Proiect Inginerie Software
Bancă de sânge

Universitatea Tehnică din Cluj-Napoca
Brînzoi Ion-Robert
Grupa 30331

Contents

1 Introducere

Proiectul constă într-o aplicație web care facilitează modalitatea de programare a persoanelor pentru a dona sânge în locațiile disponibile. Avem 3 tipuri de utilizatori:

- administratorul
- doctorii corespunzatori centrelor de donare, care se ocupă cu gestionarea programarilor
- donatorii, care se își fac programări în aplicație

2 Imagine de ansamblu

Administratorul poate:

- să efectueze operații CRUD pe doctori

Doctorul poate:

- vizioneze programările de la centrul său (toate sau cele din ziua curentă)
- să confirme o programare, care implică completarea unui formular cu rezultate
- să vizualizeze rezultatele programărilor confirmate, dar nu le mai poate edita
- să descarce rezultatele în format PDF

Donatorul poate:

- să se înregistreze în aplicație
- să-și editeze datele contului
- să-și steargă contul
- să vizualizeze lista centrelor de donare
- să se programeze la un centru în funcție de zi (nu se ține cont de oră)
- să-si vizualizeze toate programările (anterioare, curente și viitoare)
- să-și anuleze o programare (din ziua curenta sau din viitor)
- să vizualizeze rezultatele programărilor confirmate
- să descarce rezultatele în format PDF (ce conține tabelul și un header cu detalii despre cine a donat, unde, când și cine a confirmat programarea)

2.1 Diagrama de Use Case

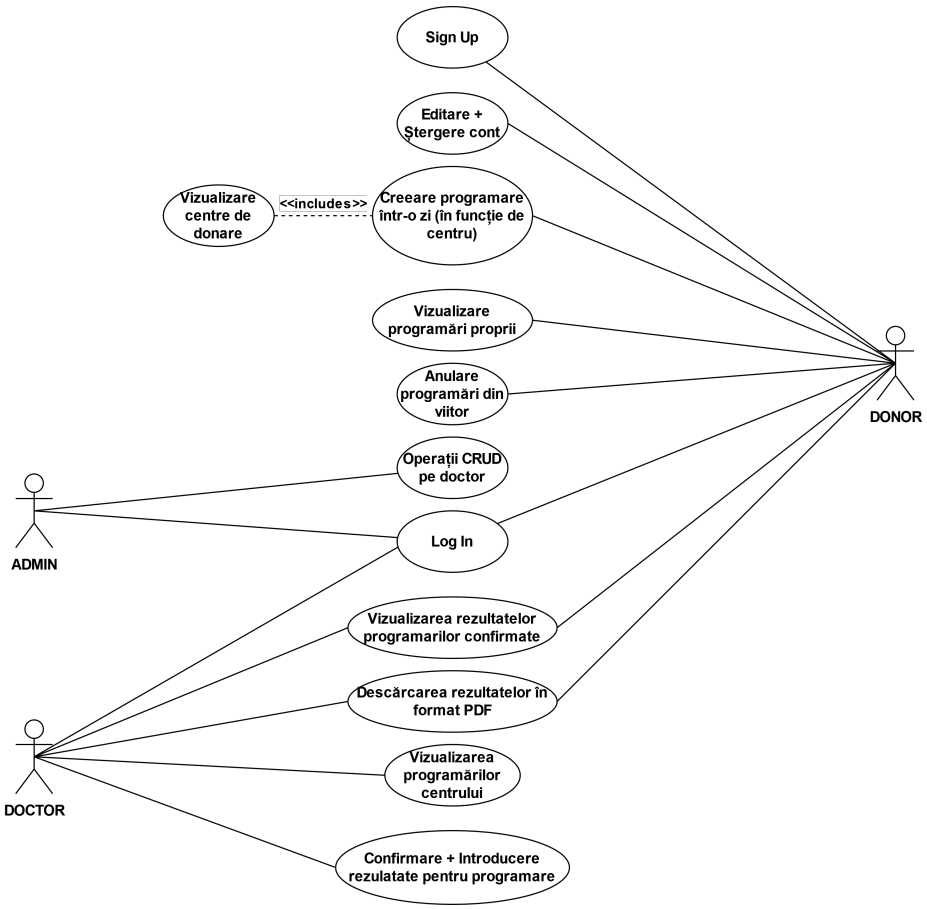

Figure 1: Diagrama de Use Case

3 Structură

3.1 Tehnologii

Partea de frontend a fost realizată folosind librăria React din JavaScript, backend-ul a fost implementat cu Java și Spring, iar pentru baza de date am folosit MySQL.

3.2 Model arhitectural

Modelul arhitectural este unul stratificat (layered architecture).

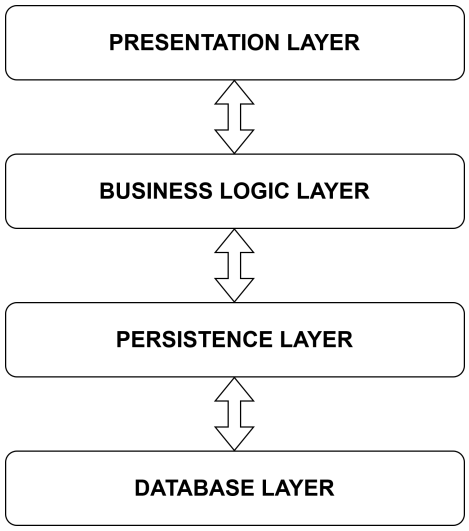

Figure 2: Modelul arhitectural

- **Presentation Layer** - Primul strat al arhitecturii Spring Boot, se ocupă cu gestionarea cererilor HTTP cu ajutorul controllerelor. Se ocupă cu conversia obiectelor primite de tip JSON in DTO-uri (Data Acces Objects) care sunt trimise mai departe la stratul de Business Logic. După returnarea raspunsului de către BL Layer, acesta este convertit la JSON și returnat printr-un ResponseEntity.
- **Business Logic Layer** - Cuprinde toate clasele de service. Acestea sunt responsabile de logica aplicației, validări și mapări (de exemplu dintr-un obiect din model într-un DTO și vice versa)
- **Persistence Layer** - Reprezintă stratul de acces de date, care se ocupă de operațiile CRUD, dar și de paginare (pentru datele mai mari, returnăm o pagină cu un index și un size primite ca și parametru). Acest layer cuprinde interfetele de repository, care extind JPARepository
- **Database Layer** - Conține baza de date, adică totalitatea tabelurilor și relațiile dintre acestea.

3.3 Baza de date

Salvează centrele de donare, programările, raportul asociat unei programări și utilizatorii. Relația de moștenire din backend de către admin, doctor și donor la un user se reflectă aici într-un singur tabel cu toti utilizatorii, care completează câmpurile necesare în funcție de tip (tipul este salvat într-un field dtype).

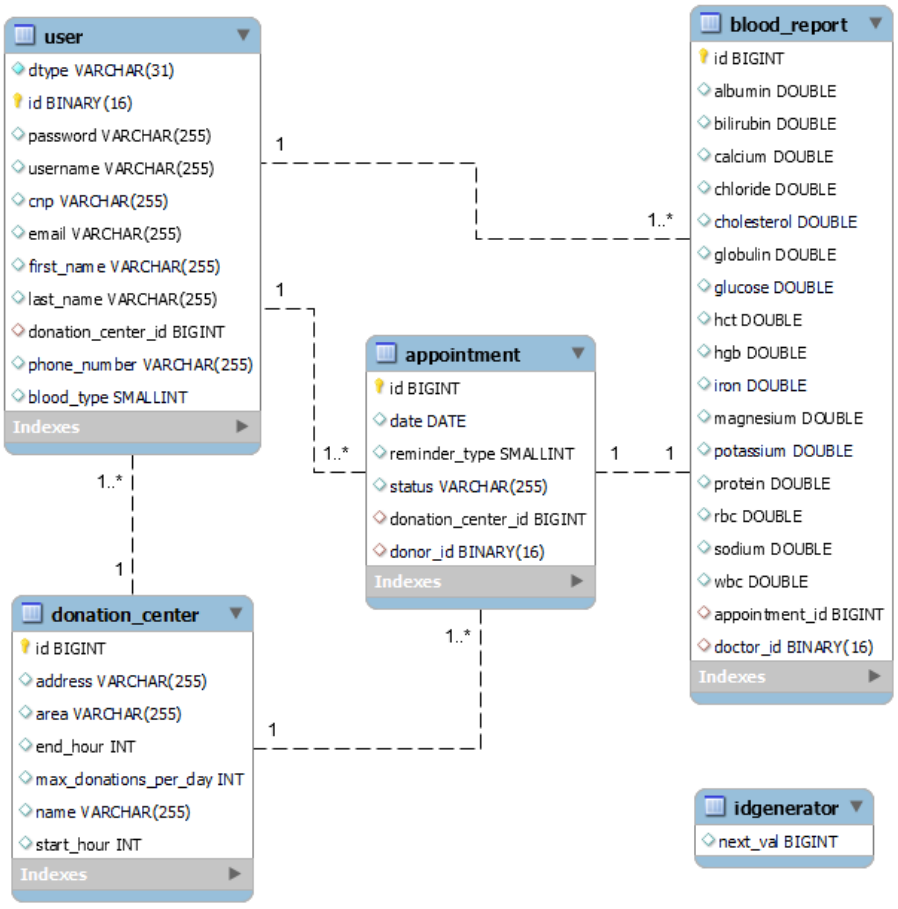

Figure 3: Arhitectura bazei de date

4 Use Cases

4.1 Admin

4.1.1 Editare doctor

După login, adminul va fi intampinat de un homepage și un navigation bar. La selectarea doctorilor, se va afișa un tabel cu aceștia, impreună cu un buton de editare sau ștergere pentru fiecare. In partea dreaptă va fi și un buton de adăugare, acestea permițându-i adminului realizarea operațiilor CRUD.

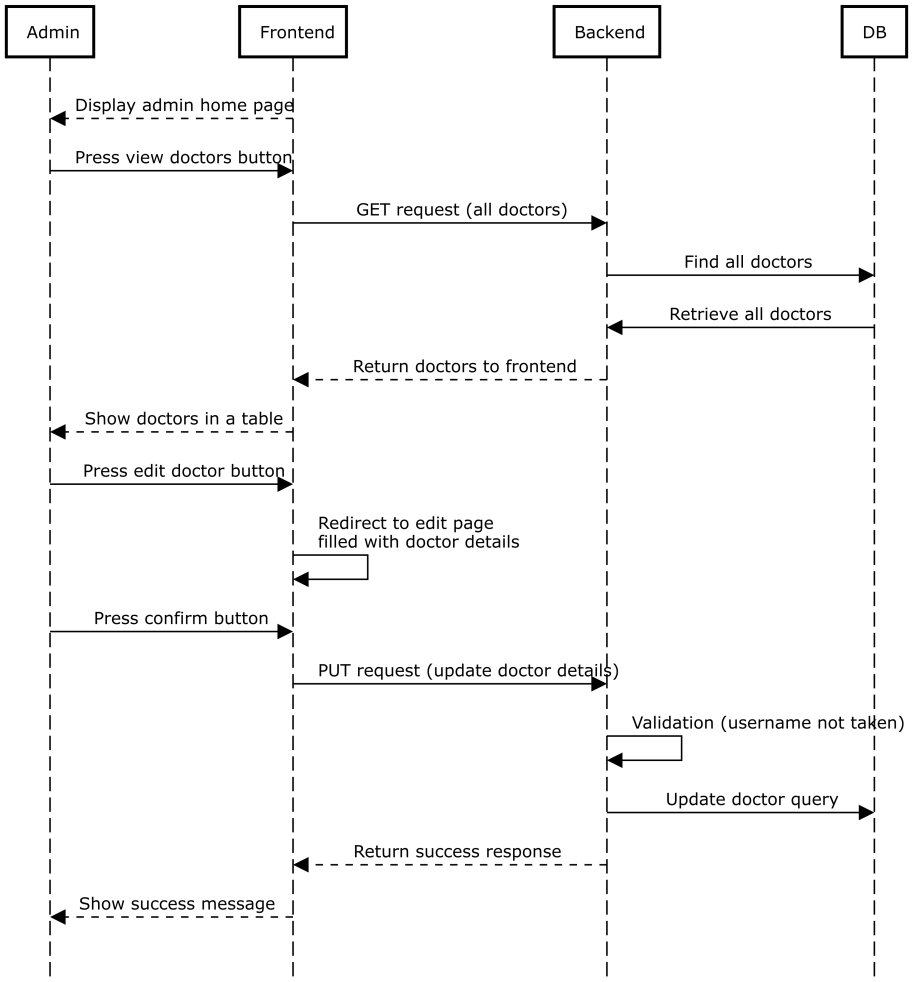

Figure 4: Use case: editare doctor admin

4.2 Donator

4.2.1 Creare programare

După login, din navigation bar donatorul va selecta tab-ul cu centrele de donare, iar acestea vor apărea într-un tabel. Fiecare centru conține detalii despre acesta, precum și un buton de programare pe fiecare linie. La apăsarea butonului respectiv, donatorul va fi redirecționat către pagina de programare. Acolo se va afișa un date picker cu zilele disponibile, și un câmp pentru selectarea tipului de notificare (sms sau email). După apăsarea butonului de confirmare, în backend se vor face anumite validări (dacă data selectată este potrivită și dacă au trecut 6 luni de la ultima programare făcută). Dacă validările trec, se inserează programarea în baza de date, se trimite mesajul și se returnează un mesaj de succes.

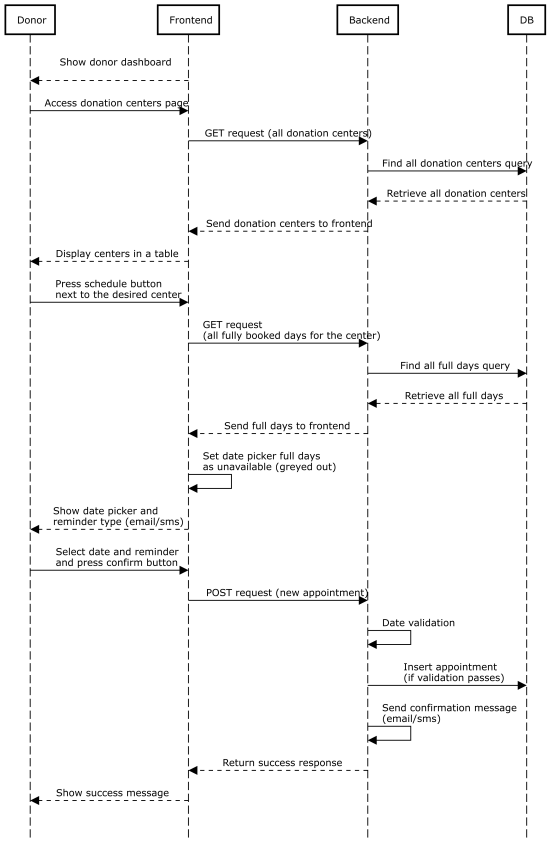

Figure 5: Use case: creare programare pentru donator

4.2.2 Vizualizare programări

Din navigation bar, donatorul poate selecta tab-ul pentru programări, și va fi redirecționat la o nouă fereastră. După ce programările au fost trimise la front end, se va face o verificare a statusului. Dacă este "PENDING", atunci se va afișa un buton de anulare, iar dacă statusul este "CONFIRMED", atunci se va afișa butonul de vizualizare rezultate. Aceste 2 use case-uri vor fi descrise mai jos.

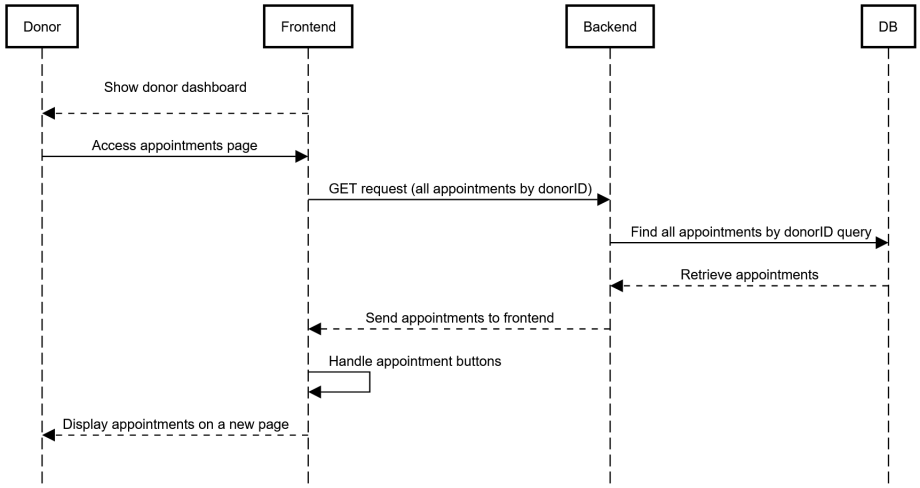

Figure 6: Use case: vizualizare programări pentru donator

4.2.3 Anulare programare

Este posibil ca deși o programare să fie în trecut, să nu fi fost confirmată, de aceea este necesar un validator în service-ul din backend pentru a verifica dacă putem anula într-adevăr programarea. O dezvoltare ulterioară poate consta în adăugarea unui nou status pentru programare (MISSED) și logică în backend astfel încât la finalul zilei, toate programările care nu au fost confirmate să fie considerate MISSED.

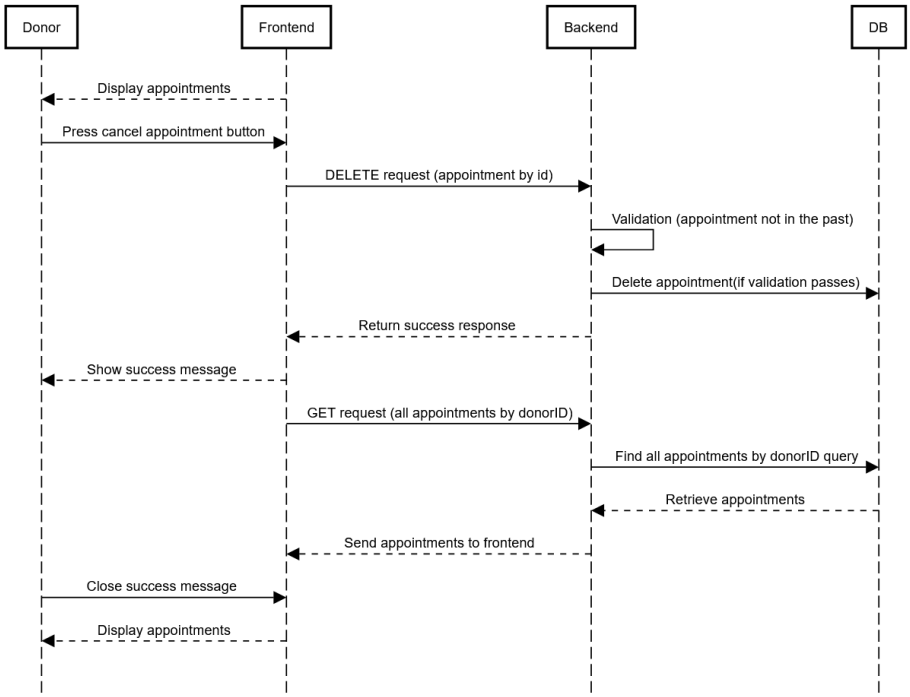

Figure 7: Use case: anulare programare pentru donator

4.2.4 Vizualizare rezultate

La apăsarea butonului de vizualizare a rezultatelor, donatorul va fi redirecționat către pagina cu datele cerute, unde va avea și o opțiune de a le descărca în format pdf, care va conține informații despre programare și rezultatele propriu-zise.

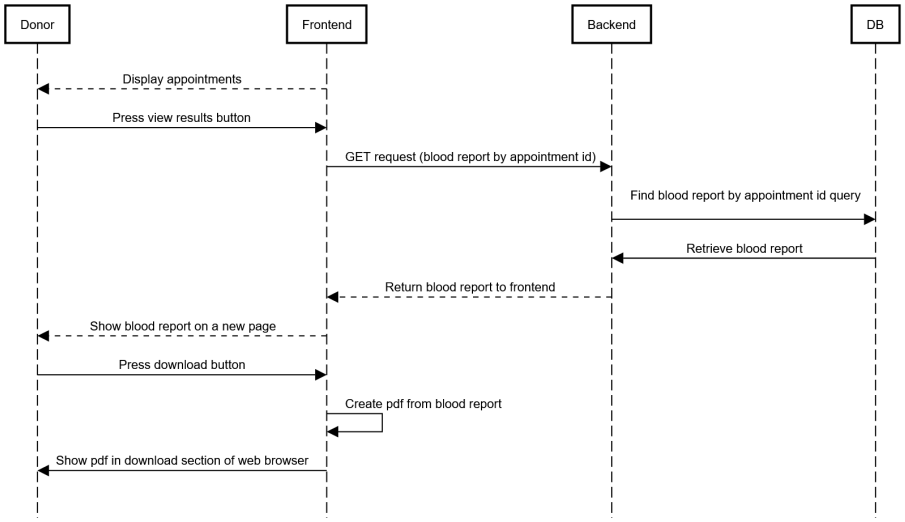

Figure 8: Use case: vizualizare rezultate pentru donator

4.3 Doctor

4.3.1 Confirmare programare

Acest use case va trece mai întâi prin partea de afișare a programărilor din ziua curentă, după care, prin apăsarea butonului de confirmare, se va deschide o nouă fereastră unde doctorul va insera rezultatele. După finalizare, raportul va fi salvat în baza de date iar programarea va fi confirmată.

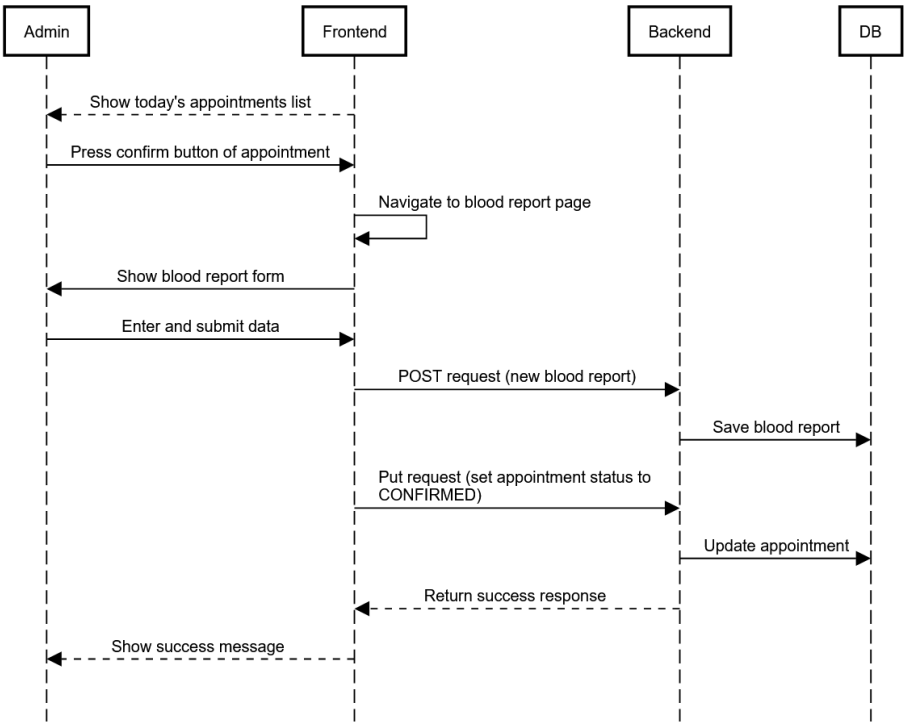

Figure 9: Use case: confirmare programare de doctor

5 Concluzie

Consider că în realizarea aplicației am dobândit o înțelegere mai bună asupra bibliotecii React și a limbajului JavaScript. Totodată, am aprofundat aspecte din OOP, am folosit Design Patterns și am învățat despre modalitatea de comunicare dintre frontend și backend, precum și concepte esențiale despre RESTful Api. Acest proiect a fost primul pas important făcut în cadrul proiectării unei aplicații web.