

**Nootch**

**Social Media App**

**Colegiul Național “Gheorghe Șincai”**

Dezvoltat de

**Mark-Theodor Erena & Vlad-Mihai Gură XII C**

Profesor coordonator  
**Rodica Smîntînă**

Aprilie 2024



**Prezentare generală**

Proiectul “Nootch” este o aplicație web și reprezintă o aplicație de socializare, similar cu instagramul, în care diferite persoane se pot înregistra și conecta pentru a putea vizualiza diferite postări.

Utilitatea unei aplicații de socializare este vastă și diversificată. De la facilitarea partajării momentelor de viață la construirea conexiunilor interpersonale și promovarea creativității, ea influențează profund modul în care ne conectăm și interacționăm în mediul online. Prin intermediul acestei platforme, oamenii pot exprima identitatea lor, pot explora lumea din jurul lor și pot construi comunități virtuale interconectate.

**Conținutul proiectului**

În realizarea acestui proiect au fost utilizate limbajele HTML, pentru a crea pagini web cu conținut, CSS, pentru a adaugă un design îmbunătățit paginii, JavaScript, pentru a verifica și a transmite date către “backend-ul” aplicației, și Java pentru a interacționa atât cu baza de date, cât și cu paginile HTML.

**Resurse hardware si software**

Proiectul a fost creat folosind mediul de programare de java IntelliJ versiunea 2023.3.3 și, pentru includerea tuturor bibliotecilor auxiliare, a fost folosit Apache Maven versiunea 3.9.6. Baza de date este ținută într-un server Amazon AWS Free Tier pe o instanță de linux cu Debian 12 în care este instalata Apache2 cu o instanță de phpmyadmin, adică o baza de date prin care se interacționează cu SQL. Pentru pozele folosite pentru grafică a fost utilizată funcția de generare de imagini de către ChatGPT.

**Cum funcționează proiectul?**

Proiectul web este împărțit în 2 părți: “frontend” și “backend”. Partea de frontend este partea cu care interacționează utilizatorul, adică paginile web scrise în HTML, CSS și JavaScript. Partea de backend facilitează conexiunea cu baza de date și interacțiunile cu paginile web. Pentru a avea un proiect cât mai organizat și eficient, a fost folosit Spring Boot configurat cu un link JDBC către baza de date.

**Ce este Spring Boot?**

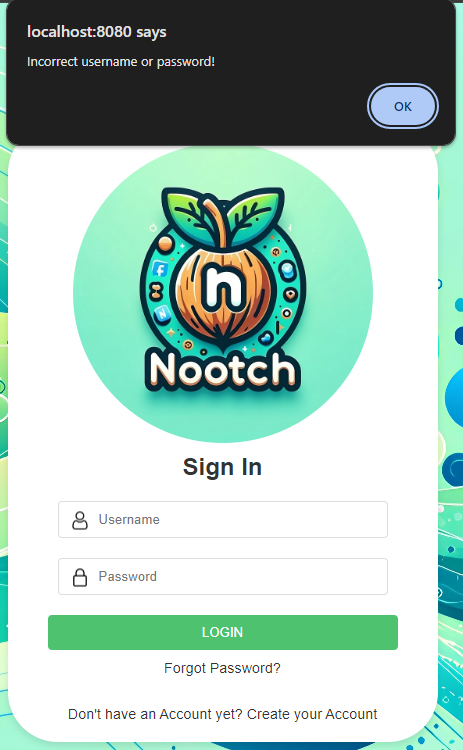
Spring Boot servește ca o platformă de dezvoltare care încurajează principiul "convenții peste configurație" și promovează dezvoltarea rapidă a aplicațiilor Java. Este proiectat pentru a elimina complexitatea configurării manuale a aplicațiilor, oferind în schimb o serie de convenții și valori implicite care permit dezvoltatorilor să-și înceapă proiectele cu ușurință. Printre principalele sale roluri se numără:

1. Auto-configurare: Un rol important al Spring Boot este acela de a facilita auto-configurarea. Detectează automat mediul de rulare și configurează aplicația în funcție de acesta, reducând astfel sarcina dezvoltatorului și eliminând nevoia de a configura manual componente comune, cum ar fi bazele de date sau serverele web.
2. Incorporare a Serverului: Spring Boot integrează un server web în aplicație, permițând dezvoltatorilor să creeze aplicații auto-suficiente, care pot fi pornite și rulate independent de un server extern. Aceasta oferă o abordare simplificată și portabilă pentru implementarea și distribuirea aplicațiilor Java.

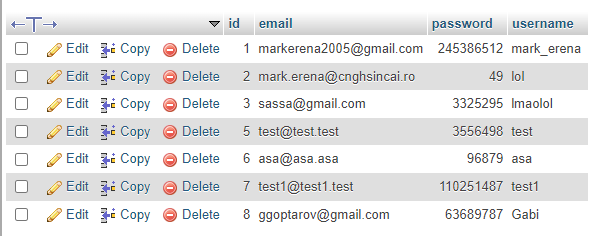
Dezvoltarea aplicatiei începe prin adăugarea dependențelor necesare în fișierul de configurare Maven al proiectului Spring Boot. Aceste dependențe includ bibliotecile și cadrele necesare pentru funcționalitatea dorită, cum ar fi Spring MVC pentru dezvoltarea web sau Spring Data pentru accesul la bazele de date. Spring Boot utilizează annotări specifice Spring pentru a marca componentele aplicației și pentru a controla comportamentul lor. De exemplu, o clasă care trebuie să fie gestionată de Spring IoC Container este marcată cu *@Component*, iar o clasă care trebuie să servească ca un controller web este marcată cu *@Controller*.

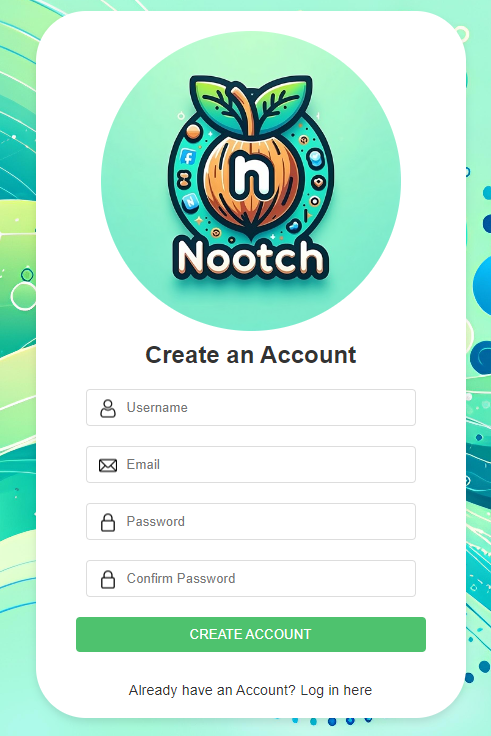
Pornirea Aplicației: Odată ce aplicația este dezvoltată și configurată, aceasta poate fi pornită folosind un container Servlet integrat (in cazul nostru, Tomcat) sau poate fi împachetată sub formă de fișier JAR executabil, care poate fi rulat independent de un server web extern.

**Interacțiunea cu baza de date**

În baza de date din serverul AWS, avem un tabel de utilizatori cu 4 coloane: id, email, username, password (reținută ca hash pentru a proteja utilizatorii).

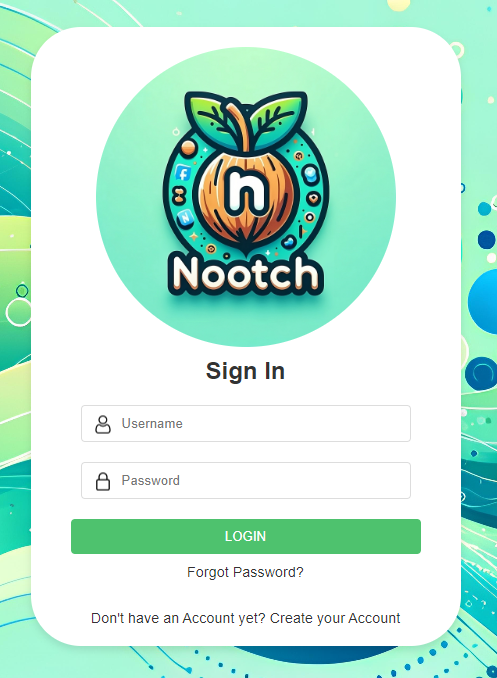
Atunci când își creează un cont sau se conectează la contul său, utlizatorul interacționează cu acest tabel, creând o linie nouă sau verificând dacă există deja în tabel.

Aici avem un exemplu în care un utilizator încearcă să se logheze cu un username care nu exista în baza de date:

**Descrierea datelor**

În primul rând, un utilizator nou este obligat să își creeze un cont nou pentru a putea accesa pagina principală a aplicației. Utilizatorul trebuie să își aleagă un username, să își introducă mailul (pentru a nu putea crea conturi de tip spam), și să își aleagă o parolă.

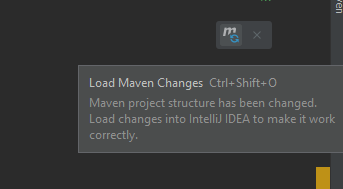
Când aceasta apasă pe butonul “create account”, datele sale sunt transmise către baza de date și un rând nou este creat în tabel, dacă email-ul și username-ul nu sunt folosite deja de alt utilizator.



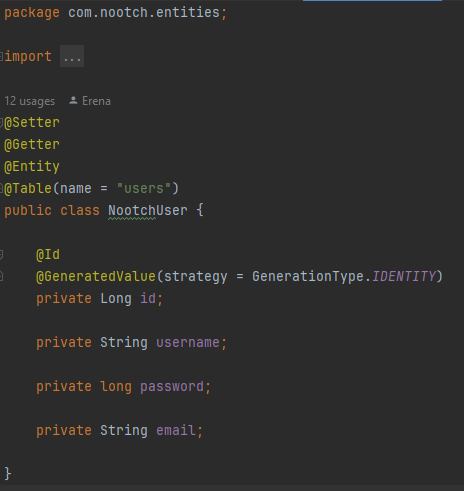
Mai departe utilizatorul își poate introduce datele pentru a se putea conecta la pagina principală. Datele sunt transmise către baza de date, iar dacă username-ul exista și parola este corectă, utilizatorul poate accesa pagina de pornire.



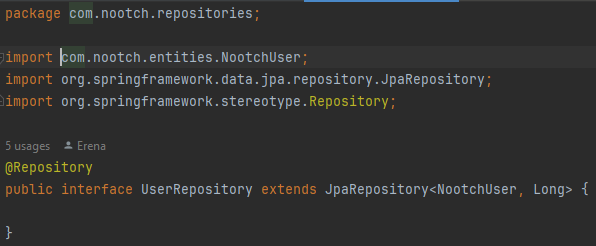
**Care sunt pașii pentru executare?**

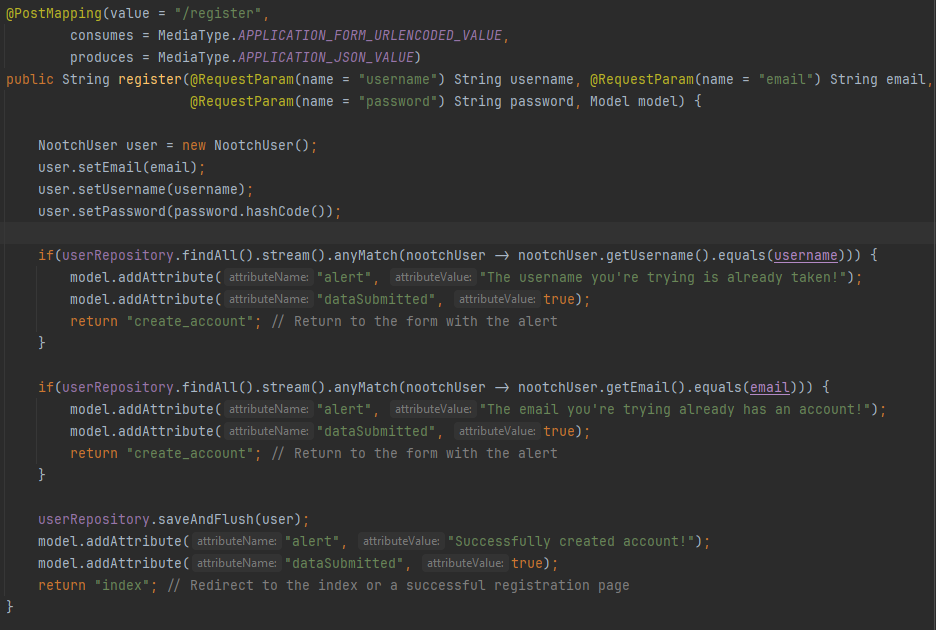
 Pentru început, trebuie instalată versiunea 2023.3.3 a mediului de programare în Java, IntelliJ. Apoi, pentru funcționarea corectă a fișierului pom.xml, trebuie instalat Apache Maven versiunea 3.9.6. În plus, pentru a putea compila un program java, este nevoie de un JDK, anume de liberica-17, care se va instala automat dacă Maven a fost instalat corect. Următorul pas este să intrăm în fișierul pom.xml și să apăsăm pe load maven changes. Dacă acești pasi au fost bine urmați programul se va putea rula dacă apăsăm pe butonul run.

**Secvențe de cod reprezentative**



În clasa NootchUser.java, definim entitatea User, care va genera tabelul din baza de date. Anotatiile @Setter și @Getter creează pentru fiecare coloana definită (id, username, password, email) o funcție de setter și de getter prin care putem obține sau modifica valorile respective pentru un anumit utilizator.



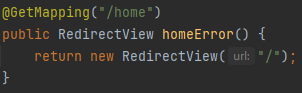
În clasa UserRepository.java definim funcțiile care prelucrează informațiile din baza de date, în cazul nostru folosim JpaRepository, o clasă cu funcții de tip CRUD (create, read, update, delete) și câteva alte funcții ajutătoare în plus.



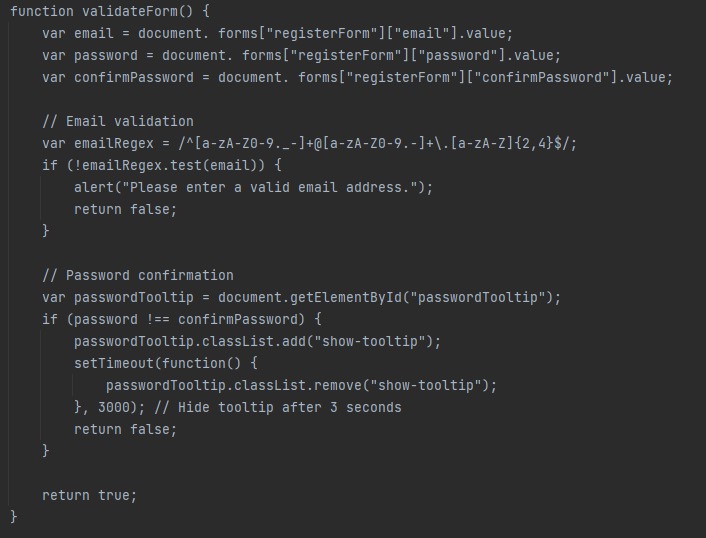
În clasa de controller a aplicației (NootchController) există funcția care ascultă dacă este accesată comanda de register, dacă da, funcția definită necesită cei 4 parametrii pentru a putea crea un utilizator nou. Apoi, printr-o filtrare făcută cu ajotorul funcțiilor lambda, se verifică dacă numele sau email-ul utilizatorului creat există deja în baza de date, dacă da, un mesaj de eroare va fi afișat și utilizatorul nu va fi adăugat în baza de date. În caz contrar, acesta va fi adăugat și va fi redirecționat automat la pagina de login.

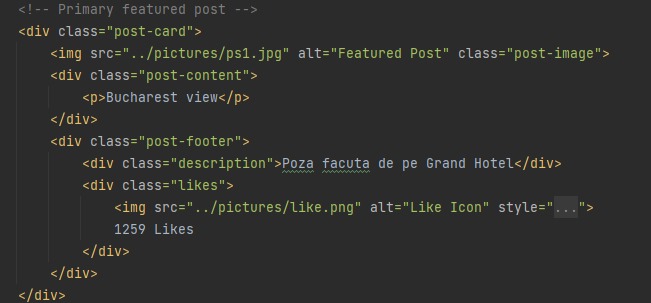
O alta funcție importantă este cea de login care necesită parametrii de username și de password. Folosind acești 2 parametri împreună cu un filtru făcut prin înlănțuirea a mai multor funcții lambda, se poate verifica dacă username-ul există în tabelul de User și dacă parola introdusă îi corespunde parolei din baza de date. Dacă aceste condiții nu sunt îndeplinite, se va afișa un mesaj de eroare, altfel utilizatorul va fi logat și redirecționat în pagina principala a aplicației.



 Încă o funcție importantă este cea prin care un utilizator este automat redirecționat la pagina de login dacă acesta încearcă să intre în pagina principală fară să fie logat.

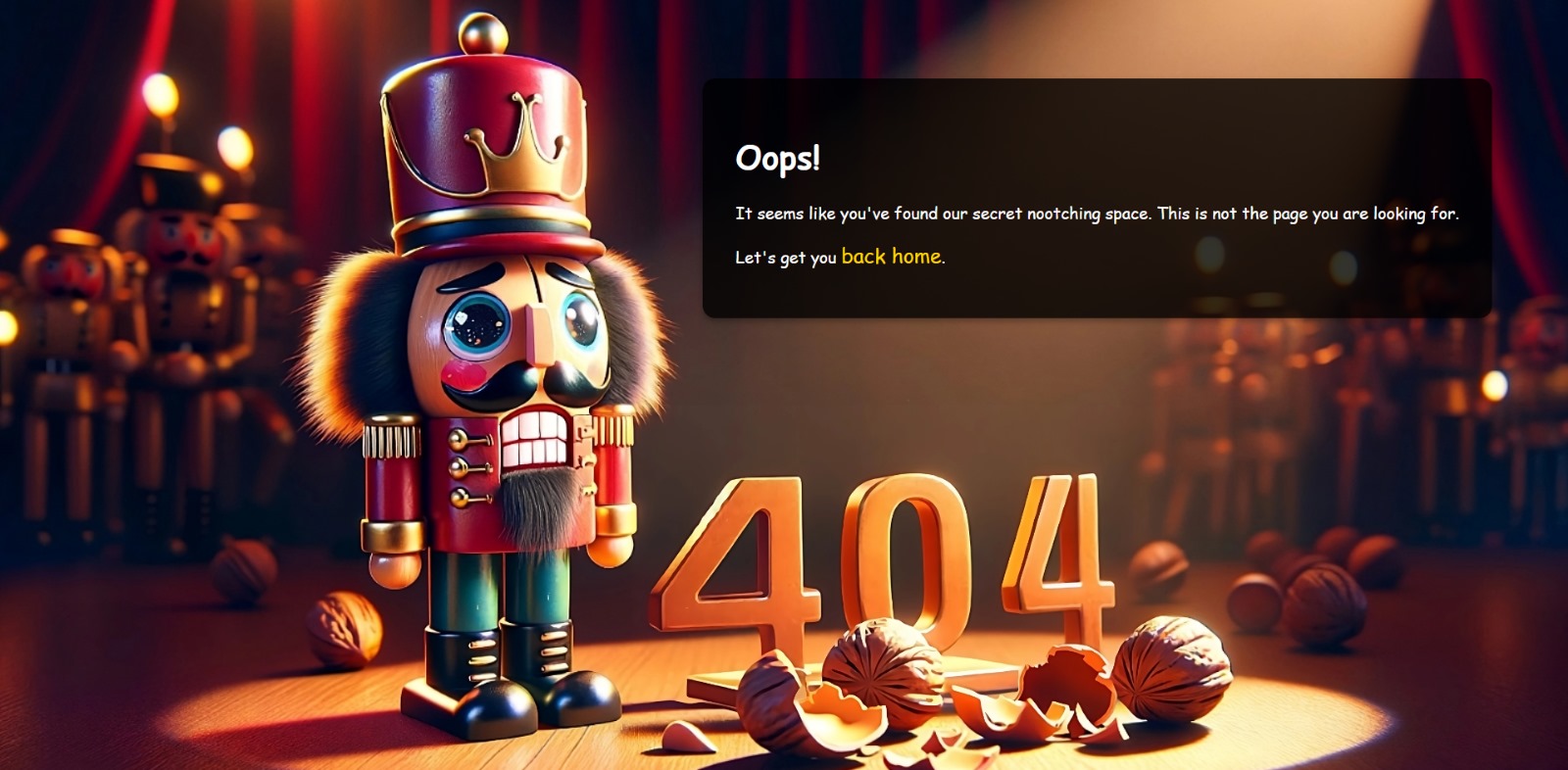
În fișierul create\_account.html, există o verificare pentru email și parolă care stabilește dacă utilizatorul a introdus datele în formatul corect..





Acesta este formatul unei postări pe care utilizatorii o pot viziona pe pagina principală. Se pot genera o infinitate de postări și se așază automat în continuare.

**Pagina de eroare**

 În cazul în care un utilizator caută o pagina care nu există sau dintr-un motiv aplicația returnează un cod de eroare http, utilizatorul va fi redirecționat automat la o pagină de eroare personalizată. Această setare este configurată în fișierul application.properties.



**Îmbunătățiri pentru viitor**

Următorul pas în dezvoltarea acestei aplicații ar fi de a adaugă mai multe funcții pentru utilizatori: de a crea postări, de a interacționa cu ele prin like-uri, prin dislike-uri, prin comentarii, de a vorbi prin mesaje cu alți utilizatori, de a urmări alți utilizatori și de a vedea pe pagina de pornire doar postări de la persoane urmărite sau postări foarte populare.

**Bibliografie**

<https://spring.io/projects/spring-boot>

<https://spring.io/guides/gs/spring-boot>

<https://en.wikipedia.org/wiki/Spring_Boot>

<https://stackoverflow.com/>

<https://www.jetbrains.com/idea/>

<https://en.wikipedia.org/wiki/Apache_Tomcat>

<https://stackoverflow.com/questions/62400393/serving-list-of-images-in-spring-boot>

<https://medium.com/@khairmuhammadmemon/spring-boot-data-access-with-spring-data-jpa-and-mysql-afe90e28b05d>

<https://gitlab.com/Mark_Erena/nootch-social-media>

<https://www.ibm.com/topics/java-spring-boot>

<https://www.javatpoint.com/spring-boot-tutorial>

<https://www.baeldung.com/>

<https://spring.io/guides#tutorials>

<https://chatgpt.com/>