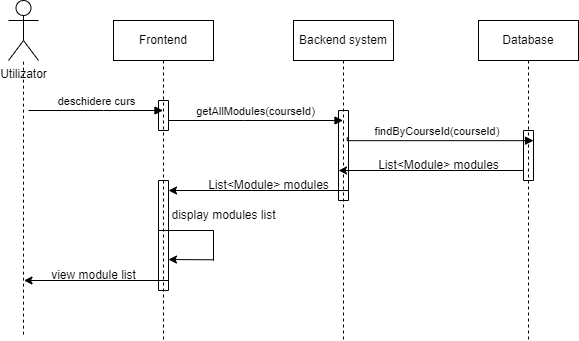
Interacțiunile dintre utilizator, front-end, back-end și managerul de autentificare sunt ilustrate în diagrama de secvență a fluxului de autentificare din figura X. Utilizatorul completează câmpurile (email și parolă) și apasă pe submit. Din front-end se trimite o cerere de tip POST către back-end. Back-end-ul folosește Authentication Manager pentru a verifica credențialele în baza de date.

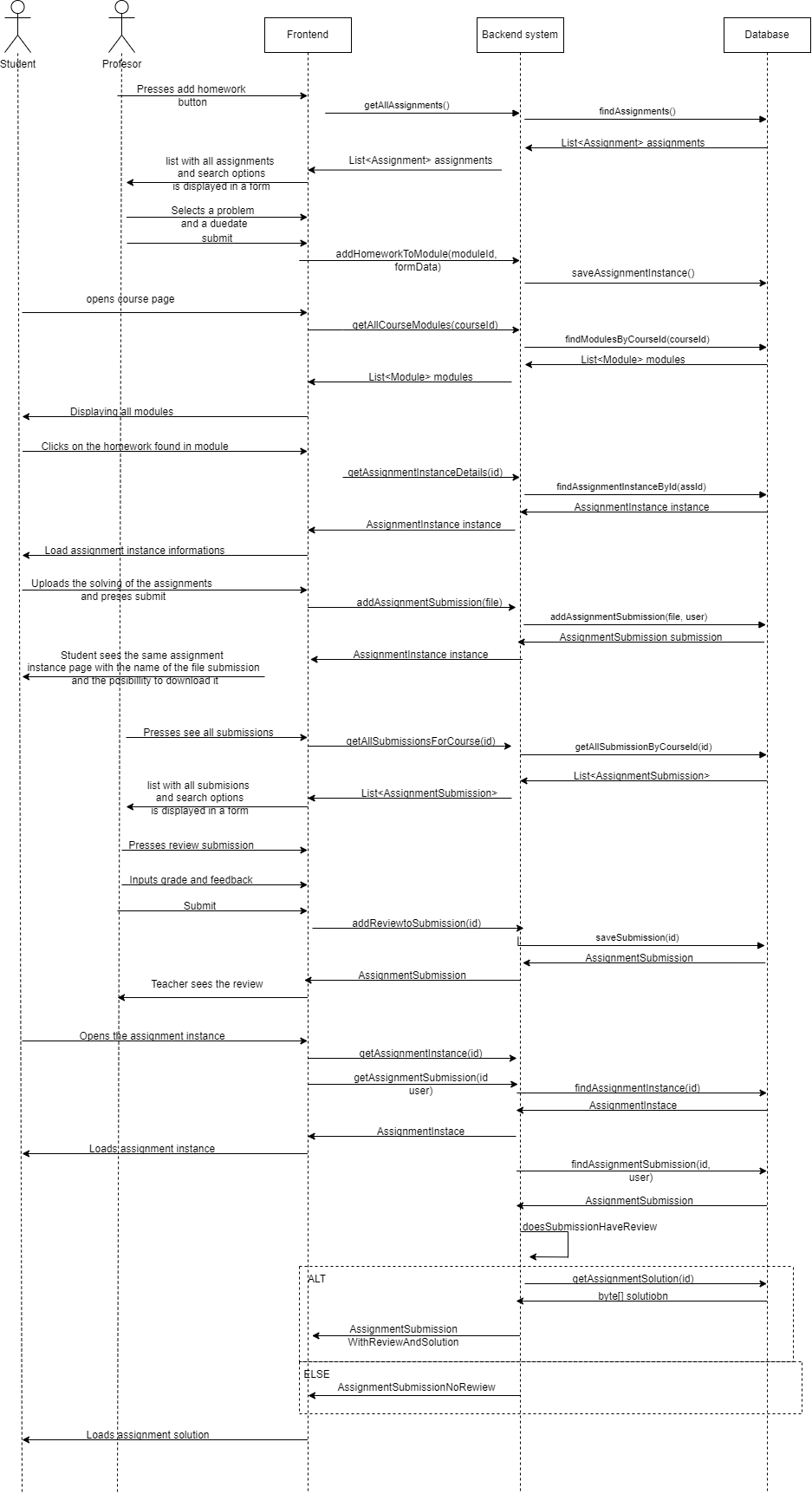
Dacă credențialele sunt corecte, se întoarce un obiect de autentificare. În back-end se generează un token JWT care conține proprietățile utilizatorului, cum ar fi rolul și emailul. Front-end-ul redirecționează utilizatorul pe pagina specifică rolului acestuia: pentru student și profesor pe pagina principală unde sunt afișate cursurile, iar pentru admin pe pagina specială pentru administrator si tokenul este salvat in local storage-ul browserului.

Dacă credențialele sunt incorecte, se întoarce un mesaj HTTP "Bad credentials", iar utilizatorului i se afișează un mesaj de eroare.



Interacțiunile dintre utilizator, front-end, back-end și baza de date sunt ilustrate în diagrama de secvență a fluxului de accesare a unui curs din figura X. Utilizatorul deschide cursul pentru a vizualiza lista de module disponibile. Din front-end se trimite o cerere de tip GET către back-end. Back-end-ul folosește baza de date pentru a găsi modulele asociate cursului specificat.

Dacă cererea este validă, back-end-ul interoghează baza de date pentru a găsi toate modulele asociate cursului cu ID-ul specificat. Baza de date returnează o listă de module către back-end. Back-end-ul trimite lista de module către front-end, care o afișează utilizatorului.

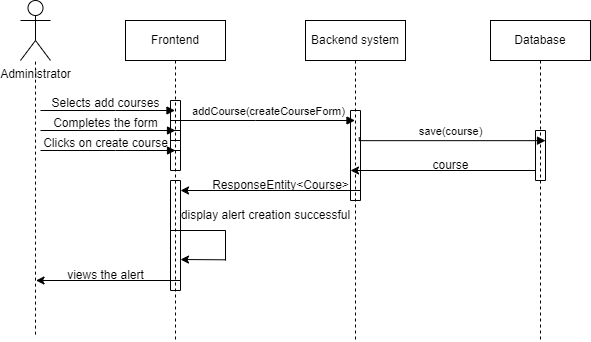


In diagrama din figura x este prezentata interactiunea a diferitelor roluri, student si profesor, cu sistemul. Profesorul este in pagina de curs, acesta apasa pe butonul add homework aflat in unul dintre module. Ii este afisat un formular in care are un modul de cautare si toate problemele existente in lista de exercitii. Acesta alege unul dintre exercitii dupa care selecteaza un termen limita pentru incarcarea temei si apasa pe butonul add homework, pagina de curs este updatata, fiind vizibila noua tema propusa.

Studentul deschide pagina de curs, vede tema propusa de profesor intr-un modul acesta apasa click pe tema. O noua pagina se deschide. Aceasta contine informatiile temei. Elevul termina tema si se decide sa o incarce. Apasa pe butonul upload homework alege fisierul si da submit.

Profesorul apasă pe butonul „See all submisions”, astfel acesta este redirectat pe o pagina care conține toate încărcările studenților. Apasă pe butonul de review, se deschide o pagina cu formular de unde acesta poate descarcă ceea ce a încărcat studentul. Acesta completează recenzie si apasă pe butonul de submit.

Studentul deschide tema, acum este disponibila recenzia profesorului, nota cat si rezolvarea corecta a temei. Daca nu ar fi fost încă adăugată o recenzie acesta avea vizibil decât ultima lui încărcare pe care acesta putea sa o descarce si sa vadă ce se afla.



Doar utilizatorii cu rol de administrator pot crea cursuri. Acestia cand se autentifica sunt redirectionati automat catre o pagina la care doar administratorii au acces. De altfel pentru adaugarea unui curs completeaza informatiile cursului cum ar fi denumirea, categoria si descrierea si apasa pe butonul de creeaza. Se face o cerere de tip POST catre backend prin care se trimit informatiile completate in formular. Se salveaza in baza de date, iar apoi se introarce un obiect de tip ResponseEntity cu STATUS 201 si cursul care a fost adaugat. Apoi utilizatorul vede in aplicatie un mesaj „Cursul a fost adaugat cu succes”.

### Accesul si interactiunea cu baza de date

In solutia mea am ales sa folosesc JPA impreuna cu hibernate. JPA fiind o solutie de a controla accesul la baza de date iar hibernate implementarea acestuia. Aceasta abstractizeaza toata complexitatea implementarii in spatele unor interfete si reguli bine definite.

JPA si hibernate sunt un mod de programare bazat pe Object-Relational Mapping (ORM)acesta fiind un mod de a alinia codul cu baza de date. Se structureaza foarte bine datorita modului de programare orientat pe obiecte, fiecare clasa din codul java este mapata la un tabel.

Am ales sa folosesc acest model la acces de date fata de un model ca MyBatis pentru optimizarea si abstractizarea pe care o aduce jpa si hibernate acestea avand incorporat caching.

Cu toate ca MyBatis ofera mai mult control asupra interogarilor sql, am decis ca automatizarea ORM este de ajutor pentru mine datorita faptului ca mentinerea interogarilor este mult mai usor de facilitat.

### Securitate

Pentru partea de securitate folosesc Spring Security si json-web token pentru autentificare si autorizarea diferitelor roluri din aplicatie. Odata ce userul se autentifica acestuia ii este generat un jwt token care este trimis din frontend inapoi, in fiecare cerere pe care acesta o face, pentru a confirma ca acesta este insusi userul pentru care tokenul a fost generat.

Explicație Detaliată

Autentificare: Utilizatorul introduce credențialele (de exemplu, nume de utilizator și parolă) și trimite o cerere de autentificare către server.

Serverul verifică credențialele utilizatorului. Dacă autentificarea este reușită, se generează un token JWT care poate contine informatii despre utilizator si rolurile acestuia. Aceste informatii se numesc claims.

Tokenul JWT este semnat folosind o cheie secretă pentru a asigura integritatea și autenticitatea acestuia.

Tokenul JWT generat este trimis către frontend și stocat de obicei in memoria locala a browserului sau un cookies.

Autorizare:

La fiecare cerere ulterioara din frontend este trimis alaturi de informatiile cererii in antet tokenul. Serverul verifica tokenul asupra validitatii, daca tokenul este valid acesta permite cererea altfel nu trimitand o eroare.