APLICAȚIE DE TESTARE ONLINE

Titea Dan Ruben

Context și Obiective

Scopul Proiectului

- Digitalizarea procesului de examinare prin teste grilă.
- Oferirea unei platforme accesibile atât vizitatorilor, cât și utilizatorilor înregistrați.

Obiective Principale

- **Scalabilitate:** Arhitectură decuplată (Client-Server).
- **Interactivitate:** Monitorizare în timp real a participanților.
- **Securitate:** Autentificare JWT și validare server-side.
- **Flexibilitate:** Sistem complet de management al testelor (CRUD).

Lista MoSCoW

Prioritizarea funcționalităților a fost realizată conform metodologiei MoSCoW, clasificând cerințele în funcție de importanța lor strategică și impactul asupra produsului finit.

Must-have

- Creare, editare, ștergere teste de către utilizatori autentificați
- Participare la teste pentru utilizatori autentificați și guest
- Autentificare și autorizare securizată (JWT)

Should-have

- Căutare teste după titlu
- Sortare teste după domeniu și data creării
- Afișare în timp real a utilizatorilor care participă (WebSockets)
- Gestionarea profilului utilizatorului

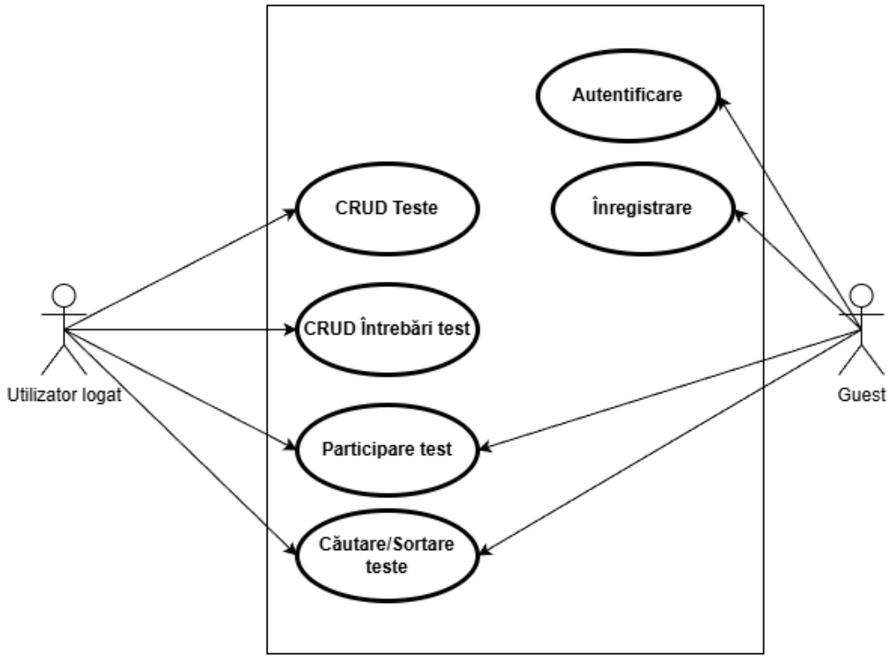
Could-have

- Statistici detaliate despre performanța utilizatorilor
- Sistem de clasament pentru utilizatori
- Notificări prin email pentru finalizarea testelor
- Personalizare interfață utilizator

Won't-have

- Generare automată de întrebări
- Integrare cu platforme externe de învățare
- Suport pentru teste multimedia (imagini, video)

Cazuri de Utilizare



1

Vizitator (Guest)

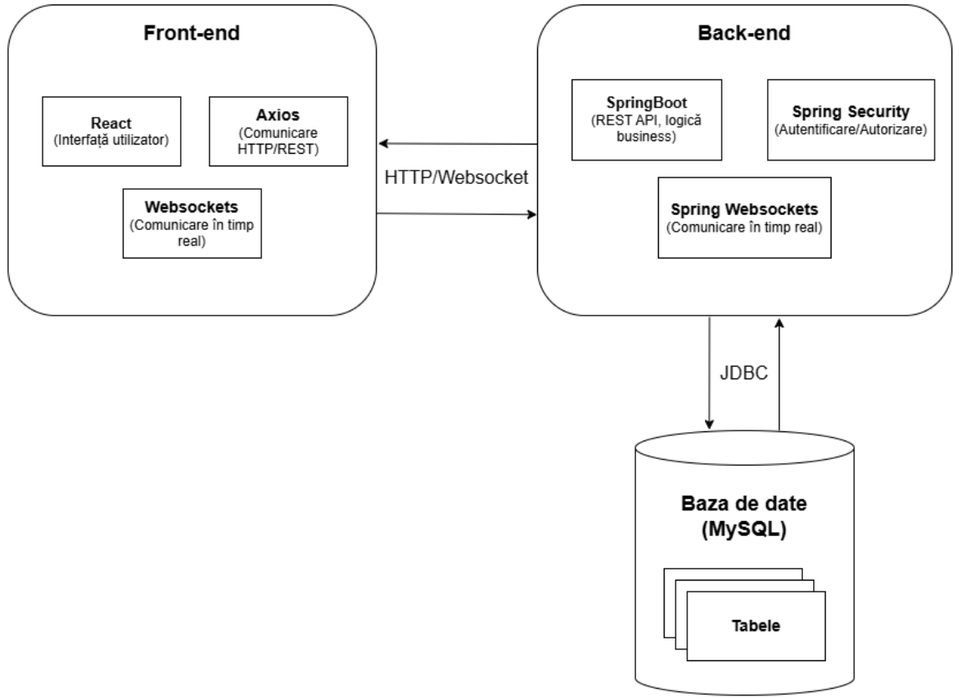
- Accesează lista de teste publice disponibile.
- Poate susține teste, cu o identitate temporară generată automat.
- Vizualizează rezultatele testelor imediat după finalizare.
- Vede alți participanți activi în timp real în timpul testului.

2

Utilizator Autentificat

- **Include toate permisiunile unui Vizitator.**
- **Creare Teste:** Poate defini titlul, domeniul, întrebările și răspunsurile pentru teste noi.
- **Editare/Ștergere:** Administrează doar testele proprii create.
- **Acces Profil:** Vizualizează profilul personal.

Arhitectura Aplicației (SPA)



Aplicația este construită pe o arhitectură modernă, de tip Single Page Application (SPA), care asigură o experiență utilizator rapidă și fluentă.

Componente Cheie

- **Front-End:** React (Single Page Application).
- **Back-End:** Spring Boot (REST API).
- **Baza de Date:** MySQL (Stocare persistentă).

Mecanisme de Comunicare

- **Comunicare Standard:** HTTP (Axios).
- **Comunicare în Timp Real:** WebSocket (STOMP).

Stack Tehnologic

Front-End

- **React 19 & Vite:** Pentru performanță și modularitate, asigurând o interfață rapidă și reactivă.
- **React Router:** Gestionarea navigației în cadrul aplicației SPA, oferind o experiență unitară.
- **Axios:** Client HTTP pentru efectuarea cererilor asincrone către API-ul back-end.

Back-End

- **Java Spring Boot:** Cadrul principal pentru dezvoltarea API-ului RESTful, oferind rapiditate și eficiență.
- **Spring Security:** Pentru autentificare și autorizare, inclusiv implementarea JWT.
- **Spring Data JPA:** Abstracție ORM peste Hibernate, simplificând interacțiunea cu baza de date.
- **Spring WebSocket:** Suport pentru comunicare în timp real, esențial pentru monitorizarea participanților.

Baza de Date & Tools

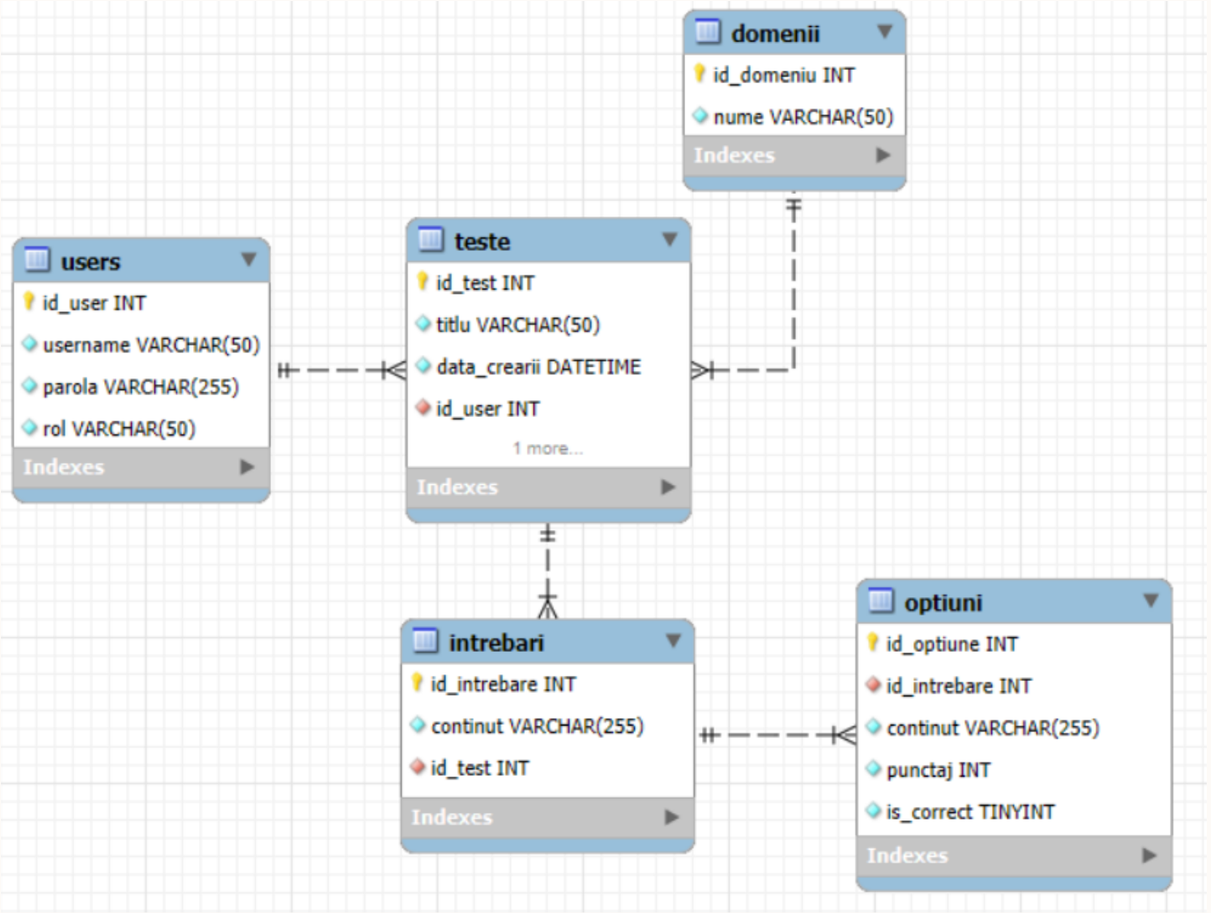
- **MySQL:** Sistem de gestiune a bazei de date relaționale pentru stocarea persistentă a datelor.
- **Postman:** Instrument indispensabil pentru testarea și dezvoltarea API-urilor REST.
- **Git:** Sistem de control al versiunilor, facilitând colaborarea și istoricul modificărilor.

Schema Bazei de Date

Structura: 5 Tabele Relaționale normalizate

Relații: 1-to-Many

Integritate: Integritate referențială cu reguli de Cascade Delete



Implementare - Logica Server

- ## 1. Salvare Teste

  Crearea unui test este o operațiune complexă, gestionată ca o tranzacție **ACID** (@Transactional). Aceasta înseamnă că toate modificările (salvarea testului, întrebărilor și opțiunilor) sunt tratate ca o singură unitate de muncă. Dacă apare o eroare la orice pas, întreaga tranzacție va face **Rollback**, asigurând integritatea datelor.

- ## 2. Securitate JWT

  Sistemul este **Stateless** și se bazează pe **JWT** (JSON Web Token) pentru autentificare. La login se generează un token care este verificat mai apoi la fiecare request folosind un filtru.

- ## 3. Calculare punctaj

  Calculul punctajului se face exclusiv **Server-Side**: clientul trimite doar răspunsurile, iar serverul calculează scorul. Această abordare previne **fraudarea** rezultatelor prin manipularea datelor pe partea clientului.

Monitorizare în Timp Real cu WebSockets

1

1. Conectare Client

Clientul accesează pagina unui test specific, de exemplu, **/teste/1**.

2

2. Abonare Browser

Browserul se abonează automat la topicul WebSocket dedicat, **/topic/test/1**.

3

3. Detectare Eveniment

Serverul detectează un eveniment: un nou utilizator intră în test sau unul existent părăsește testul.

4

4. Broadcast Server

Serverul trimite lista actualizată de participanți către toți clienții abonați la acel topic.

Această funcționalitate este implementată folosind **Spring WebSocket**, **STOMP** (Simple Text Oriented Messaging Protocol) și **SockJS**, asigurând o comunicare bidirecțională și persistentă între client și server.

Concluzii și Direcții Viitoare

Realizări Principale

- Aplicație **funcțională**, **securizată** și **scalabilă**.
- Experiență utilizator **fluidă**, fără reîncărcări inutile ale paginii.
- Cod **modular** și **ușor de întreținut**, respectând bunele practici de programare.

Dezvoltări Ulterioare

- **Statistici Grafice:** Implementarea unor rapoarte detaliate și vizuale pentru creatorii de teste.
- **Export Rezultate:** Funcționalitate de export a rezultatelor testelor în format **PDF**.
- **Timer Test:** Introducerea unei limite de timp pentru susținerea testelor, cu un timer vizibil.

VĂ MULȚUMESC!