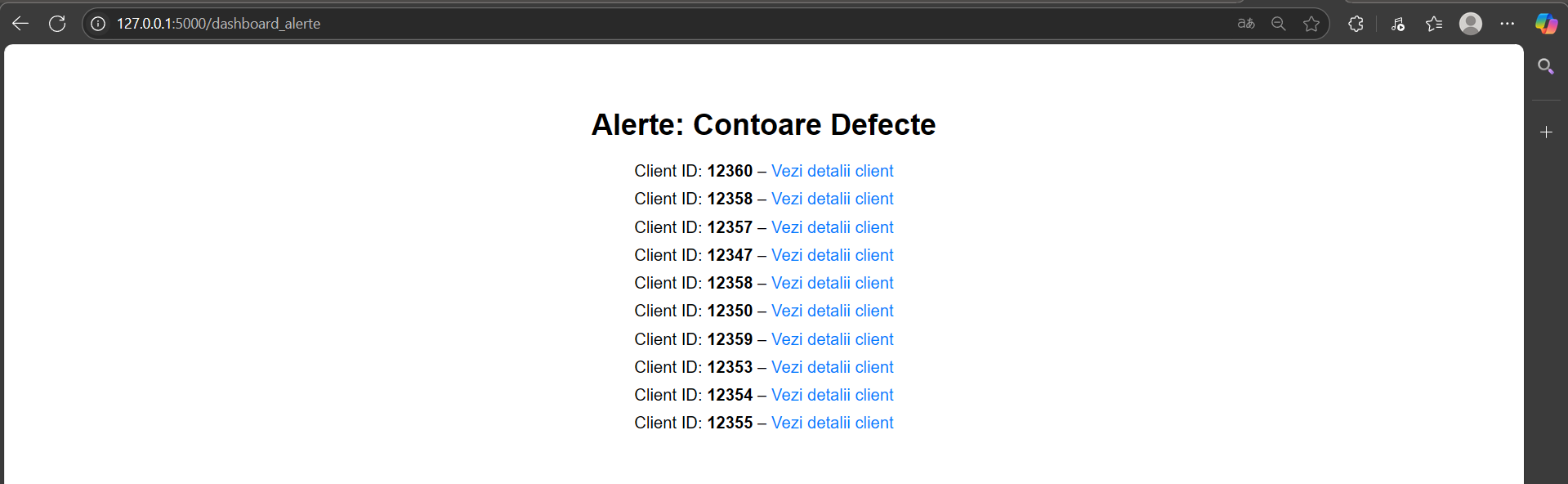


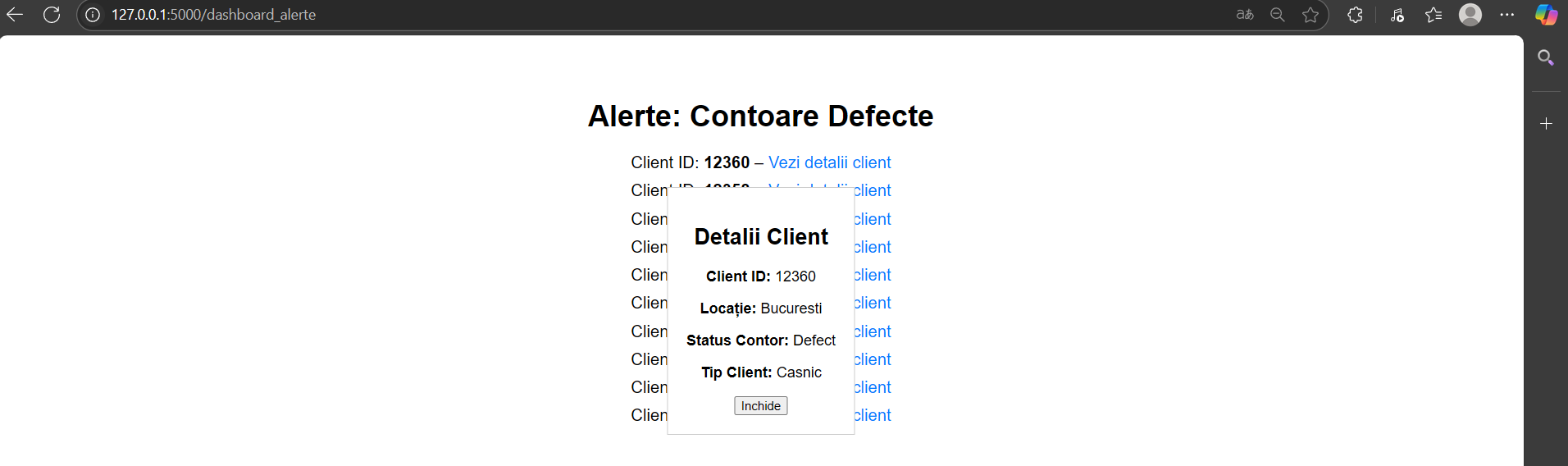


Am făcut o cerere de tip GET către endpoint-ul care se ocupa cu luarea datelor din tabelul din baza de date,calcularea consumului total și mediu între perioada selectată si care trimitea înapoi aceste 2 sume pe care le am adăugat in conținutul HTML .De asemnea am folosit Chart.js pentru desenarea unui grafic de tip bar.

## 4.3 Alerte contor defect

În acest tab se vor afișa toți clienții ai cărora contoare furnizează ca „defect”. De asemenea avem posibilitate sa apăsam pe ,,Vezi detalii client” pentru a deschide o mini-fereastră cu informații suplimentare despre clientul respectiv.

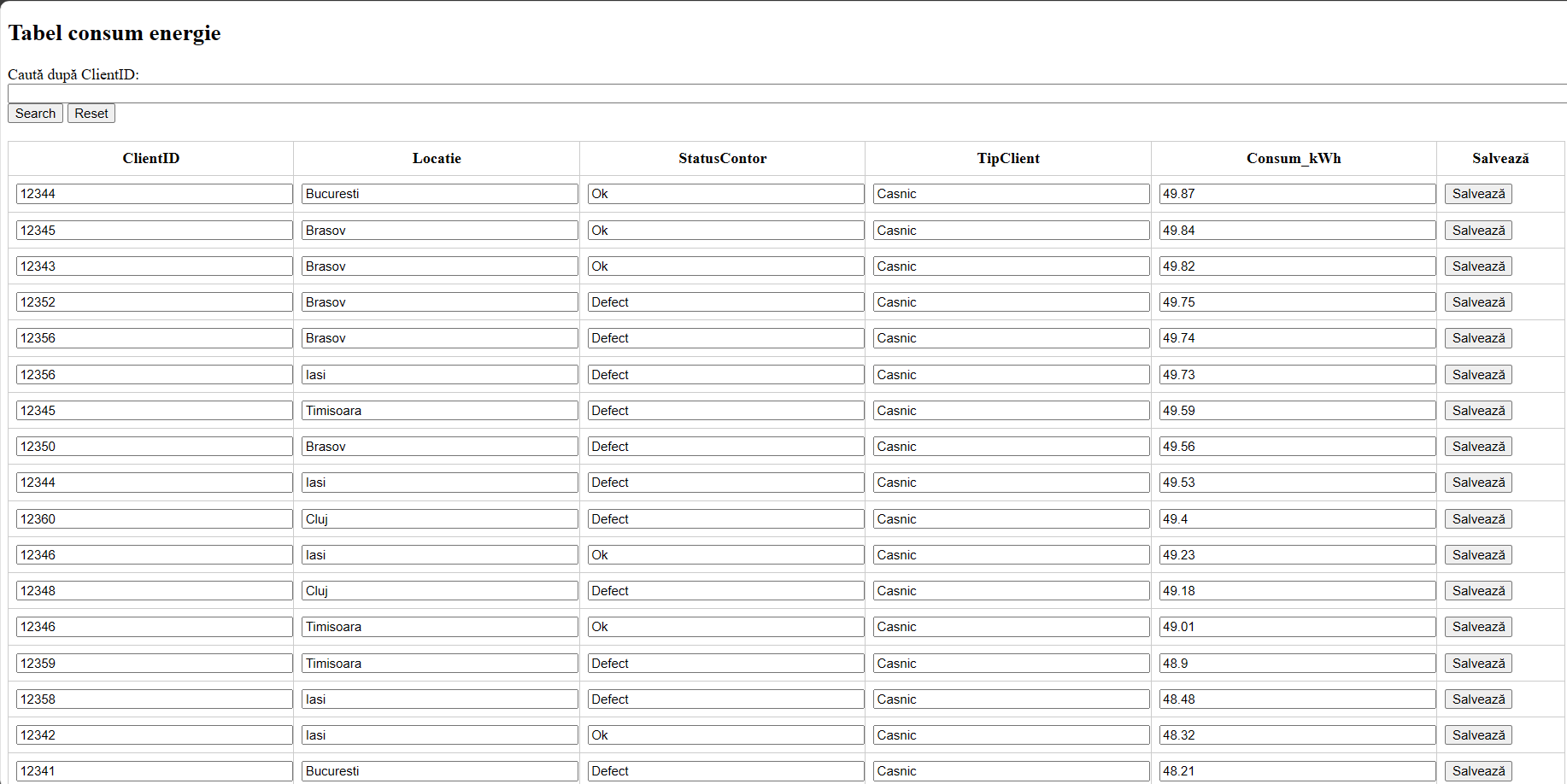




Pe partea de backend am folosit un subquerry care returneaza lista clienților cu contorul defect pana la ultima dată în care s-a citit un contor.Iar partea de frontend am folosit puțin JavaScript pentru preluarea clienților care aveau ca statusul contorului „defect” si listarea lor și a ferestrei modale pentru afișarea detaliilor

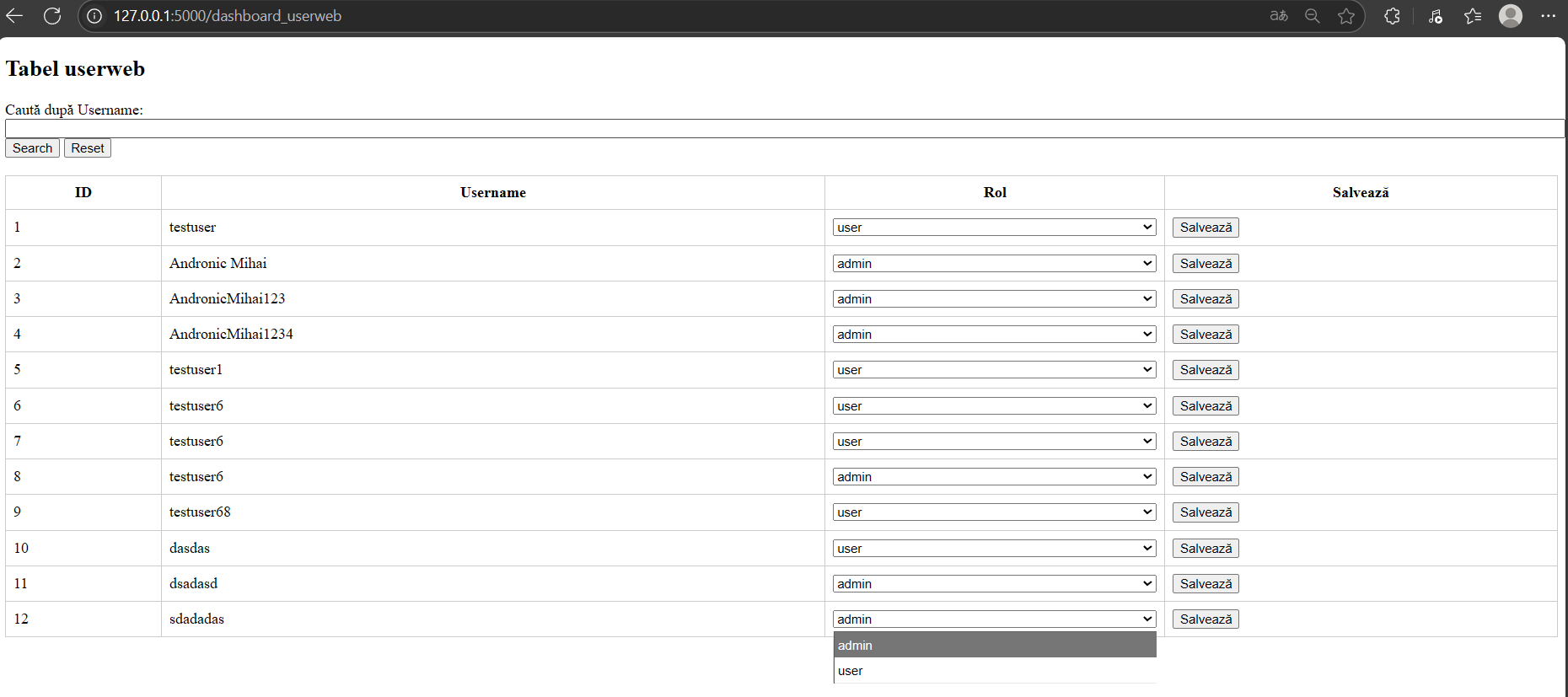
## 4.4 Afișare si Modificare a bazei de date

În acest tab se va afișa tot tabelul referitor la consumul clienților . Avem de asemenea posibilitatea să căutam pe cineva anume după ClientID. Afișarea acestui tabel are ca scop corectarea unor valori cum ar fi din contor „Defect” în contor „Ok” sau a Locației etc. . După modificarea valorilor se va apăsa pe butonul „Salvează” aferent liniei valorii modificate.Pe partea de backend în intercțiunea cu api am făcut doua cereri de timp GET una care returnează toate inregistrările din tabel si una pentru salvarea modificărilor în format JSON. Pe partea de JavaScript am folosit un loadTable(clientID) care atrage datele din api și populeaza tabelul. De asemnea am folosit si un saveRAW(clientID) care colecteaza datele modificate si trimite o cerere de tip PUT către api pentru actualizare.



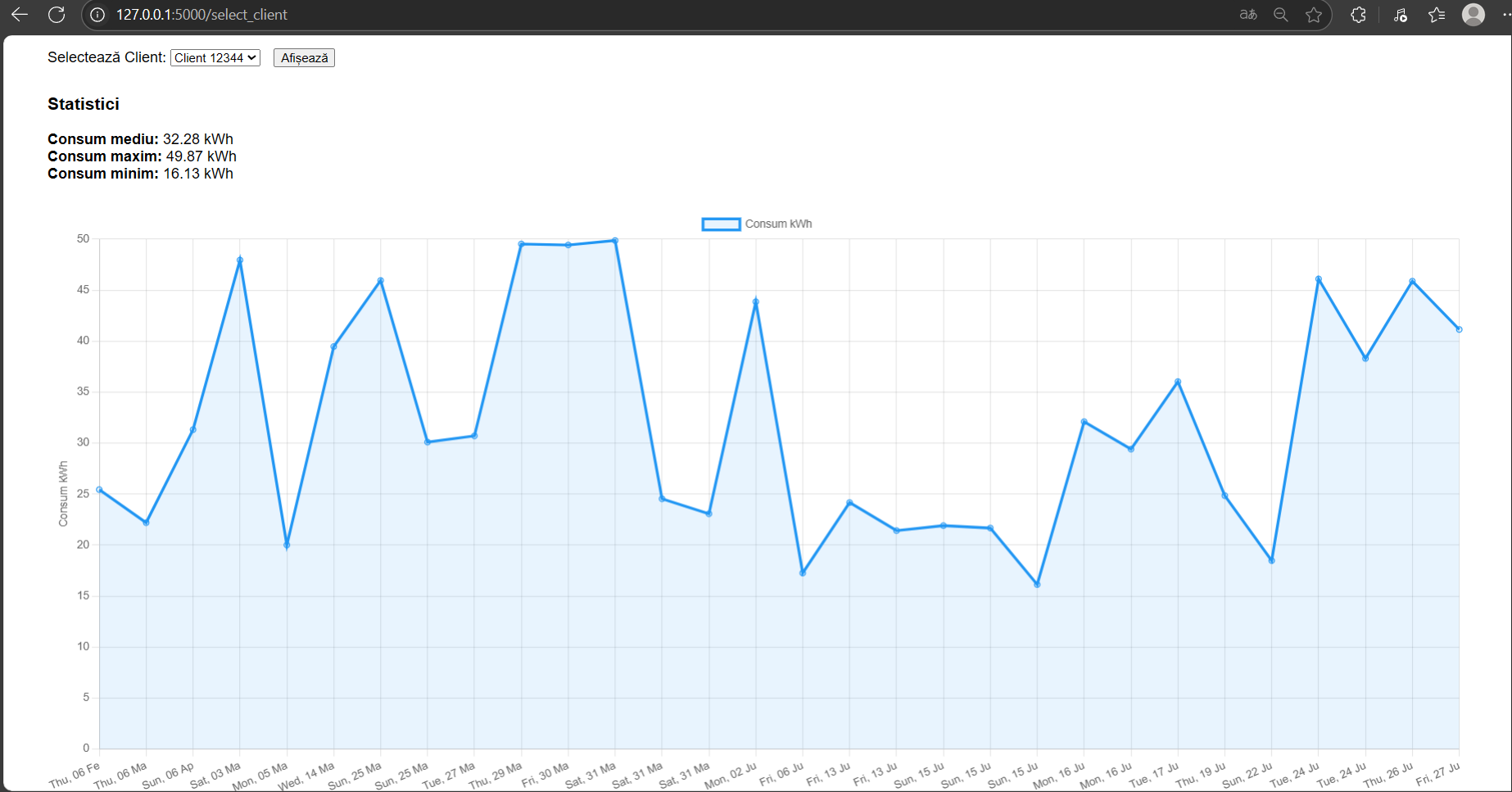
## 4.5 Afișarea tabelului de useri

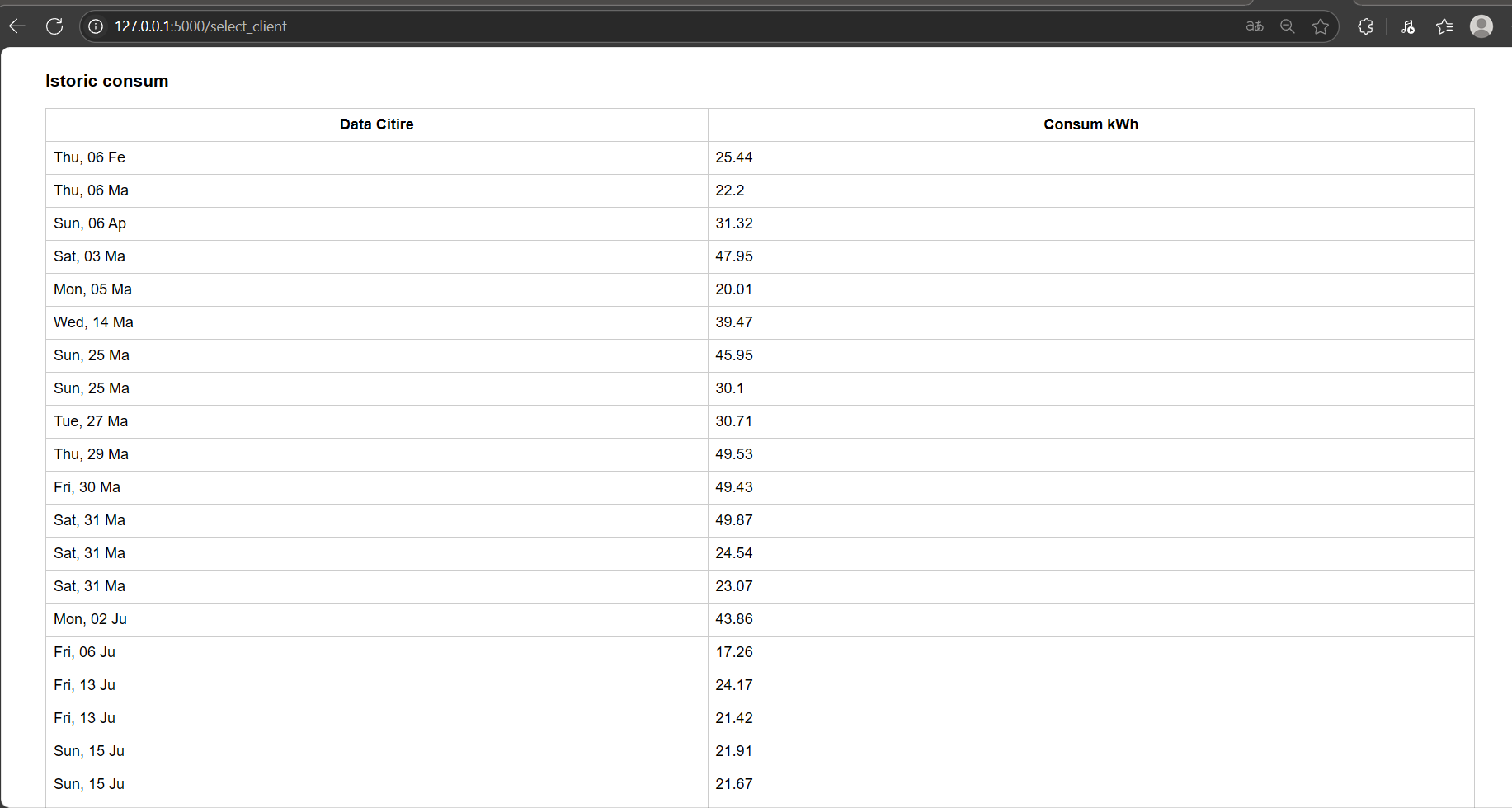
Ca și în tabul precedent, ni se va afișa un tabel dar de data aceasta cu useri. Avem posibilitatea de a schimba rolul fiecărui user din Admin în User și invers. După modificări trebuie apăsat butonul de „Salvează ” corespunzator liniei modificate. Am folosit funcția loadTable(username) în care daca este introdus un username, aceasta adaugă parametrul de căutare in URL și creeaza rânduri în tabel cu datele utilizatorilor si dropdown-ri pentru rol. De asemnea am folosit și funcția saveRow(id) care preia rolul selctat din dropdown si trimite o cerere PUT cu rolul nou.Pe partea de backend avem un GET /api/userweb?username=X, și un PUT /api/userweb/<id> care actualizează coloana rol din tabela userweb



## 4.6 Afișare informatii client

În cazul în care dorim mai multe informații referitoare la consumul unui anumit client, vom intra pe tabul de „Informații Client” unde, putem selecta id-ul clientului care ne interesează și ne va apărea un grafic despre consumul de kWh/dată, consumul minim,maxim si mediu dar și un tabel cu fiecare dată în care a fost citit controrul si valoarea indicată de acesta.Am folosit un dropdown care este populat cu toți clienții dupa id si care dupa apăsarea butonului „Afișează”, se declanșează functia loadClientInfo(). Ca și la Consumul total/mediu am folosit chart.js pentru grafice.





# 5.Deploy-ul aplicației pe Azure

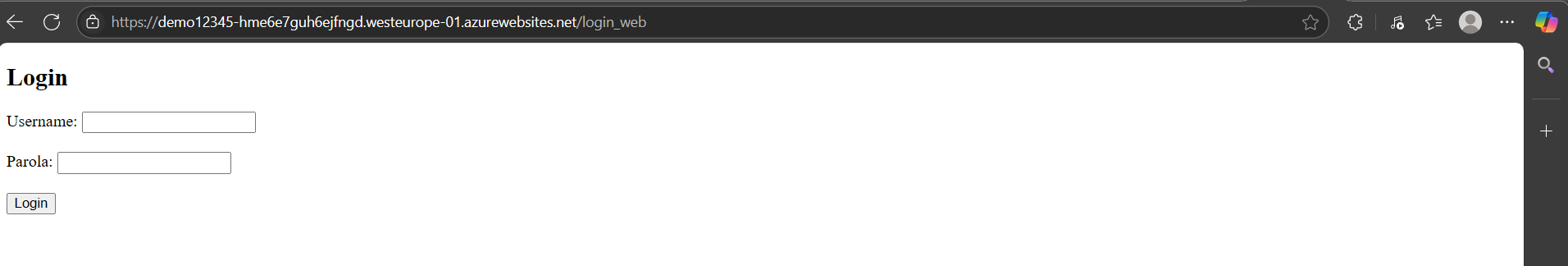
Pentru publicarea aplicației în mediu cloud, mi s-a oferit un **resource group** dedicat în Azure, denumit RG-Practica-Sandbox, în cadrul căruia am realizat toate configurările necesare pentru lansarea aplicației. Aici am creat și selectat instanța de **Web App** destinată site-ului meu.

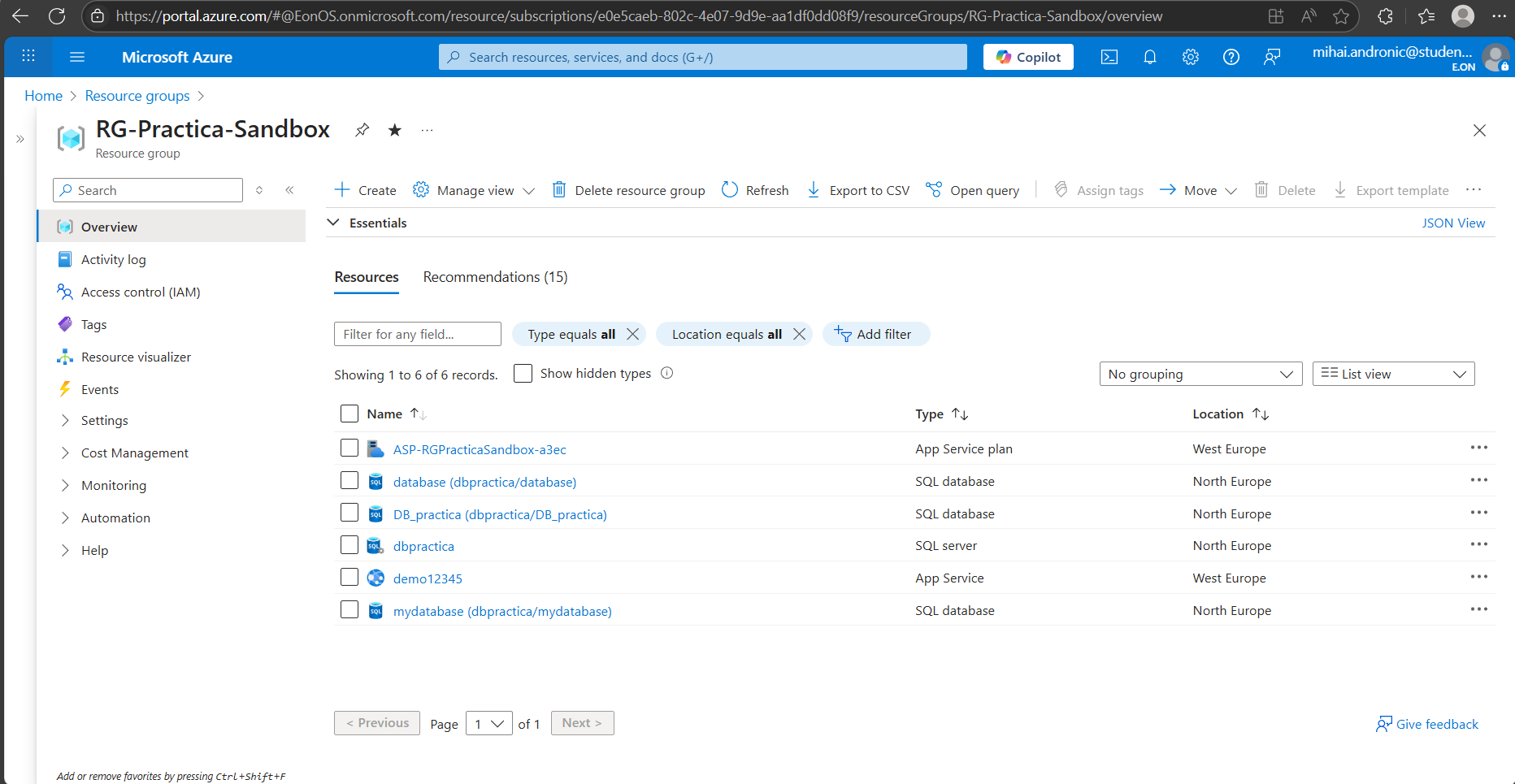
Primul pas tehnic a fost migrarea bazei de date din MySQL Workbench către serviciul **Azure SQL Database**. Am recreat structura tabelelor și le-am populat cu datele existente, păstrând integritatea informațiilor.

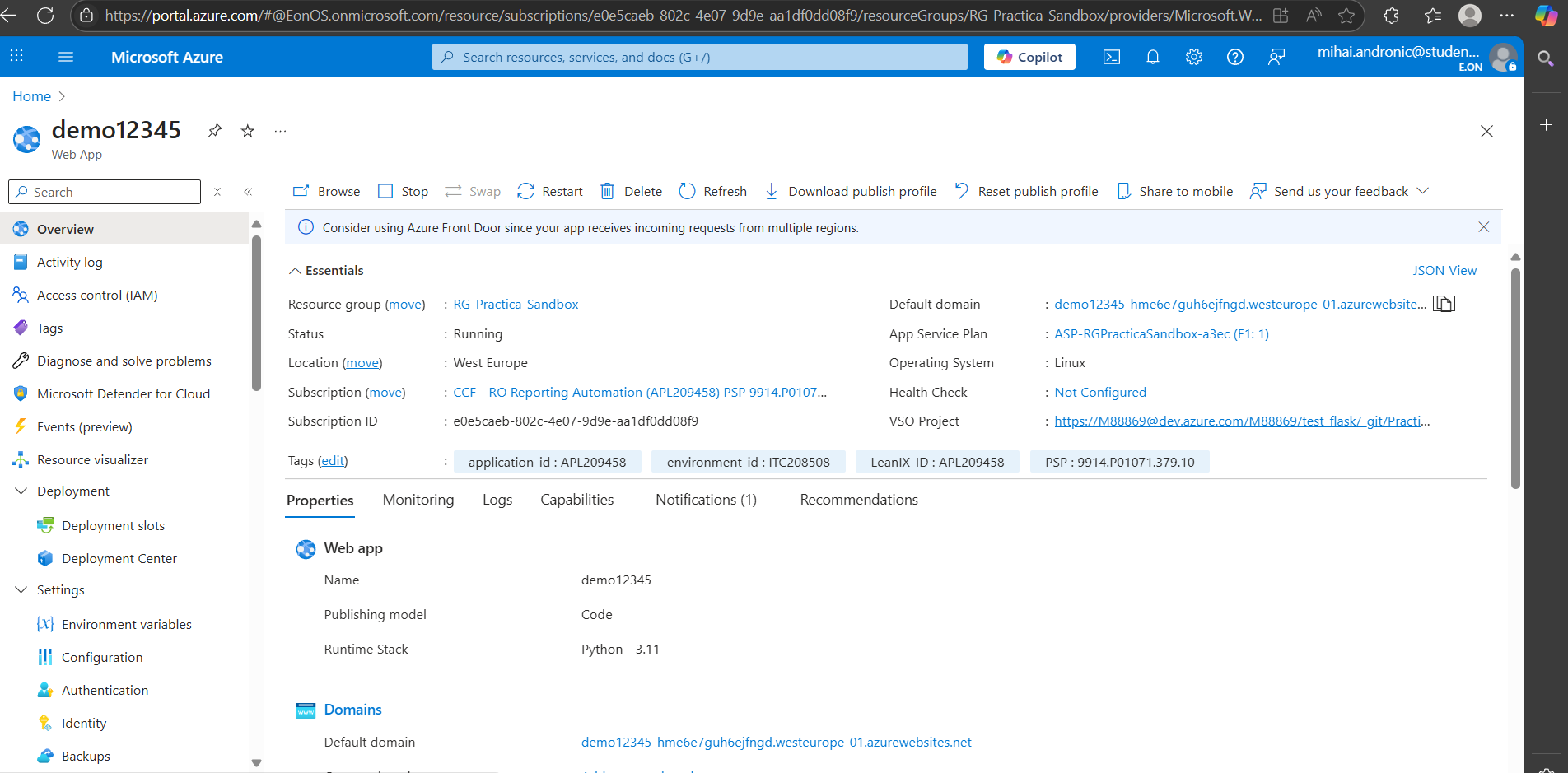
Pentru a permite conectarea aplicației la noua bază de date, am modificat parametrii corespunzători din fișierele database.py și .env. A urmat o etapă de ajustări în sintaxa HTML și JavaScript, pentru compatibilitate deplină în mediul Azure.

Codul sursă al aplicației a fost urcat într-un repository nou creat pe GitHub. În final, în cadrul **Deployment Center** din Web App-ul configurat, am conectat contul GitHub și am lansat procesul de deploy direct din repository.

După acești pași, aplicația a fost publicată cu succes și funcționează conform cerințelor proiectului.







# 6.Posibile adăugări

# Aplicația, deși funcțională, nu dispune momentan de o interfață grafică modernă care să ofere o experiență plăcută și intuitivă utilizatorilor. Designul este minimal, lipsit de stilizare vizuală. O posibilă extindere ar fi integrarea unui framework CSS precum Bootstrap, care ar îmbunătăți aspectul vizual și ar permite organizarea elementelor într-un mod mai ergonomic și atractiv.

# 7.Skill-uri dobândite

Pe parcursul dezvoltării acestui proiect, am dobândit o serie de competențe tehnice esențiale. Deși cunoșteam doar bazele limbajului Python la început, am învățat să construiesc o aplicație web complet funcțională folosind framework-ul Flask. Am aprofundat lucrul cu baze de date relaționale, în special SQL, reușind să structurez și să interoghez date eficient. Totodată, am explorat partea de securitate (criptarea parolelor, gestionarea sesiunilor) și am integrat mecanisme de autentificare și autorizare. O altă etapă importantă a fost lucrul în mediu cloud, unde am învățat să folosesc Azure SQL Database și să public aplicația prin Azure Web App. Am parcurs pași concreți în configurarea conexiunii, sincronizarea cu GitHub și realizarea procesului de deploy. Aceste competențe, dobândite în mod practic, mi-au permis să trec de la un nivel de bază la o aplicare reală a tehnologiilor moderne.