UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

**DEPARTAMENTUL DE INGINERIE ELECTRICĂ, ELECTRONICĂ ȘI CALCULATOARE**

**Programul de studii: CALCULATOARE**

# TESTARE ONLINE

PROIECT PROIECTAREA APLICAȚIILOR WEB

Autor: **Bogdan-Sebastian BUIE**

**2024**

# Cuprins


**1 INTRODUCERE ..... 2**

1.1 CONTEXT GENERAL ..... 2
1.2 OBIECTIVE ..... 2
1.3 LISTA MOSCOW ..... 2
1.4 CAZURI DE UTILIZARE ..... 3

**2 ARHITECTURA ..... 4**

2.1 FRONT-END REACT ..... 4
2.2 BACK-END SPRING BOOT ..... 4
2.3 COMUNICARE ÎNTRE FRONT-END ȘI BACK-END ..... 4
2.4 BAZA DE DATE POSTGRESQL ..... 4
2.5 BENEFICII ALE ACESTEI ARHITECTURII ..... 5

**3 IMPLEMENTARE FRONTEND ..... 5**

3.1 TEHNOLOGII FRONTEND ..... 5
3.2 STRUCTURA COMPONENTELOR ..... 5
*3.2.1 Componentele folosite ..... 6*
3.3 UTILIZAREA APLICAȚIEI DIN PERSPECTIVA GUEST-ULUI ..... 7
3.4 UTILIZAREA APLICAȚIEI DIN PERSPECTIVA ADMINISTRATORULUI ..... 9

**4 IMPLEMENTARE BACKEND ..... 12**

4.1 TEHNOLOGII BACKEND ..... 12
*4.1.1 Spring Boot ..... 13*
4.1.1.1 Spring Data JPA ..... 13
4.1.1.2 Dependency Injection (DI) ..... 14
4.1.1.3 Spring Security ..... 15
*4.1.2 WebSockets. ..... 16*
4.1.2.1 Implementarea comunicației WebSocket: ..... 16
*4.1.3 PostgreSQL ..... 16*
*4.1.4 Maven ..... 17*
4.2 GESTIONAREA CERERILOR HTTP ..... 17

**5 CONCLUZII ..... 18**

**6 BIBLIOGRAFIE ..... 19**

# 1 Introducere

## 1.1 Context general

Aplicația web de testare online a cunoștințelor s-a născut din nevoia de a oferi utilizatorilor o modalitate eficientă și modernă de evaluare a competențelor în diverse domenii. Scopul fundamental este acela de a facilita procesul de evaluare a cunoștințelor într-un mod accesibil și interactiv.

## 1.2 Obiective

Scopul acestei aplicații web este de a ușura cât mai mult procesul de evaluare a cunoștințelor studenților/elevilor în cadrul unui examen sau test.

Prin dezvoltarea acestei aplicații se urmărește atingerea următoarelor obiective:

- Efectuarea unor teste/quiz-uri cât mai ușor de către studenți/elevi
- Gestiunea întrebărilor și quiz-urilor de către administrator
- Notificarea administratorului atunci când un student/elev face submit la test

## 1.3 Lista Moscow

În cadrul procesului de dezvoltare a aplicației de testare online a cunoștințelor, este foarte important să stabilim prioritățile și să identificăm cerințele în funcție de importanța lor strategică. Metoda Moscow, o abordare de analiză a cerințelor, oferă o structură clară pentru clasificarea acestora în patru categorii distincte:

- Must-haves (Trebuie aibă)
- Should-haves (Ar trebui să aibă)
- Could-haves (S-ar putea să aibă)
- Won't-haves (Nu va avea)

| Must have | Should have | Could have | Won't have |
|---|---|---|---|
| Efectuare Quiz | Vizualizarea notelor | Operații Update pentru quiz | Întrebări asociate fiecărui admin |
| Operații CRUD pentru întrebare | Notificarea terminării unui quiz | | |
| Operații Create, Read, Delete pentru quiz | | | |

*Tabel 1.1 Lista Moscow*

## 1.4 Cazuri de utilizare

Această aplicație este destinată atât studenților/elevilor cât și administratorului aplicației. În cadrul aplicației am identificat diverse cazuri de utilizare care reflectă scenariile principale pentru utilizatorii noștri. Aceste cazuri de utilizare ilustrează modul în care atât administratorii (Admin), cât și utilizatorii obișnuiți (Guest) interacționează cu aplicația, evidențiind funcționalitățile esențiale și oferind o viziune detaliată asupra experienței utilizatorului. Aceste cazuri de utilizare sunt evidențiate în următoarea figură:

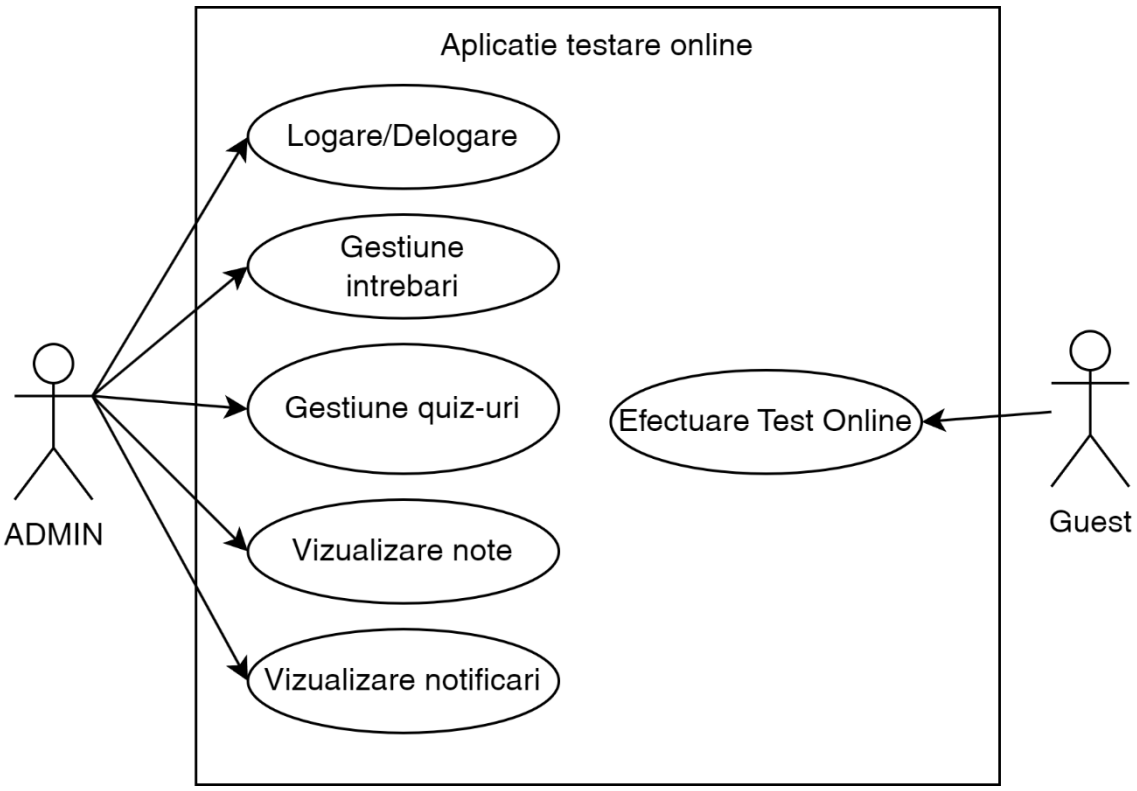



*Figure 1.1 Diagrama cazurilor de utilizare*

# 2 Arhitectura

Arhitectura aplicației este concepută pentru a asigura o performanță de înaltă calitate, scalabilitate și o interfață utilizator prietenoasă. Cu un front-end dezvoltat în React și un backend bazat pe Spring Boot, această arhitectură modernă îmbină tehnologii de ultimă generație pentru a oferi o experiență coerentă și robustă utilizatorilor.

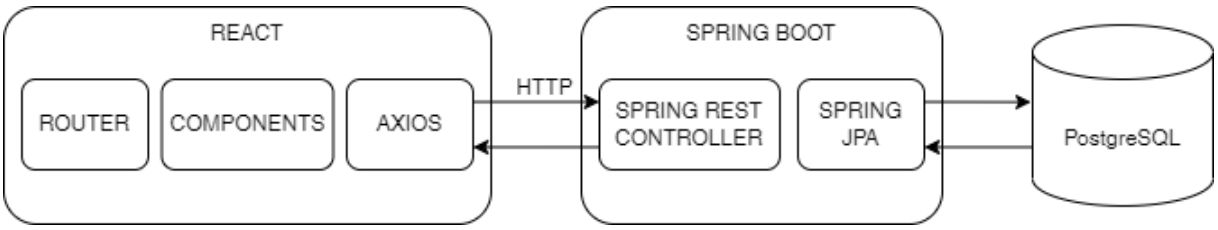



*Figure 2.1 Arhitectura aplicației*

## 2.1 Front-End React

React a fost ales pentru partea de front-end datorită modularității, flexibilității și vitezei sale de dezvoltare. Componentele React permit o gestionare eficientă a stării aplicației și facilitarea interacțiunii fluide cu utilizatorul. Structura bazată pe componente oferă un cod ușor de întreținut și extins.

## 2.2 Back-End Spring Boot

Spring Boot reprezintă coloana vertebrală a back-end-ului, furnizând un cadru robust și ușor de utilizat pentru dezvoltarea serviciilor. Utilizând Spring JPA, interacțiunea cu baza de date PostgreSQL devine transparentă și eficientă. Spring Boot oferă, de asemenea, facilități pentru gestionarea dependențelor, configurării, și dezvoltarea rapidă a serviciilor RESTful.

## 2.3 Comunicare între Front-End și Back-End

Comunicarea între front-end și back-end este gestionată prin intermediul bibliotecii Axios. Aceasta permite realizarea de cereri HTTP eficiente și gestionarea răspunsurilor, facilitând schimbul de date între cele două părți ale aplicației. Abordarea asincronă a Axios contribuie la optimizarea timpului de încărcare și la o experiență de utilizare mai fluidă.

## 2.4 Baza de Date PostgreSQL

PostgreSQL a fost ales ca sistem de gestionare a bazelor de date datorită robusteții sale, scalabilității și suportului pentru caracteristici avansate. Interfața Spring JPA simplifică interacțiunea cu baza de date, oferind o abordare elegantă pentru manipularea datelor într-un mod eficient și securizat.

## 2.5 Beneficii ale acestei arhitecturii

Scalabilitate: Arhitectura permite extinderea facilă a capacităților aplicației pe măsură ce numărul de utilizatori crește.

Eficiență în dezvoltare: Utilizarea framework-urilor populare precum React și Spring Boot contribuie la o dezvoltare rapidă și ușoară.

Performanță optimizată: Comunicarea eficientă între front-end și back-end, împreună cu utilizarea unei baze de date robuste, asigură o performanță optimă a aplicației.

Flexibilitate și extensibilitate: Modularitatea și structura componentelor permit o adaptabilitate și extensibilitate ușoară, facilitând adăugarea de funcționalități noi.

Arhitectura noastră reflectă angajamentul nostru față de o experiență utilizator de calitate și o dezvoltare durabilă și eficientă. Prin integrarea tehnologiilor de vârf, ne propunem să construim o aplicație care să satisfacă atât nevoile actuale și să avem în vedere pe nevoile viitoare ale utilizatorilor noștri.

# 3 Implementare frontend

În dezvoltarea front-end-ului al acestei aplicații de testare online a cunoștințelor, am adoptat o abordare meticuloasă pentru a oferi utilizatorilor o experiență intuitivă și plăcută. Front-end-ul reprezintă interfața principală prin care utilizatorii interacționează cu platforma noastră, iar alegerea tehnologiilor potrivite și implementarea unei structuri coerente au fost priorități cheie în acest proces.

## 3.1 Tehnologii frontend

Am optat pentru React ca și bibliotecă principală pentru dezvoltarea front-end-ului datorită modularității sale, reutilizabilității componentelor și performanței sale. Integrarea React ne-a permis să construim o interfață de utilizator dinamică și ușor de întreținut.

În plus, am folosit limbajele de bază ale web-ului: HTML și JavaScript, pentru a crea structura și funcționalitatea paginilor, iar pentru aspectul paginilor am folosit CSS și framework-ul Bootstrap.

## 3.2 Structura componentelor

Structura componentelor este organizată modular pentru a facilita întreținerea și extinderea ulterioară. Componentele create sunt concepute să fie reutilizabile, favorizând coerența și eficiența în dezvoltarea front-end-ului.

### 3.2.1 Componentele folosite

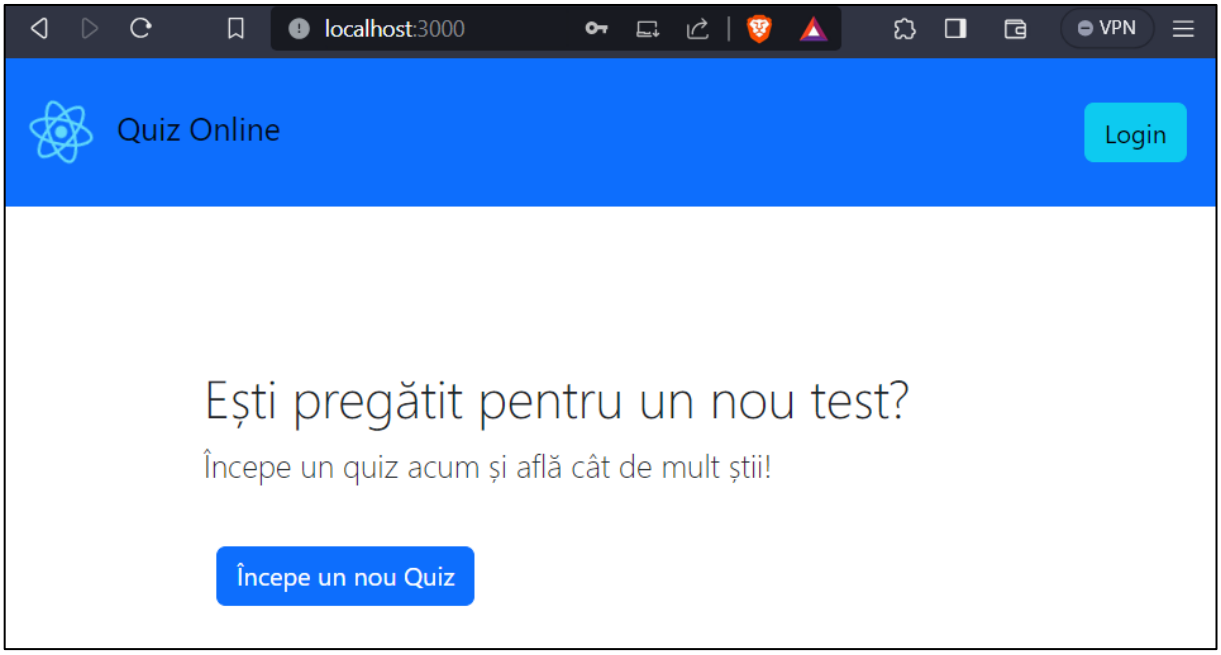
Principalele componente folosite în cadrul frontend-ului sunt centralizate în tabelul următor.

| Numele componentei | Rolul componentei | Utilizată la endpoint-ul |
|---|---|---|
| App | Componenta centrală | |
| Header | Header-ul care se află pe orice pagină a aplicației și care conține câteva butoane utile | Toate |
| Home | Pagina de pornire. Conține un buton către o altă pagină | / |
| PublicContent | Pagină pentru introducerea unei parole și a unui id pentru a accesa un anumit quiz | /public |
| QuizComponent | Facilitează introducerea datelor despre student/elev și alegerea răspunsurilor la quiz | /getQuiz/:id |
| FinishedQuiz | Informații după terminarea unui quiz | /finishedQuiz |
| LoginContent | Facilitează logarea și delogarea din aplicație pentru admin | /login |
| Menu | Facilitează alegerea unui activități:<br>• Gestiune întrebări<br>• Gestiune quiz-uri<br>• Vizualizare note<br>• Vizualizare notificări | /menu |
| AddQuestion | Formular pentru introducerea unei noi întrebări | /addQuestion |
| ViewQuestions | Afișează o lista cu toate întrebările din sistem | /questions |
| EditQuestion | Formular pentru editarea unei întrebări | /editQuestion/:id |
| QuestionDetail | Afișează toate detaliile unei întrebări | /viewQuestion/:id |
| AddQuiz | Formular pentru crearea unui quiz | /addQuestion |
| ViewQuizzes | Afișează toate quiz-urile disponibile | /quizzes |
| ViewQuizQuestions | Afișează toate întrebările asociate unui quiz | /viewQuizQuestions/:id |
| ViewSubmissions | Afișează lista de note a studenților/elevilor | /note |

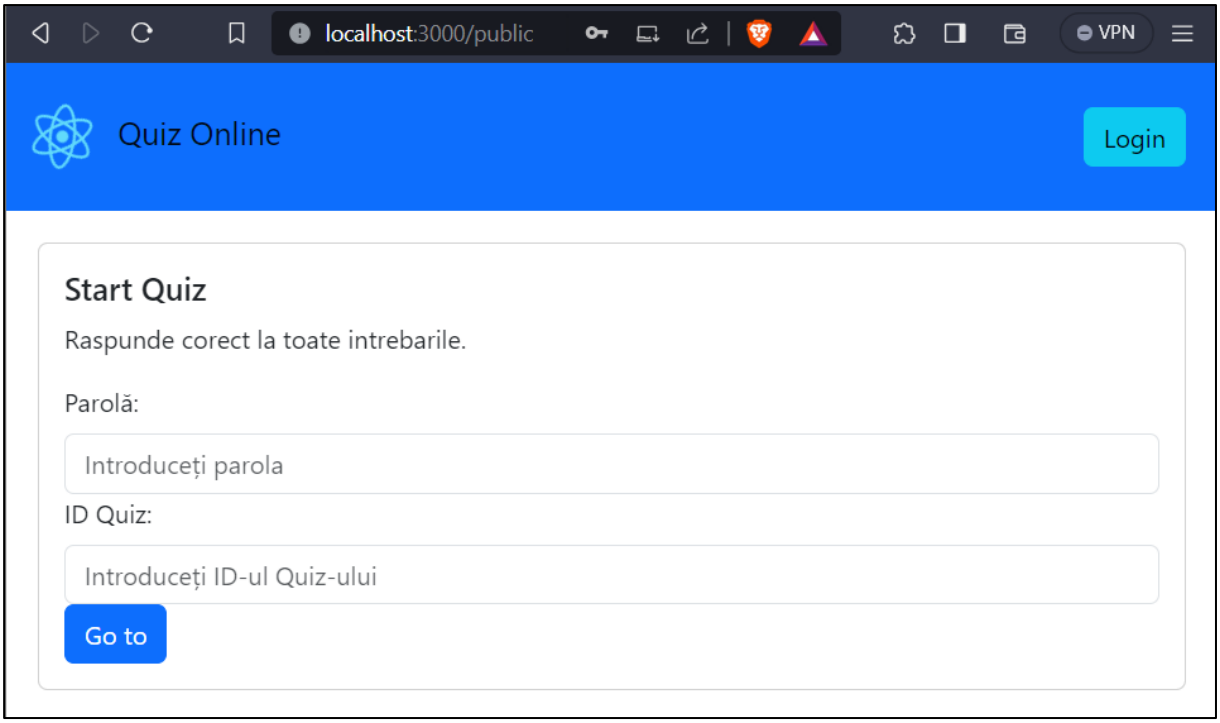
| | | |
|---|---|---|
| NotificationComponent | Afișează notificări atunci când un student/elev face submit la un quiz | /notification |

## 3.3 Utilizarea aplicației din perspectiva guest-ului

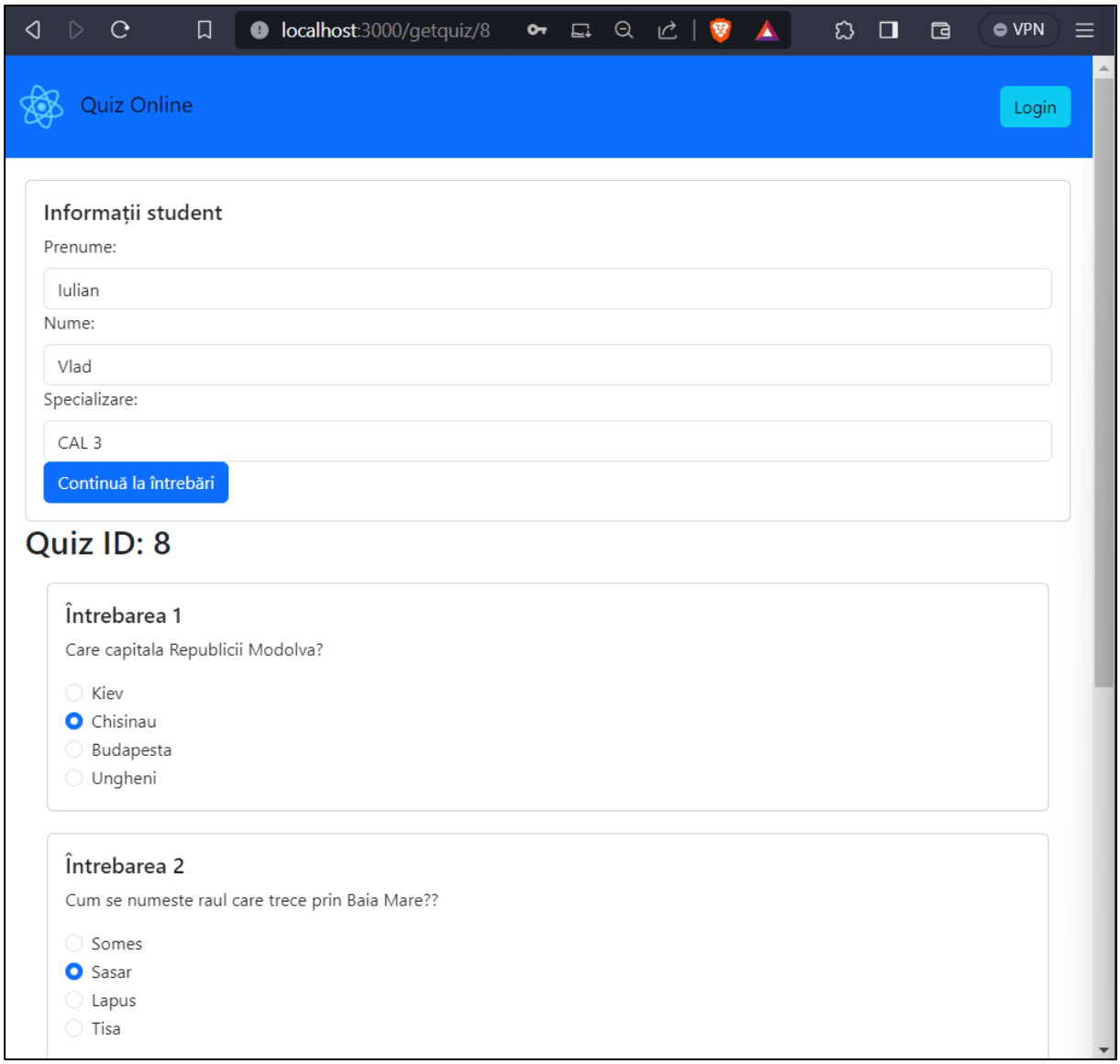
1. Pagina de pornire




2. După apăsarea pe butonul *Începe un nou quiz* va apărea un formular în care trebuie introduse o parolă și un id (acestea sunt furnizate de către profesor/admin). Guest-ul nu are nevoie de cont în această aplicație.

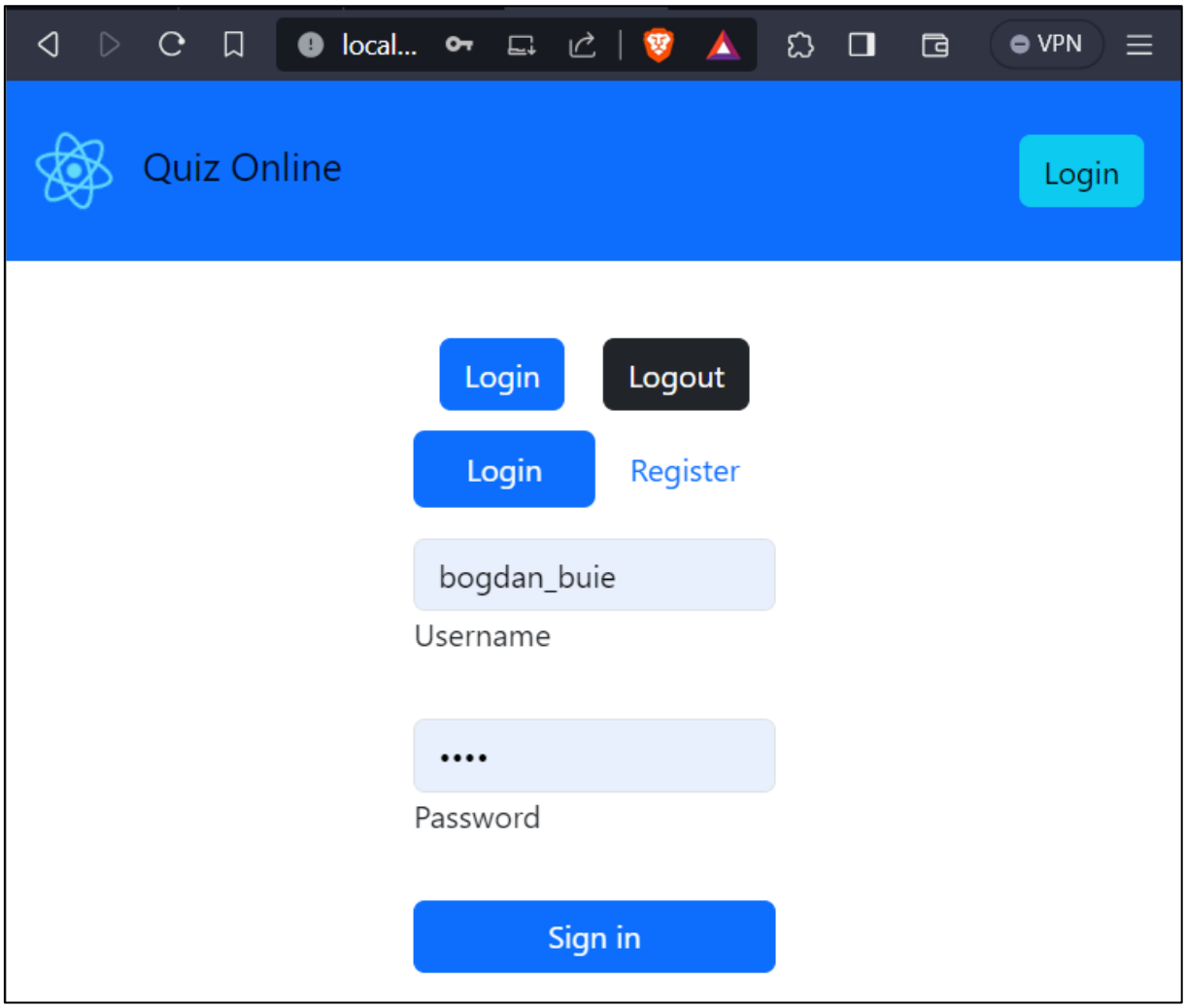


3. După completarea datelor se deschide o nouă fereastră unde vor trebui completate câteva date despre student. După care se apasă butonul *Continuă la întrebări,* iar în continuare se vor afișa întrebările quiz-ului. Dacă toate datele din formular sunt complete, atunci se vor putea afișa întrebările. La finalizarea quiz-ului se va apăsa butonul *Finalizaeză quiz-ul.* În continuare se va deschide o nouă pagină unde utilizatorul este informat că quiz-ul s-a terminat și că va primi nota de la profesor în scurt timp.

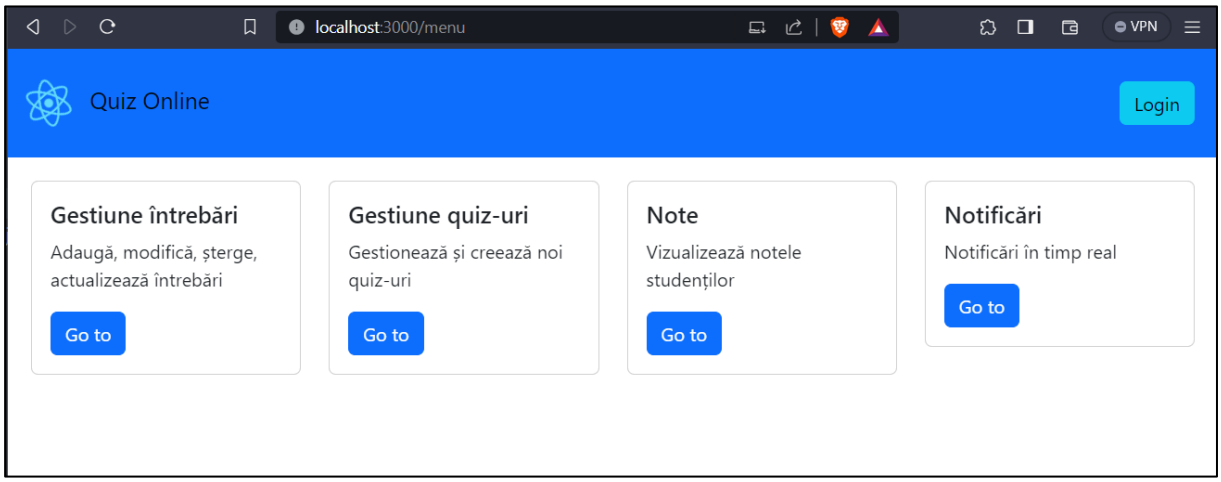


## 3.4 Utilizarea aplicației din perspectiva administratorului

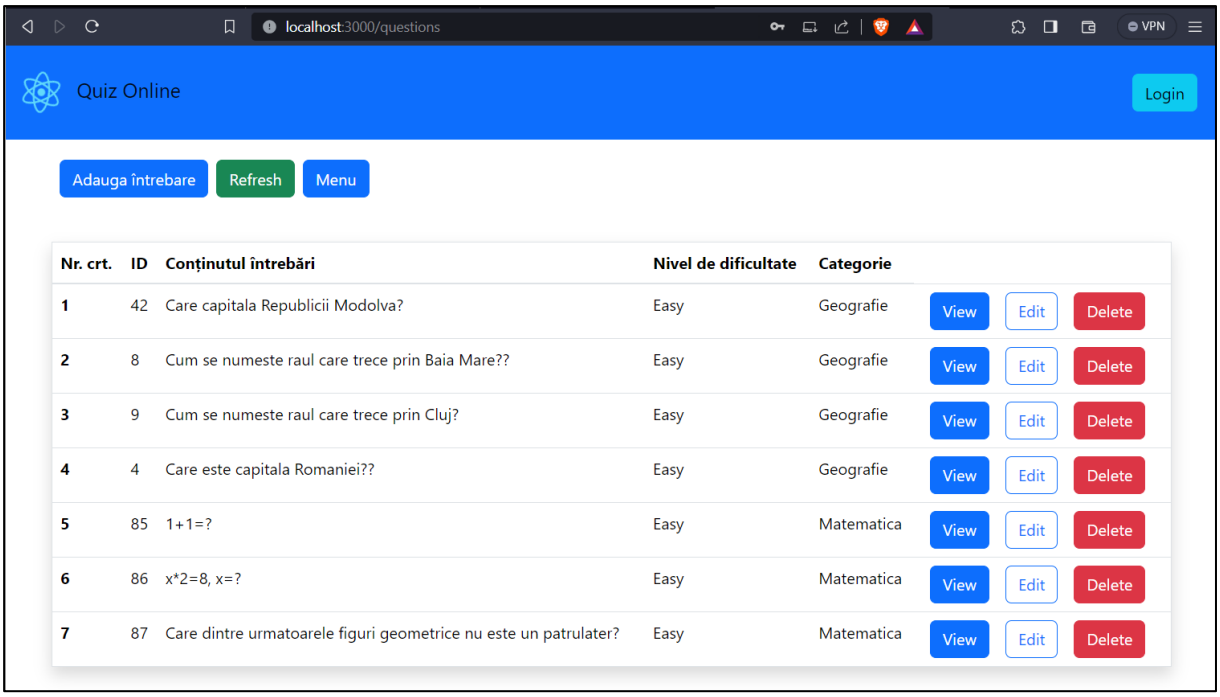
1. După apăsarea butonului Login de la Header se deschide următoarea fereastră:




2. După completarea credențialelor se afișează o pagină cu un meniu care conține mai multe opțiuni

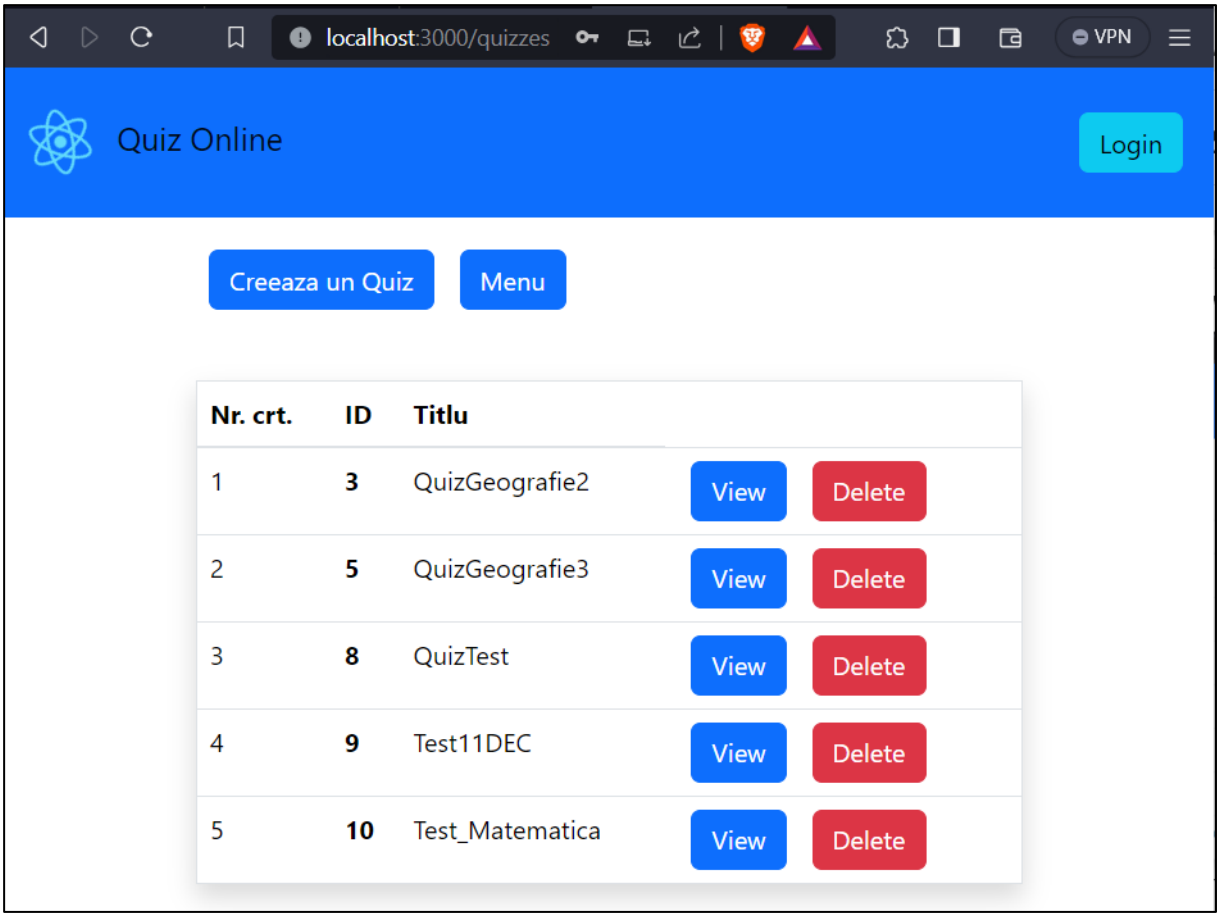



3. Dacă se alege opțiunea *Gestiune întrebări* se va deschide următoarea pagină

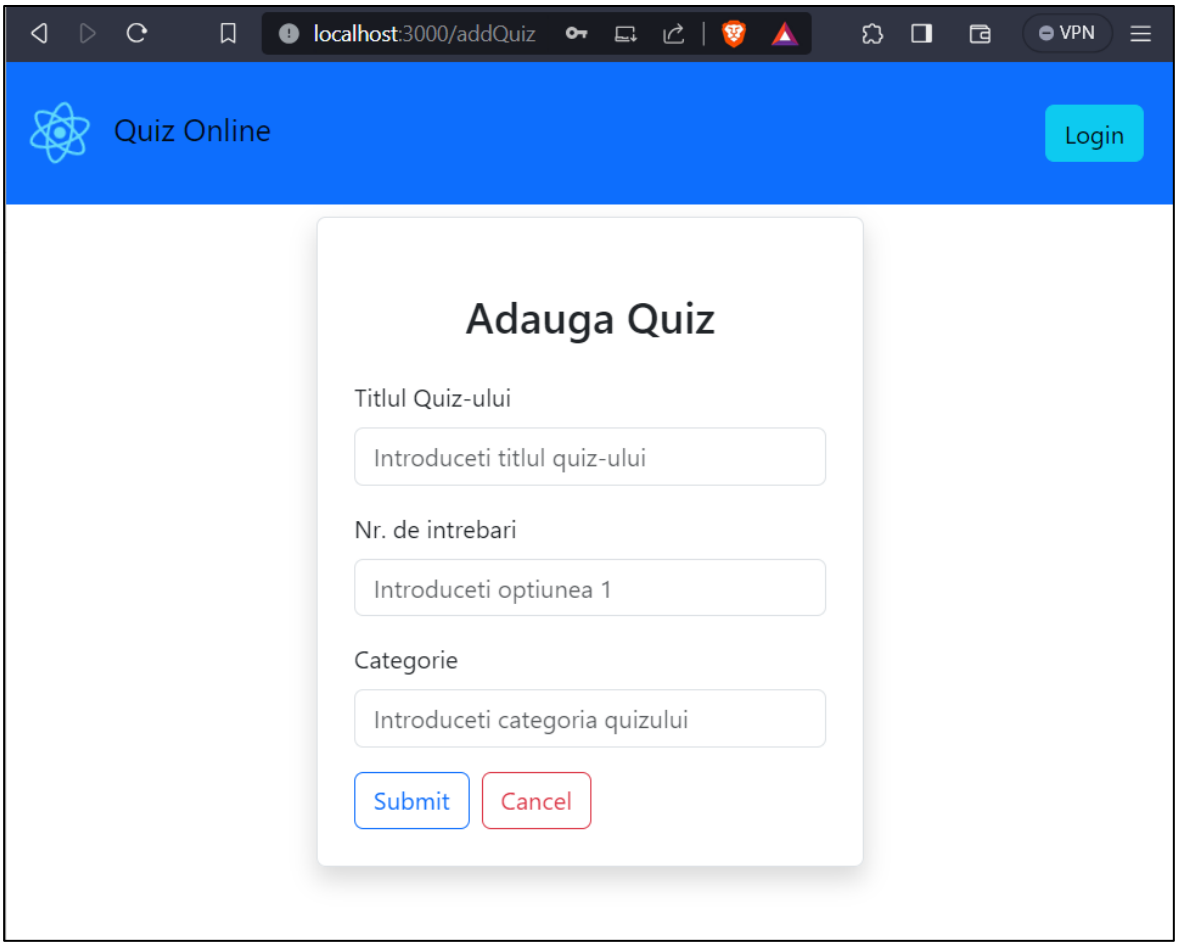



Această pagină conține toate întrebările din sistem. Din această pagină se poate crea o nouă întrebare sau se poate edita/vizualiza/șterge o anumită întrebare.

4. Dacă se alege opțiunea *Gestiune Quiz-uri* se va deschide următoarea fereastră:

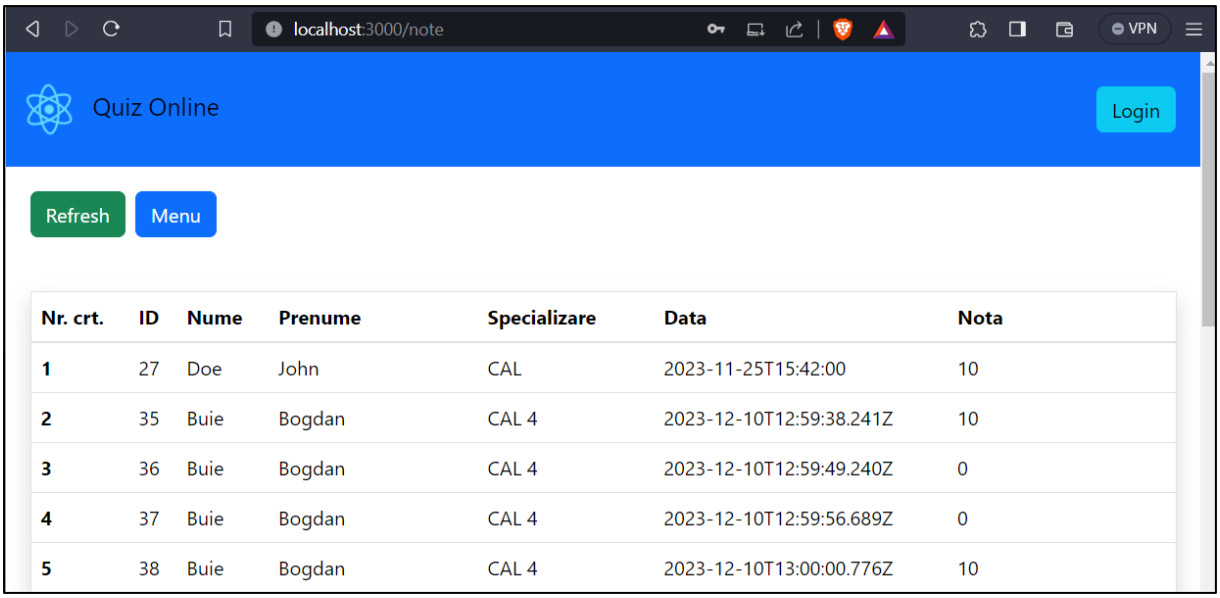


Această fereastră conține toate întrebările din sistem. Din această pagină se poate creea un quiz nou, se poate șterge un quiz sau se poate vizualiza întrebările asociate unui quiz. Atunci când se creează un nou quiz se deschide un formular în care trebuie specificate: titlul quiz-ului, numărul de întrebări și categoria quiz-ului. Quiz-ul se creează automat pe baza acestor date, selectând din baza de date întrebări random din categoria specificată anterior.

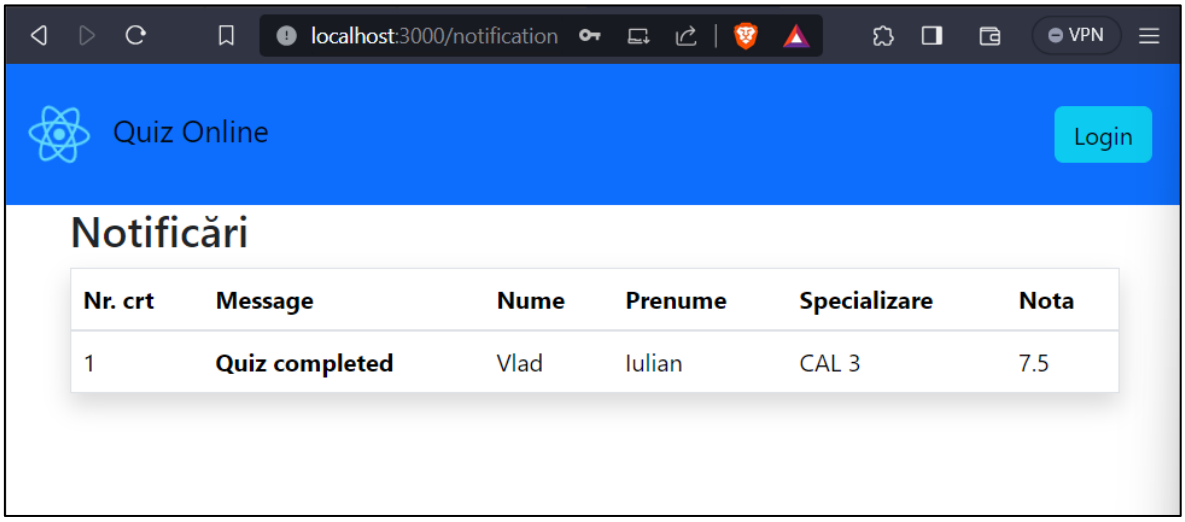



5. Dacă se alege opțiunea *Note* se va afișa o listă care va conține:

- Numele studentului
- Prenumele studentului
- Specializarea
- Ora și data când s-a efectuat submit la quiz
- Nota obținută




6. Dacă se alege opțiunea *Notificări* se va afișa o fereastră goală la început, dar pe măsură ce studenții fac submit la quiz va apărea o notificare din partea fiecăruia.




# 4 Implementare backend

## 4.1 Tehnologii backend

Backend-ul aplicației servește drept coloană vertebrală, gestionând interacțiunile, prelucrarea datelor și furnizând suportul necesar pentru funcționalitățile cheie. Alegerea tehnologiilor pentru backend a fost orientată spre eficiență, scalabilitate și ușurință în dezvoltare.

### 4.1.1 Spring Boot

Spring Boot este un framework Java care facilitează dezvoltarea rapidă a aplicațiilor Java. Alegerea acestuia se datorează modularității sale, capacității de a gestiona dependențe, și abordării convention-over-configuration, care aduce un nivel înalt de consistență în proiect.

#### *4.1.1.1 Spring Data JPA*

Java Persistence API (JPA) reprezintă un standard Java pentru gestionarea datelor în aplicații Java. Integrarea JPA în proiect aduce beneficii semnificative în ceea ce privește manipularea datelor în baza de date, permițând lucrul cu obiecte Java și persistența eficientă a acestora într-o bază de date relațională.

- Exemplu de implementare:
  - **Entitate JPA**: Definirea entităților JPA pentru obiectele de domeniu, care vor fi mapate în tabele în baza de date, oferind o abordare obiect-relațională.

```java
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
@Entity
@Table(name = "app_user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name", nullable = false)
    @Size(max = 100)
    private String firstName;

    @Column(name = "last_name", nullable = false)
    @Size(max = 100)
    private String lastName;

    @Column(nullable = false)
    @Size(max = 100)
    private String login;

    @Column(nullable = false)
    @Size(max = 100)
    private String password;
}
```

  - **Repository JPA**: Utilizarea interfețelor JPA Repositories pentru gestionarea operațiunilor CRUD (Create, Read, Update, Delete) pentru entități.

```java
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByLogin(String login);
}
```

- Avantaje ale Spring Boot JPA:
    - **Abordare Obiect-Relațională**: JPA permite lucrul cu obiecte Java în locul interogărilor SQL directe, oferind o abordare mai orientată pe obiect în gestionarea datelor.
    - **Reutilizabilitate și Consistență**: Utilizarea entităților JPA și a repository-urilor abstractizează detaliile tehnice ale manipulării datelor, facilitând reutilizarea și menținerea consistenței.

#### *4.1.1.2 Dependency Injection (DI)*

Dependency Injection este un concept fundamental în programarea orientată pe obiect, unde obiectele nu își creează direct dependențele, ci le primesc injectate din exterior. Spring Framework facilitează implementarea Dependency Injection, aducând beneficii semnificative în gestionarea dependențelor în întreaga aplicație.

- Exemplu de implementare
    - **Componente Spring**: Definirea claselor care vor beneficia de Dependency Injection prin anotarea lor cu @Component, @Service, sau @Repository, în funcție de rolul lor.

```
...
import lombok.RequiredArgsConstructor;

@RequiredArgsConstructor
@Service
public class UserService {

    private final UserRepository userRepository;

    private final PasswordEncoder passwordEncoder;

    private final UserMapper userMapper;

    public UserDto login(CredentialsDto credentialsDto) {
        ...
    }

    public UserDto register(SignUpDto userDto) {
        ...
    }

    public UserDto findByLogin(String login) {
        ...
    }

}
```

- Avantaje:
    - **Decuplare și testare ușoară**: Dependency Injection promovează decuplarea între componente, facilitând testarea unitară și schimbul simplu al implementărilor.

- **Gestionarea ciclurilor de viață**: Spring se ocupă de gestionarea ciclurilor de viață ale componentelor, asigurând crearea și distrugerea corespunzătoare a acestora.
- **Configurare ușoară**: Configurarea dependențelor se face în mod declarativ, adesea prin intermediul XML sau a notărilor, facilitând gestionarea și modificarea dependențelor.

#### *4.1.1.3 Spring Security*

JSON Web Token (JWT) este un standard deschis (RFC 7519) care definește un format compact și autentic de transmitere a informațiilor între părți sub formă de obiecte JSON. În contextul unei aplicații Spring Boot, JWT este adesea utilizat pentru gestionarea autentificării și autorizării utilizatorilor. Iată cum funcționează:

1. **Generarea token-ului JWT**:
   - După ce un utilizator se autentifică cu succes, serverul generează un token JWT.
   - Acest token conține informații precum identitatea utilizatorului, rolurile și alte atribute relevante.
   - Token-ul este semnat digital, adăugându-i o cheie de securitate pentru a garanta integritatea.
2. **Validarea și extragerea informațiilor din token**
   - Utilizatorii autentificați trimit token-ul în header-ul cererilor lor către resursele protejate.
   - Serverul validează și decriptează token-ul pentru a extrage informații, cum ar fi identitatea utilizatorului și drepturile de acces.
3. **Configurarea Spring Security pentru JWT**

   Spring Security este configurat pentru a utiliza filtre personalizate care gestionează autentificarea și autorizarea bazate pe JWT. Aceste filtre sunt integrate în lanțul de securitate al aplicației.
4. **Utilizarea informațiilor din token**

   Informațiile extrase din token sunt utilizate pentru a lua decizii la nivelul aplicației, precum gestionarea accesului la anumite funcționalități.

**Avantaje ale JWT:**

- Stateless și Eficiență:

  JWT permite aplicației să fie stateless, eliminând necesitatea de a menține sesiuni la nivel de server. Token-ul conține toate informațiile necesare, evitând interacțiunile constante cu o bază de date pentru validare.

- Flexibilitate și Siguranță:

  JWT oferă flexibilitate prin includerea unui payload semnat digital, asigurând integritatea și autenticitatea informațiilor.

Prin implementarea JWT în Spring Boot, se obține o metodă eficientă și sigură de gestionare a autentificării și autorizării, oferind un mecanism robust pentru securitatea aplicației.

### 4.1.2 WebSockets

WebSockets reprezintă o tehnologie de comunicație bidirecțională între client și server, permitând transmiterea instantanee a informațiilor fără necesitatea reîncărcării paginii.

Folosirea WebSockets pentru notificarea administratorului în momentul în care un student finalizează un quiz aduce un nivel de interactivitate și timp real în experiența utilizatorului. În contextul notificării administratorului la submit-ul unui quiz, iată câteva aspecte relevante:

#### *4.1.2.1 Implementarea comunicației WebSocket:*

**1. Stabilirea conexiunii WebSocket**

Admin-ul stabilește o conexiune WebSocket cu serverul înainte de a începe monitorizarea quiz-urilor. Acesta poate face asta într-o secțiune dedicată a aplicației.

**2. Procesarea pe backend**

Când un student finalizează un quiz, datele sunt trimise către backend pentru procesare. Serverul calculează nota studentului și orice alte informații relevante în urma submit-ului.

3. **Trimiterea notificării către admin**

După ce nota este calculată cu succes, serverul trimite o notificare prin conexiunea WebSocket către admin. Notificarea conține informații despre studentul care a finalizat quiz-ul, nota obținută și alte detalii relevante.

### 4.1.3 PostgreSQL

Am optat pentru PostgreSQL ca și sistem de gestionare a bazelor de date, datorită stabilității, performanței, și funcționalităților avansate pe care le oferă. PostgreSQL suportă tipuri de date complexe, tranzacții ACID, și este recunoscut pentru gestionarea eficientă a volumelor mari de date.

### 4.1.4 Maven

Maven este folosit pentru gestionarea dependințelor și construirea proiectului. Prin intermediul Maven, putem gestiona eficient dependințele externe, asigurând astfel coerența și ușurința în gestionarea codului.

Aceste tehnologii formează baza solidă a backend-ului nostru, contribuind la dezvoltarea și întreținerea eficientă a aplicației noastre de testare online a cunoștințelor. Prin alegerea acestor instrumente și tehnologii, ne propunem să atingem un echilibru optim între performanță, scalabilitate și ușurința în dezvoltare.

## 4.2 Gestionarea cererilor HTTP

În cadrul aplicației, gestionarea cererilor HTTP reprezintă un aspect fundamental al backend-ului, facilitând comunicarea eficientă între frontend și server. Ne bazăm pe principiile arhitecturii RESTful pentru a proiecta API-urile noastre, asigurând o structură clară și ușor de înțeles.

Am implementat o serie de endpoint-uri care oferă funcționalități diverse în funcție de cerințele aplicației noastre. Acestea sunt expuse printr-un set de endpoint-uri bine definite, care acceptă metode HTTP standard, precum GET, POST, PUT și DELETE. Aceste endpoint-uri sunt centralizate în tabelul următor:

| Endpoint | | | Metoda HTTP | Utilizare |
|---|---|---|---|---|
| http://localhost:8080/api/v1 | /login | | POST | Autentificare în aplicație |
| | /register | | POST | Creează un cont nou |
| | /question | /all | GET | Furnizează toate întrebările |
| | | /category{category} | GET | Furnizează toate întrebările după o categorie dată |
| | | /add | POST | Adaugă o întrebare nouă |
| | | /update/{id} | PUT | Modifică o întrebare deja existentă |
| | | /delete/{id} | DELETE | Șterge o întrebare deja existentă |
| | /quiz | /get/{id} | GET | Furnizează toate întrebările asociate unui quiz, fără furniza și răspunsul acestora |
| | | /getSecret/{id} | GET | Furnizează toate întrebările asociate unui quiz |
| | | /create | POST | Creează un quiz pe baza unor date |
| | | /submit3/{id} | POST | Calculează și salvează nota studentului de la quiz |
| | | /delete/{id} | DELETE | Șterge un anumit quiz |

Link Github: https://github.com/bogdan-buie/TestareOnline/tree/branch1

# 5 Concluzii

Posibilitățile de dezvoltare în cadrul acestei aplicații sunt ample și promițătoare, oferind oportunități pentru îmbunătățiri continue și extinderea funcționalităților.

Începând cu cele mai mici îmbunătățiri ale aplicației, precum ar fi: posibilitatea de a avea întrebări cu răspunsuri multiple, răspunsuri și întrebări care să conțină o imagine sau chiar un scurt videoclip, implementarea conceptului de clasă(fiecare profesor să poată gestiona un grup de studenți).

Prin integrarea tehnologiei WebSocket pentru notificarea în timp real a administratorului atunci când se efectuează submit la un quiz, s-a deschis o poartă către un mediu mai dinamic și interactiv. În continuare, există perspective pentru adăugarea unor caracteristici precum analize statistice avansate, rapoarte personalizate, sau chiar integrarea unor opțiuni de gamification pentru a stimula angajamentul studenților.

În concluzie, întreaga aplicație reprezintă un ansamblu coerent și eficient de tehnologii și funcționalități menite să ofere o experiență avansată și personalizată atât pentru administratori, cât și pentru utilizatori. Implementarea front-end-ului cu React, backend-ul cu Spring Boot și integrarea WebSockets aduc un nivel înalt de interactivitate, eficiență și comunicare în timp real. Capacitatea de a crea, administra și evalua quiz-uri se îmbină armonios cu tehnologiile moderne, precum JWT pentru securitate, PostgreSQL pentru gestionarea datelor, și Spring JPA pentru manipularea acestora. Posibilitățile ample de dezvoltare permit extinderea funcționalităților, adăugarea de caracteristici inovatoare și adaptarea la nevoile specifice ale utilizatorilor. În ansamblu, aplicația nu doar îndeplinește scopul de a facilita procesul de testare online a cunoștințelor, ci și deschide orizonturi pentru o evoluție continuă și adaptare la cerințele în schimbare ale domeniului educațional și tehnologic.

# 6 Bibliografie


1. “Spring  Boot Documentation”, 2023, Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
2. “React Documentation”, 2023, Available: https://legacy.reactjs.org/docs/getting-started.html
3. Note de curs ”Proiectarea aplicațiilor Web”, 2023
4. Note de curs ”Programare Web”, 2022
5. “Spring Boot WebSockets”, 2023, Available: https://spring.io/guides/gs/messaging-stomp-websocket/