UNIVERSITATEA BABEŞ-BOLYAI

Facultatea de Ştiinţe Economice şi Gestiunea Afacerilor

Informatică Economică

Lucrare de licenţă

Absolvent,

Dragoș **ILIEȘ**

Coordonator ştiinţific,

Lect. univ. dr. Cristian **BOLOGA**

2024

UNIVERSITATEA BABEŞ-BOLYAI

Facultatea de Ştiinţe Economice şi Gestiunea Afacerilor

Informatică Economică

Lucrare de licenţă

Aplicație web implementată în Node.js cu bază de date MySQL pentru gestionarea accesului angajaților în sălile de fitness

Absolvent,

Dragoș **ILIEȘ**

Coordonator ştiinţific,

Lector univ. dr. Cristian **BOLOGA**

2024

Rezumat

Scopul principal al acestei lucrări este dezvoltarea unei aplicații web destinată angajaților unei corporații de nivel internațional. Cu ajutorul acestei aplicații se gestionează accesul angajaților în sălile de fitness de care dispune o companie. Platforma web va contribui la administrarea numărul de persoane permise simultan într-o locație, în funcție de capacitatea acesteia. Utilizatorii își vor crea un cont cu ajutorul adresei de email corporative și datele personale. Apoi se vor autentifica pe platformă, unde pot vizualiza sălile de fitness puse la dispoziție și pot rezerva un anumit interval orar, în locația aleasă. Rezervările pentru un anumit interval orar vor fi condiționate de capacitatea sălilor de fitness. În procesul de dezvoltarea al aplicației, am folosit Node.js cu framework-ul Express pentru partea de back-end. Pentru front-end am utilizat framework-ul React. De asemenea, în MySQL am stocat baza de date care cuprinde informații despre utilizatori, săli de fitness, intervale orare și rezervări. Soluția poate fi utilizată și pentru alte facilități ale companiei, demonstrând aplicabilitatea și adaptabilitatea în contextul managementului resurselor corporative.

Cuprins

Lista tabelelor și figurilor vi
Introducere 1
1. Identificarea și descrierea problemei 4
1.1 Motivația 6
1.2 Context 9
1.2.1 Fațeta subiect 9
1.2.2 Fațeta utilizare 9
1.2.3 Fațeta IT 10
1.2.4 Fațeta dezvoltare 11
2. Cerințe de sistem 13
2.1 Surse de cerințe 13
2.2 Elicitația cerințelor 13
2.2.1 Metoda interviului 14
2.2.2 Metoda brainstormingului 15
2.2.3 Modelul use-case 18
2.3 Documentarea cerințelor 19
3. Model de dezvoltare 21
4. Proiectarea logică 23
4.1 Procese și activități 23
4.2 Managementul datelor 29
5. Proiectarea tehnică 32
5.1 Arhitectura sistemului 32
5.2 Diagrama de deployment 33
5.3 Diagrama de componente 35
6. Implementarea 37
6.1 Configurarea conexiunii la baza de date 37
6.2 Modele (Models) 38
6.3 Controlere (Controllers) 44
6.4 Scripturi (Scripts) 49
6.5 Interfața utilizatorului (Pages) 51
7. Testarea 57

7.1 Verificarea funcționalităților back-end cu Postman 57

7.2 Testarea interfeței cu Selenium 63

8. Dezvoltări viitoare ale aplicației 73
Concluzii 74
Bibliografie 75
Anexe 76

Lista tabelelor și figurilor

Tabele:

Tabel 1 - Actori, Roluri și Obiective 14

Figuri:

Figura 1 – Business Model Canvas 5
Figura 2 – Diagrama Fishbone (cauză – efect) 7
Figura 3 – Diagrama de motivație Archimate (descompunerea obiectivelor) 8
Figura 4 – Diagrama Pareto 17
Figura 5 – Modelul use-case 18
Figura 6 - Diagrama de activități As-Is 24
Figura 7 - Diagrama de activități To-Be din perspectiva utilizatorilor finali ai aplicației 25
Figura 8 - Diagrama de activități To-Be din perspectiva administratorilor sălilor de fitness/platformei 26
Figura 9 - Diagrama de stări 28
Figura 10 - Diagrama ERD 29
Figura 11 - Diagrama de deployment 34
Figura 12 - Diagrama de componente 36

Introducere

În zilele noastre, există tot mai mulți factori care influențează negativ starea de spirit sau sănătatea angajaților dintr-o companie. Printre acești factori se numără: mediul de lucru stresant, colectivul neprietenos, strictețea superiorilor sau monotonia sarcinilor. Acestea sunt câteva din cauzele care reduc motivația și implicit productivitatea angajaților la locul de muncă. Pentru combaterea acestor probleme, companiile de nivel internațional au pus accent pe dezvoltarea unei ambianțe atractive și prietenoase, la locul de muncă. Astfel, o practică abordată este aceea de a investi în programe de fitness destinate angajaților.

Programele de fitness încurajează o cultură a sănătății la locul de muncă. Conform unui studiu realizat de Marín-Farrona, Wipfli, Thosar, Colino, Garcia-Unanue, Gallardo, Felipe și López-Fernández (2023), programele de wellness la locul de muncă bazate pe activitate fizică pot îmbunătăți semnificativ sănătatea și productivitatea angajaților. Tematica acestei lucrări de licența este realizarea unei aplicații web care să faciliteze gestionarea accesului angajaților în sălile de fitness din cadrul unei companii.

Fitness-ul este una dintre cele mai bune modalități de a îmbunătății sănătatea și bunăstarea angajaților. Conform unui blog publicat pe site-ul oficial al SanoPass, câteva dintre efectele pozitive pe care implementare unor programe de acest tip le au, sunt:

> „Îmbunătățirea sănătății fizice: exercițiile fizice regulate reduc riscul de boli cronice, precum bolile de inimă, diabetul și obezitatea, ceea ce poate reduce absenteismul la locul de muncă. Angajații care fac exerciții fizice regulate au un sistem imunitar mai puternic, ceea ce îi ajută să rămână sănătoși și să prevină bolile care pot afecta performanța la locul de muncă.
>
> Îmbunătățirea sănătății mentale: exercițiile fizice regulate ajută la reducerea stresului și a anxietății, contribuind la îmbunătățirea sănătății mentale a angajaților. Un studiu de la Harvard a arătat că exercițiile fizice regulate pot avea un efect similar cu cel al medicamentelor antidepresive în cazul persoanelor cu depresie ușoară sau moderată. Angajații care se simt mai fericiți și mai echilibrați mental sunt mai productivi la locul de muncă.
>
> Creșterea energiei și a concentrării: exercițiile fizice regulate ajută la creșterea nivelului de energie și a capacității de concentrare a angajaților. Fluxul crescut de sânge și oxigen către creier poate îmbunătăți nivelul de energie și poate

crește capacitatea de concentrare. Angajații care se simt energici și alerți sunt mai eficienți la locul de muncă.

Îmbunătățirea interacțiunii sociale: activitățile fizice pot fi o modalitate excelentă de a construi relații și de a interacționa cu colegii de muncă într-un mod diferit față de cel din timpul lucrului. Antrenamentele în grup pot îmbunătăți interacțiunile sociale și pot construi o comunitate mai puternică la locul de muncă. Acest lucru poate duce la o comunicare mai bună și la o mai bună colaborare între angajați, ceea ce poate îmbunătăți productivitatea și performanța la locul de muncă."

În plus, un articol publicat în Corporate Wellness Magazine subliniază alte beneficii ale implementării programelor de fitness la locul de muncă:

„Gestionarea mai eficientă a timpului și creșterea eficienței: angajații care participă la programele de fitness de la locul de muncă pot dezvolta abilități mai bune de gestionare a timpului și eficiență, deoarece învață să echilibreze responsabilitățile de serviciu cu rutinele lor de exerciții fizice.

Creșterea loialității și angajamentului angajaților: investirea în bunăstarea angajaților trimite un mesaj puternic prin care compania își valorizează angajații și bunăstarea acestora. Acest lucru poate duce la creșterea loialității și angajamentului din partea angajaților, care sunt mai probabil să rămână într-o companie care prioritizează sănătatea și fericirea lor. Ratele reduse de fluctuație a personalului nu doar economisesc bani pe recrutare și formare, dar contribuie și la o forță de muncă mai stabilă și experimentată.

Atracția și retenția talentelor: pe măsură ce piața muncii devine din ce în ce mai competitivă, companiile trebuie să găsească modalități de a se diferenția pentru a atrage și păstra talentele de top. Oferirea de programe și facilități de fitness la locul de muncă este o modalitate eficientă de a îmbunătăți reputația companiei ca angajator de top. Căutătorii de locuri de muncă sunt mai predispuși să ia în considerare companiile care demonstrează un angajament față de bunăstarea angajaților."

Această lucrare prezintă modul în care a fost dezvoltată aplicația web pentru a îndeplini nevoile angajaților unei corporații. Prin intermediul platformei, angajații își pot planifica intervalele de timp pentru fiecare antrenament, în funcție de locația aleasă, după ce în prealabil și-au creat un cont. Principalul obiectiv al lucrării este acela de a implementa o

interfață prietenoasă și intuitivă pentru utilizator care să faciliteze modalitatea de rezervare a unor intervale orare. Prin acest mod, se evită supraaglomerarea și se respectă capacitatea sălilor. Accentul este pus pe design-ul software și implementarea practică a aplicației web. Astfel, în cadrul procesului de dezvoltare a fost folosit framework-ul Express, specific Node.js, pentru partea de back-end. Pentu partea de front-end, framework-ul React a reprezentat unealta principală în modelarea aspectului user-friendly. Baza de date a aplicației este împărțită în 4 tabele specifice și este stocată pe un serve local cu ajutorul Xampp și MySQL.

Lucrarea este împărțită în câteva capitole esențiale după cum urmează: capitolul 1 reprezintă identificarea problemei și motivația care a determinat implementarea acestei aplicații. Capitolul 2 se axează pe beneficiarii platformei și tehnicile utilizate pentru constatarea cerințelor, în timp ce capitolul 3 scoate în evidență modelul de dezvoltare ales pentru implementarea sistemului. Capitolele 4 și 5 pun accent pe proiectarea sistemului, mai exact proiectarea logică, respectiv proiectarea tehnică. Apoi, în capitolul 6 este prezentată implementarea funcționalităților esențiale, iar capitolul 7 este destinat metodologiilor de testare folosite. În capitolul 8 voi vorbi despre ideile viitoare de dezvoltare ale aplicației. În final, sunt expuse concluziile lucrării. Materialele suport sunt incluse în zona Anexelor.

1. Identificarea și descrierea problemei

Fitness-ul este una dintre ramurile sportului care s-a aflat într-o continuă creștere în ultima perioadă. Principalul motiv este reprezentat de conștientizarea oamenilor despre importanța abordării unui stil de viață sănătos. După cum se știe, mediul corporativ modern este unul plin de obstacole și provocări care pot influența într-un mod negativ sănătatea oamenilor. Anxietatea, stresul, monotonia sau presiunea șefilor sunt câțiva factori resimțiți de către angajați care pot duce la scăderea productivității sau la unele probleme de sănătate. Astfel, angajatorii au început să conștientizeze importanța unui mediu de lucru sănătos în cadrul firmelor care să asigure un randament cât mai ridicat din partea personalului. În acest sens, companiile de nivel internațional încep să integreze programe de fitness și wellness în facilitățile de lucru, destinate exclusiv propriilor angajați.

Studiului realizat de Marín-Farrona et al. (2023) sugerează că programele de wellness la locul de muncă bazate pe activitate fizică pot îmbunătăți semnificativ sănătatea și productivitatea angajaților. Studiul a arătat că angajații care participă la astfel de programe au raportat o reducere a oboselii și o îmbunătățire a stării de bine generale. În plus, s-a observat o creștere a productivității și o scădere a absenteismului, ceea ce se traduce prin beneficii semnificative atât pentru angajați, cât și pentru angajatori.

Platforma pe care am implementat-o vine în răspuns la nevoile angajaților pentru a facilita accesul acestora la programele și sălile de fitness. Principala problemă este lipsa unei aplicații care să gestioneze participarea eficientă pentru a se evita supraaglomerarea. Majoritatea persoanelor preferă să facă sport într-un colectiv cât mai restrâns pentru a nu se simți inconfortabil în prezența prea multor persoane. În plus, nimănui nu îi place să aștepte până se eliberează un anumit aparat la care dorește să lucreze. Astfel, prezența mai redusă ar rezolva această problemă. De asemenea, programul angajaților dintr-o multinațională este destul de diversificat. În acest context, platforma trebuie să fie cât mai bine personalizată pentru cerințele utilizatorilor, pentru a asigura folosirea sălilor de fitness la capacitate maximă.

Un sistem automatizat care gestionează accesul angajaților la facilitățile oferite de compania în care activează ar rezolva toate aceste probleme.

Figura 1 – Business Model Canvas

1.1 Motivația

Principala sursă de motivație provine din analizarea unei situații de business în cadrul companiei unde lucrează tatăl meu. Compania este una de nivel internațional, cu sedii în țări de pe toate continentele. În ultima perioadă, managementul companiei s-a axat pe crearea unui mediu cât mai plăcut pentru proprii angajați. Una dintre preocupările principale a fost construirea unor săli de fitness, în cât mai multe dintre locații, destinate exclusiv pentru cei care activează în firmă. Compania a înțeles că pentru a crește productivitatea, sunt nevoiți să pună accent pe sănătatea angajaților. Astfel, le-au pus la dispoziție facilitățile necesare pentru a duce un stil de viață sănătos și pentru a veni cât mai motivați la locul de muncă.

Cu toate acestea, implementarea unei platforme care să gestioneze accesul angajaților în sălile de fitness este esențială pentru un bun mers al lucrurilor. Angajații trebuie să aibă posibilitatea să aleagă o locație și un interval orar în funcție de nevoile lor. Datorită faptului că sălile de fitness au o capacitate maximă recomandată, rezervarea unui loc este necesară pentru a se evita supraaglomerarea și situațiile în care trebuie să aștepți eliberarea unui aparat. Prin acest mod, întreg procesul de rezervare este automatizat și centralizat. Se elimină necesitatea de a gestiona manual programările, pe o foaie sau într-un fișier Microsoft Excel, ceea ce reduce erorile umane și crește eficiența administrativă.

Mai mult de cât atât, administratorii sălilor de fitness vor putea gestiona mai bine întreg procesul cu ajutorul aplicației. Pot să modifice intervalele orare existente în cazul în care apar unele probleme de mentenanță. Datele colectate prin intermediul platformei oferă o perspectivă amănunțită cu privire la preferințele și comportamentul angajaților legate de utilizarea sălilor de fitness. În acest fel, compania se poate asigura de o îmbunătățire continuă a facilităților în funcție de nevoile angajaților.

Prin urmare, dezvoltarea unei platforme online care permite angajaților să facă rezervări online este o investiție în infrastructura fizică a companiei și capitalul uman. Compania va beneficia pe termen lung pentru atragerea, sănătatea și productivitatea personalului.

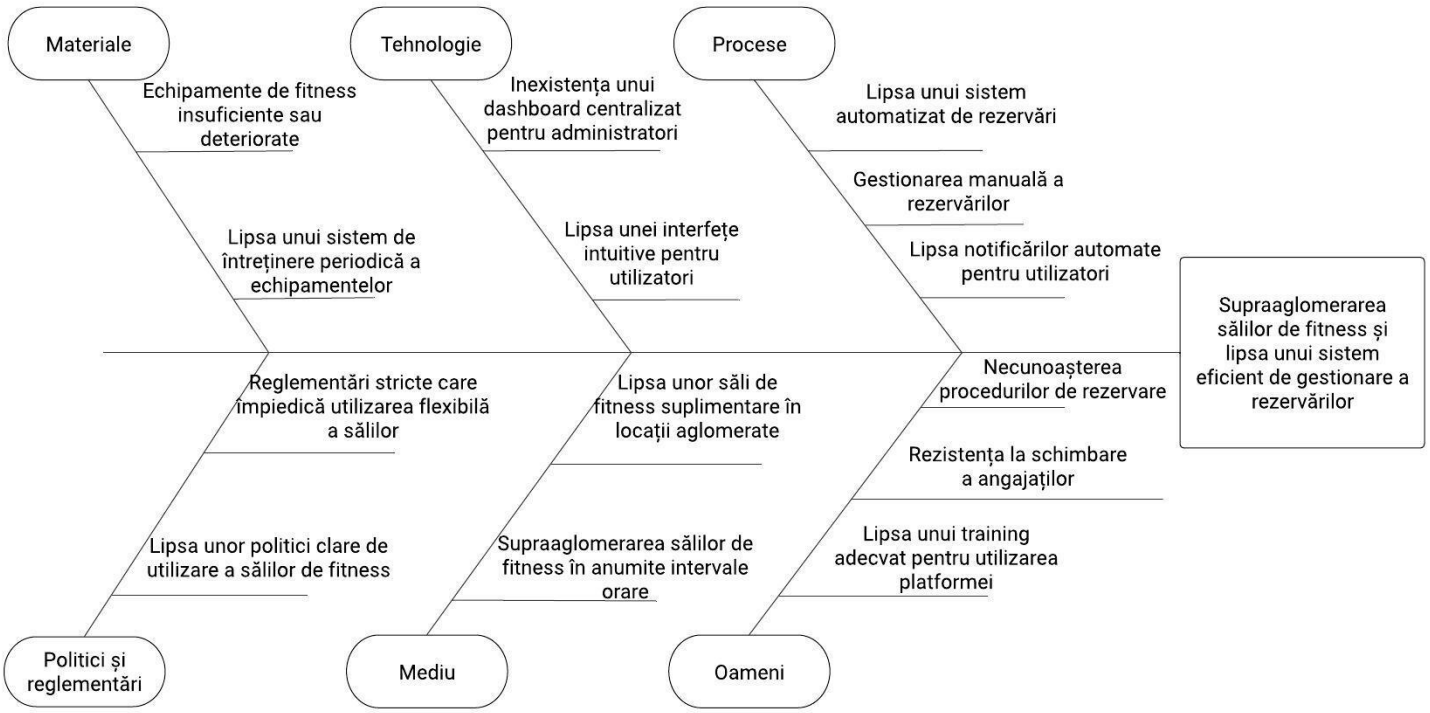


Figura 2 – Diagrama Fishbone (cauză – efect)

Cu ajutorul unei diagrame de tipul cauză-efect (Fishbone) am încercat să identific principalele probleme care contribuie la necesitatea dezvoltării unei platforme web de gestiune a rezervărilor în sălile de fitness. Din Figura 2 se poate desprinde cauza generală care a dus la nevoia unei astfel de aplicații. De obicei, oamenii sunt mai reticenți la unele schimbări, însă implementarea unui sistem automatizat al rezervărilor ar îmbunătăți viața multor persoane. Interfața aplicației va fi una cât mai intuitivă pentru a asigura o experiență plăcută a utilizatorilor și pentru a-i îndemna să folosească platforma. De asemenea, va avea loc o sesiune în care se va explica în detaliu modul în care funcționează aplicația și se vor prezenta toate funcționalitățile disponibile.

Astfel, prin implementarea aplicației web de gestiune a rezervărilor se va urmări rezolvarea a cât mai multor probleme prezente în Figura 2.

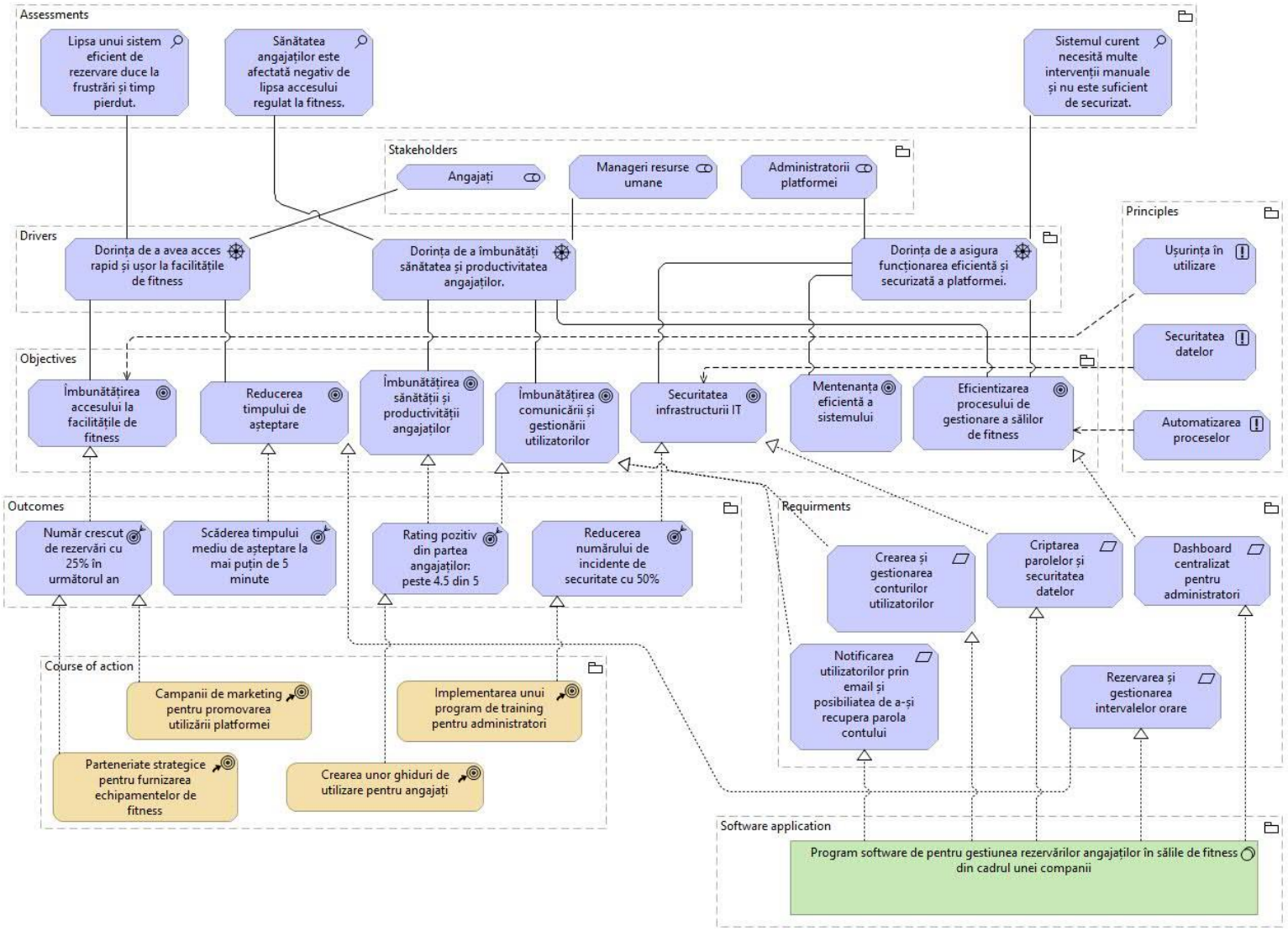


Figura 3 – Diagrama de motivație Archimate (descompunerea obiectivelor)

Pasul următor a fost cel de stabilire a obiectivelor. Figura 3 oferă o perspectivă clară asupra relațiilor dintre actorii implicați și propriile dorințe. De asemenea, am surprins obiectivele și indicatorii acestora, care reies din dorințele stakeholderilor. Am evidențiat principiile, cerințele și acțiunile necesare pentru realizarea proiectului. Se poate observa organizarea componentelor pe diferite niveluri pentru a avea o imagine de ansamblu care va fi de ajutor în procesul de analiză a datelor.

1.2 Context

Companiile se confruntă, în prezent, cu provocări continue în ceea ce privește îmbunătățirea sănătății angajaților și eficientizarea utilizării resurselor pe care le au la dispoziție. Investirea într-o platformă web dedicată gestionării sălilor de fitness din cadrul organizației are un dublu beneficiu: încurajarea angajaților înspre adoptarea unui stil de viață sănătos și maximizarea folosirii facilităților de fitness de care dispune compania. Acest aspect subliniază tendința globală de creștere a productivității prin investirea în sănătatea și satisfacția angajaților.

În continuare, voi transpune cele 4 dimensiuni fundamentale pe care aplicația le urmărește: fațeta subiect, fațeta utilizare, fațeta IT și fațeta dezvoltare.

1.2.1 Fațeta subiect

Fațeta subiect include dispunerea și tratarea obiectivelor tangibile sau intangibile, precum și a evenimentelor relevante pentru această platforma online de gestiune accesului în sălile de fitness.

Scopul aplicației este acela de a furniza o interfață intuitivă și accesibilă pentru toți utilizatorii. Prin această platformă, se asigură un flux continuu și se evită aglomerația în cadrul facilităților de fitness.

Astfel, aplicația oferă angajaților o pagină principală în care sunt expuse toate sălile de fitness din diferitele locații ale companiei. În fiecare sală de fitness este expusă locația, urmată de o descriere succintă și capacitatea de care dispune. Sălile de fitness au atribuite intervale de timp din oră în oră, iar utilizatorii pot să rezerve un astfel de interval de timp. Apoi, pot vizualiza rezervarea făcută și eventual să o anuleze dacă este cazul.

Platforma are și o secțiune dedicată utilizatorilor cu rol de administrator în care sunt stocate informațiile legate de utilizatori, săli de fitness, intervale orare și rezervări. Această secțiune facilitează procesul de gestionare a sălilor de fitness printr-o interfață ușor de utilizat. Aici sunt afișate toate informațiile stocate în baza de date, cu posibilitatea de adăugare, ștergere sau editare a datelor.

1.2.2 Fațeta utilizare

Utilizatorii platformei sunt cei care lucrează în cadrul companiei, fiecare cu nevoi și obiective diferite în ceea ce privește utilizarea aplicației web. Persoanele care pot interacționa cu acest

sistem se încadrează în 3 categorii: vizitatori (cei care nu dețin încă un cont), membri (cei care și-au deschis un cont pe platforma folosind adresa de email) și administratorii.

Utilizatorii vizitatori sunt angajați ai companiei care nu s-au înregistrat încă pe platformă. Ei au posibilitatea de a citi o scurtă descriere a aplicației și de a se informa cu privire la antrenorii personali prezenți în sălile de fitness, precum și clasele și programele susținute de aceștia. De asemenea, există o rubrică dedicată întrebărilor frecvente care apar în rândul celor care folosesc platforma. În finalul paginii, este prezentă o galerie cu poze pentru ca persoanele care accesează aplicația să își poată forma o imagine despre aspectul și facilitățile de care dispun sălile de fitness. Aceștia au posibilitatea să își deschidă un cont prin accesarea paginii de „SignUp" și completarea unui formular cu datele lor personale. După înregistrare, pot avea acces la toate beneficiile platformei.

Cei care și-au creat deja un cont dispun de acces nelimitat în cadrul aplicației. Acești utilizatori pot să aleagă una dintre sălile de fitness disponibile în funcție de locație. Apoi, au posibilitatea să consulte intervalele de timp disponibile și să își rezerve un loc. Precedent pasului de confirmare, angajații pot consulta detaliile rezervării pentru a fi convinși că au completat datele conform propriilor dorințe. În plus, fiecare utilizator are opțiunea de a-și vizualiza următoarea rezervare sau de a anula rezervarea în cazul în care au apărut anumite modificări în programul personal. Utilizatorii nu pot face mai mult de o rezervare, până în momentul expirării rezervării aflate în curs. Totodată, cei care folosesc aplicația din postura de membru, pot să își modifice datele personale cum ar fi: email-ul, numele complet sau parola, în cazul în care au fost introduse greșit.

Nu în ultimul rând, utilizatorii cu rol de administrator sunt redirecționați către o interfață grafică, unde pot să urmărească informațiile care sunt stocate în baza de date a platformei. De asemenea, au permisiunea de a adăuga, edita sau șterge informații din tabelele prezente în baza de date.

1.2.3 Fațeta IT

Fațeta IT a aplicației web cuprinde totalitatea tehnologiilor implicate în dezvoltarea platformei destinate gestionării accesului angajaților în sălile de fitness. Această componentă a proiectului a fost esențială pentru a asigura o experiență fluidă și eficientă utilizatorilor.

Partea de back-end a aplicației a fost creată prin folosirea Node.js care este runtime de JavaScript orientat spre evenimente. Prin integrarea unui astfel de server web rapid și scalabil, am încercat să răspund cât mai bine la nevoile utilizatorilor. Cererile HTTP sunt o componentă cheie în cadrul unei aplicații web. Pentru a putea simplifica structura rutelor și

pentru a organiza mai bine componentele de back-end am decis să folosesc framework-ul Express. Un serviciu important care m-a ajutat în implementarea corectă a cererilor HTTP este Postman. Acesta a facilitat testarea, verificarea și documentarea API-urilor înainte de a trece la etapa de integrare cu interfața utilizatorului. În plus, în componenta de back-end, am implementat un script care să asigure fluiditate la nivelul bazei de date. Acest script verifică dacă există intervale orare pentru toate sălile de fitness, pentru următoarele 3 luni iar, în caz contrar, le generează. De asemenea, pentru a mă asigura că baza de date să nu fie mult prea încărcată, am programat acest script sa ruleze în fiecare zi de luni a săptămânii cu ajutorul Task Scheduler din sistemul meu de operare.

Baza de date a platformei este găzduită de către MySQL, un sistem de management al bazelor de date relaționale pe care l-am utilizat prin intermediul Xampp. Am încercat să proiectez și să stochez cât mai bine datele necesare unei astfel de aplicații prin distribuirea lor în tabele specifice care să gestioneze informații despre utilizatori, săli de fitness, intervale orare și rezervări.

Interfața grafică interactivă, destinată utilizatorilor, am dezvoltat-o folosind framework-ul React. Această bibliotecă pune un accent important pe reutilizabilitate și are abilitatea de a gestiona starea aplicației într-un mod dinamic. Cu ajutorul React, am reușit să implementez partea de front-end într-un mod eficient care să nu necesite reîncărcări ale paginii în momentul în care sunt depistate anumite schimbări de stare. Pentru a testa interfața aplicației am folosit Selenium.

Un serviciu extern pe care l-am folosit este SendGrid cu ajutorul căruia pot trimite email-uri pentru resetarea parolelor. Acest serviciu mi-a generat o cheie de API pe care o folosesc în partea de back-end pentru a trimite mail-ul de resetare al parolei. Prin implementarea funcționalității de resetare a parolei, utilizatorii care doresc să acceseze platforma dar și-au uitat parola, au posibilitatea să primească pe email un link de resetare a parolei. După ce aceștia accesează link-ul primit pe email, vor fi redirecționați către o pagină nouă care conține un formular de resetare al parolei.

Mediul de dezvoltare al aplicației web a fost Visual Studio Code, un mediu care a oferit un set de unelte avansate pentru scrierea, testarea și depanarea codului într-un mod eficient.

1.2.4 Fațeta dezvoltare

Aplicația web prezentă a fost dezvoltată pentru a satisface cerințele angajatorilor de a pune la dispoziția angajaților un sistem simplu și ușor de accesat. Pentru procesul de dezvoltare software am adoptat modelul Scrum. Aceasta abordare este specifică unui model de tip Agile

care mi-a permis să ajustez rapid funcționalitățile pe baza feedback-ului primit din partea clientului (tatăl meu, care mi-a expus necesitatea unei astfel de platforme în compania în care activează) și coordonatorului meu. Prin faptul că am organizat întreaga activitate în sprinturi de una sau doua săptămâni, am reușit să îmbunătățesc în mod continuu aplicația și să mențin flexibilitatea pe întreg parcursul. Astfel, săptămânal am integrat, testat și evaluat câte o nouă funcționalitate, iar apoi am gândit noi sarcini pentru viitor.

Metodologia specifică acestei abordări este realizarea unor ședințe zilnice cu membri echipei de dezvoltare în cadrul cărora sunt dezbătute mai multe subiecte cum ar fi: progresul realizat până în momentul de față, următoarele sarcini de lucru și probleme care s-ar putea să apară pe parcurs. De asemenea, la finele oricărui sprint este evaluată munca efectuată și se planifică următorul sprint cu obiectivele specifice. Având în vedere faptul că eu am fost singura persoană care a asigurat dezvoltarea acestei platforme, întâlnirile specifice metodei Scrum nu au avut loc.

Dezvoltările viitoare ale aplicației constă în extinderea funcționalităților pentru a se putea face o analiză amănunțită cu privire la comportamentul utilizatorilor. Acest proces ar putea include generarea unor rapoarte pentru managerii de resurse umane. Prin analiza acestor rapoarte se pot implementa noi funcționalități prin care să se ofere angajaților anumite recomandări personalizate. O altă îmbunătățire posibilă ar fi integrarea notificărilor prin email cu privire la rezervările angajaților. De asemenea, s-ar putea lua în calcul implementarea unor algoritmi de machine learning care să sugereze cele mai eficiente programe de antrenament și cei mai buni antrenori în funcție de obiectivele personale ale angajaților. Nu în ultimul rând, optimizarea infrastructurii de back-end, odată cu creșterea numărului de utilizatori și a volumului de date, este esențială.

2. Cerințe de sistem

În acest capitol, voi prezenta cerințele necesare pentru dezvoltarea și implementarea aplicației web care gestionează accesul angajaților în sălile de fitness. Această parte a proiectului cuprinde mai multe aspecte importante cum ar fi: sursele de cerințe și metodologiile de elicitație folosite pentru a colecta date și informații relevante. De asemenea, voi pune accent și pe documentarea cerințelor. Aspectele de mai sus reprezintă o componentă importantă în crearea unei aplicații eficiente, care să aducă satisfacție atât angajaților, cât și angajatorului.

2.1 Surse de cerințe

În contextul aplicației web pe care am dezvoltat-o, sursele de cerințe provin în urma unei discuții pe care am avut-o cu tatăl meu. El mi-a expus o problemă pe care a întâmpinat-o în compania în care lucrează. Această companie s-a axat, în ultima perioadă, pe construirea unor facilități destinate propriilor angajați, pentru a încerca să le stimuleze productivitatea. Cu toate acestea, ca să se asigure utilizarea la capacitate maximă a acestor facilități, ar fi necesară implementarea unui sistem online de gestiune a rezervărilor din toate locațiile. După o scurtă analiză făcută pe baza nevoilor companiei și a feedback-ului utilizatorilor vizați, am identificat următoarele surse de cerințe:

- Angajații companiei: principalii utilizatori ai platformei, care au nevoie de un sistem ușor de utilizat și adaptat propriilor nevoi, pentru a-și putea rezerva intervalul orar dorit în locația dorită
- Administratorii platformei: persoanele responsabile de gestiunea sălilor de fitness, care au nevoie de un sistem bine pus la punct care să eficientizeze și să maximizeze modul în care sunt utilizate aceste facilități de către angajații firmei
- Managerii de resurse umane: cei care se ocupă de personalul companiei și încearcă să asigure o ambianță plăcută pentru angajați.

2.2 Elicitația cerințelor

Procesul de elicitație a cerințelor, a contribuit la identificarea cerințelor de sistem. Prin interacțiunile pe care le-am avut cu actorii implicați în proces, am înțeles exact nevoile acestora . Pentru a prezenta într-o manieră sistematică rolurile și obiectivele tuturor actorilor, am organizat informațiile într-un tabel.

Actori	Roluri	Obiective
Angajații	Utilizatorii finali pentru care a fost realizată aplicația, astfel încât să își poată rezerva intervalul orar disponibil în locația dorită.	Planificarea antrenamentelor în funcție de preferințe. Evitarea supraaglomerării în sălile de fitness ale companiei.
Administratorii platformei	Responsabilii de gestionarea platformei și a sălilor de fitness.	Asigurarea funcționării optime a aplicației, actualizarea informațiilor legate de sălile de fitness, conturile de utilizatori și rezervările acestora. Semnalarea eventualelor probleme tehnice.
Managerii de resurse umane	Persoanele care supraveghează utilizarea sălilor de fitness de către angajați și analizează datele statistice pentru a face o evaluare asupra impactului programelor de wellness.	Monitorizarea și optimizarea utilizării facilităților companiei. Îmbunătățirea programelor de wellness pentru a satisface nevoile tuturor angajaților.

Tabel 1 - Actori, Roluri și Obiective

În vederea realizării acestei etape, am folosit mai multe metode care mi-au permis să colectez informațiile importante legate de toți actorii implicați. Astfel, metodele utilizate sunt:

- Interviul
- Brainstorming
- Modelul use-case

Toate aceste metode enumerate mai sus au avut un rol important în înțelegerea exactă a cerințelor de sistem. Aplicația web prezentată în această lucrare are ca scop principal gestiunea rezervărilor angajaților în sălile de fitness ale unei companii. Însă, din Tabelul 1 se poate observa prezența mai multor actori în cadrul acestui proces. În consecință, interacțiunea cu toate părțile implicate este fundamentală pentru a extrage toate informațiile necesare dezvoltării platformei.

2.2.1 Metoda interviului

Interviul este o tehnică de elicitație directă, prin care am încercat să identific și să înțeleg principalele cerințe ale sistemului, din perspectiva utilizatorilor, administratorilor, managerilor de resurse umane. În cadrul acestor interviuri, am abordat atât subiecte mai generale, urmate de întrebări mai specifice pentru a aprofunda subiectul. Am optat pentru

intervievarea persoanelor din cele trei categorii relevante de actori. Din fiecare categorie au participat un anumit număr de persoane, mai exact am stat de vorbă cu trei persoane din categoria managerilor de resurse uman. Dintre cei care se vor ocupa de administrarea platformei au fost prezenți la interviu trei persoane, în timp ce, dintre angajații din celelalte departamente, am avut ocazia să iau legătura cu zece persoane.

Printr-un set de întrebări concise, am reușit să extrag anumite cerințe și probleme, formulate din perspectiva tuturor persoanelor implicate în acest proces. Pentru prezentarea interviului și a concluziilor finale, am ales răspunsurile cele mai elaborate. Acestea pot fi găsite în Anexa 1, unde am expus întrebările și răspunsurile din cadrul interviului.

Din răspunsurile pe care le-am primit, se pot trage anumite concluzii. Astfel, aproape toți angajații frecventează sălile de fitness de cel puțin trei ori pe săptămână. Acest lucru subliniază nevoia unui sistem care să gestioneze eficient fluxul de persoane. De asemenea, din cauza programului de lucru similar pentru majoritatea clienților, o bună parte optează pentru aceeași perioadă a zilei în care să își efectueze antrenamentul, ceea ce duce la supraaglomerarea sălilor. Angajații s-au exprimat, totodată, în legătură cu implementarea unor funcționalități care să notifice utilizatorii cu privire la datele rezervării în curs.

Pe de altă parte, administratorii platformei s-au axat pe necesitatea unei interfețe cât mai intuitive pentru utilizatori. Implementarea unui dashboard centralizat care să gestioneze întreaga bază de date și securitatea datelor utilizatorilor sunt necesare. Aceștia au nevoie să gestioneze cât mai ușor platforma și sălile de fitness.

Cerința managerilor de resurse umane a fost întocmirea unor rapoarte generate automat. Aceste rapoarte ar putea ajuta managerii să aibă o viziune de ansamblu asupra aspectelor pozitive odată cu înființarea sălilor de fitness. Analiza datelor contribuie la evaluarea și îmbunătățirea programelor de wellness.

Aceste interviuri m-au ajutat să înțeleg mai bine nevoile și așteptările utilizatorilor finali, dar și cerințele și eventualele sugestii de îmbunătățire venite din partea managerilor de resurse umane și administratorilor platformei.

2.2.2 Metoda brainstormingului

Următoarea metodă pe care am aplicat-o în procesul de elicitație a cerințelor a fost metoda brainstormingului. După cum evidenția și Dr. Gina Carmen Goleșteanu într-un articol:

> „Brainstorming-ul este o tehnică ce se folosește pentru stimularea gândirii creatoare în grup. Este o metodă interactivă care se bazează pe dezbaterea unei

probleme, prin emiterea liberă și spontană a opiniilor, indiferent cât de hazardate sunt. Scopul acesteia este de a înlesni găsirea celei mai adecvate soluții, printr-o intensă mobilizare a ideilor tuturor participanților la discuție. Este un excelent exercițiu de cultivare a creativității în grup și stimulează inventivitatea participanților."

Astfel, am organizat o sesiunea de brainstorming la sediul firmei în care lucrează tatăl meu. Această sesiune a avut ca scop discutarea principalelor nevoi pe care platforma web trebuie să le îndeplinească, din perspectiva tuturor actorilor. Aplicația web pentru gestionare a rezervărilor în sălile de fitness va ușura viața multor persoane din cadrul companiei. Din acest motiv e important ca fiecare dintre ei să își poată exprima propriile idei, dar și să asculte opiniile celorlalți. În cadrul sesiunii de brainstorming, am stabilit obiective clare și am încurajat persoanele participante să vină cu cât mai multe idei. Sesiunea a fost structurată pentru a permite colectarea și organizarea ideilor. Apoi am combinat și filtrat cele mai importante soluții.

Sesiunea de brainstorming a decurs astfel:

1. Pregătirea: am stabilit grupul de participanți format din angajați ai oricărui departament, cei care vor urma să administreze platforma și managerii de resurse umane. Locația aleasă a fost sediul firmei, mai exact sala de mese pentru a avea la dispoziție un spațiu suficient de generos. Data și ora au fost stabilite de comun acord, pentru a asigura o participare cât mai numeroasă. Am pregătit materialele necesare cum ar fi hârtii, pixuri, notițe adezive, etc. Apoi am prezentat scopul acestei întâlniri și am explicat regulile brainstormingului, inclusiv interzicerea criticilor.

2. Colectarea ideilor: fiecare participant a notat câte o idee legată de funcționalitățile necesare platformei, iar apoi ideile au fost atașate pe o tablă magnetică. După această primă rundă de prezentare, participanții au venit cu anumite completări, stimulați probabil de ideile prezentate anterior.

3. Adaptarea ideilor: în această etapă am grupat ideile care erau asemănătoare și am eliminat ideile care erau puțin prea fanteziste. Principalele idei s-au referit la implementarea unei interfețe ușor de înțeles de către toți utilizatorii, realizarea unui serviciu de notificare pentru rezervări, securizarea datelor și crearea unui dashboard pentru administratori.

4. Structurarea ideilor: Ideile clasificate anterior au fost sistematizate în pagini și funcționalități ale aplicației, prin care am încercat să răspund cât mai multor nevoi venite din

partea tuturor actorilor. Astfel, aplicația va conține o pagină în care sunt prezentate toate sălile de fitness. Aici utilizatorii pot alege în ce locație să își rezerve un antrenament. De asemenea, va fi posibilă vizualizarea rezervării curente, vizualizarea și modificarea datelor de profil. Va fi inclus un serviciu pentru resetarea parolei. Pentru partea de administrator, se va asigura securitatea datelor și se va permite accesul la baza de date centralizată.

Tehnica brainstormingului a fost de mare ajutor pentru a identifica nevoile pe care trebuie să le îndeplinească aplicația web. Printr-o astfel de discuție deschisă, în care au participat persoane din toate categoriile relevante, a fost mai ușor să stabilim cerințele și scopul platformei, din perspectiva tuturor persoanelor implicate.

În urma aplicării celor doua metode de elicitație a cerințelor de sistem, mai exact metoda interviului și brainstormingul, am reușit să extrag și să sintetizez anumite concluzii legate de necesitatea unei aplicații de gestiune a rezervărilor în sălile de fitness. Cunoașterea acestor probleme m-a ajutat în implementarea unei platforme care să răspundă cerințelor de sistem. Pentru a vizualiza problemele identificate, am realizat o diagramă Pareto. După cum se poate observa în Figura 3, pe axa orizontală sunt identificate problemele. Pe axa verticală din stânga este prezent numărul de apariții pentru fiecare din aceste probleme. Corelarea între ele s-a făcut cu ajutorul unui grafic de tip bară. În acest context, linia roșie este reprezentarea procentului cumulativ al problemelor. Aceste concluzii vor ajuta la dezvoltarea ulterioară a platformei și la implementarea funcționalităților cerute.

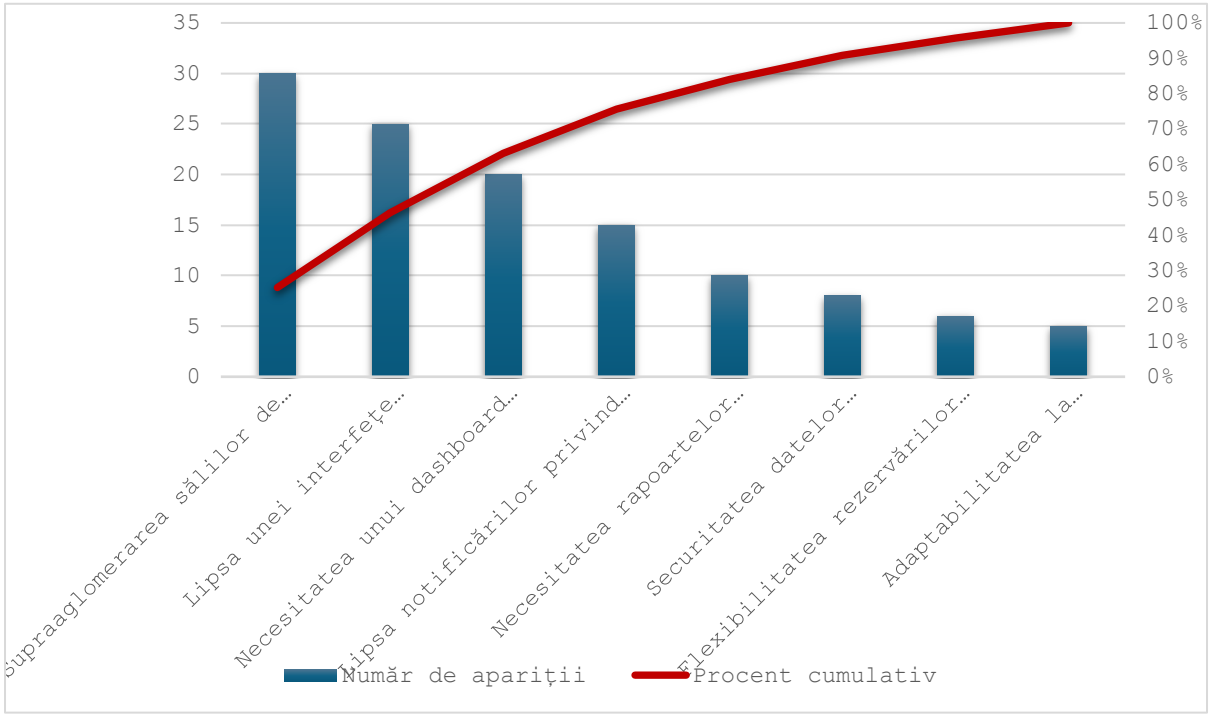


Figura 4 – Diagrama Pareto

2.2.3 Modelul use-case

Modelul cazurilor de utilizare este un model de diagramă specific limbajului UML. Această diagramă surprinde interacțiunile dintre principalele grupe de actori menționate în Tabelul 1 și funcționalitățile platformei. Cu toate acestea, în Figura 5 se poate observa lipsa managerilor de resurse umane ca actori. Funcționalitatea care corespunde acestei categorii de actori este reprezentată de întocmirea unor rapoarte de analiză. Aceste rapoarte au rolul de a ajuta managerii de resurse umane să facă o analiză obiectivă asupra impactului pe care programele de fitness l-au avut. Funcționalitatea în cauză nu a fost implementată, momentan, în aplicație, dar voi discuta mai pe larg de acest subiect în capitolul de idei viitoare de dezvoltare ale platformei.

Astfel, în Figura 5, am creionat cele trei tipuri de utilizatori ai aplicației, mai exact: angajații care nu și-au deschis un cont momentan și doar vizualizează pagina de prezentare, angajații care și-au deschis cont și au acces la funcționalitățile aplicației și administratorii care se ocupă de gestionarea bazei de date și buna funcționare a aplicației. Use-case-urile reprezintă acțiunile propriu-zise pe care le poate realiza platforma la cererea actorilor, iar săgețile reprezintă interacțiunea dintre actori și use-case sau dintre doua use-case-uri.

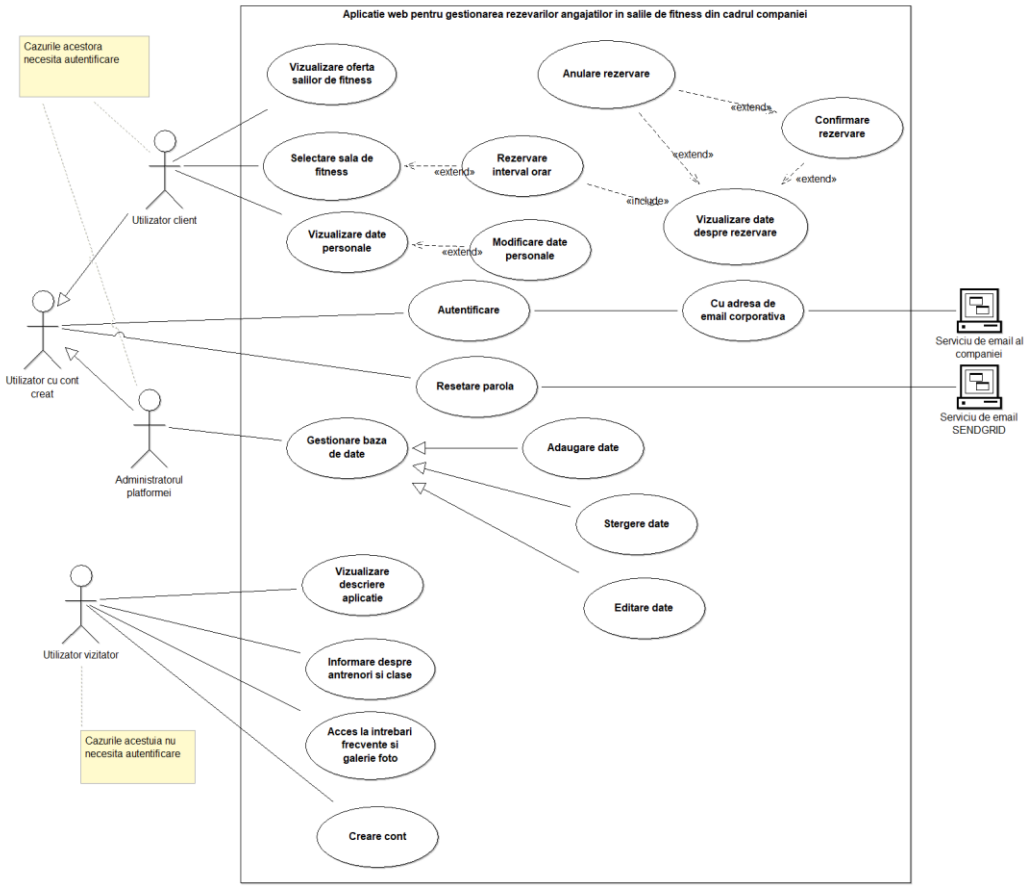


Figura 5 – Modelul use-case

2.3 Documentarea cerințelor

Procesul de elicitație a cerințelor a fost realizat prin metoda interviului, a brainstormingului și a modelului use-case. Cu ajutorul acestor metode, am identificat obiectivele actorilor implicați și cerințele la care trebuie să răspundă aplicația. Următoarea etapă este documentarea cerințelor. Pentru a vizualiza mai ușor cerințele, le-am grupat în patru categorii distincte: cerințe de sistem, cerințe funcționale, cerințe non-funcționale și cerințe calitative.

1. Cerințele de sistem:

- Memorie RAM: Minim 4GB RAM, preferabil 8GB RAM pe calculatorul care va găzdui aplicația.
- Procesor: Minim un procesor dual-core, preferabil un procesor quad-core pentru calculatorul care va găzdui aplicația.
- Sistem de management al bazelor de date: Instalare Xampp pentru MySQL server pe calculatorul care va găzdui aplicația.
- Platformă de dezvoltare: Instalare Node.js pe calculator, minim versiunea 14.
- Sistem de operare: Un sistem de operare Windows, MacOS sau Linux, preferabil Windows 10 sau o versiune mai nouă.
- Conexiune la internet: Obligatorie pentru calculatorul care găzduiește aplicația și pentru utilizatori.
- Browser web: Instalarea unui browser pe dispozitivul utilizatorului, preferabil Chrome, Firefox sau Microsoft Edge.

2. Cerințe funcționale:

- Autentificare și înregistrare: Utilizatorii trebuie să se poată înregistra cu adresa de email corporativă și să se autentifice pentru a accesa platforma. De asemenea, trebuie să aibă posibilitatea de a-și reseta parola în cazul în care și-au uitat parola curentă.
- Rezervarea sălilor: Utilizatorii trebuie să poată vizualiza sălile de fitness disponibile, să selecteze o locație și să rezerve un interval orar.
- Notificări: Utilizatorii vor primi notificări prin email cu privire la rezervările făcute.
- Administrarea sălilor: Administratorii trebuie să poată adăuga, edita și șterge informațiile din baza de date.
- Raportare: Managerii de resurse umane trebuie să aibă acces la rapoarte generate automat care să ofere informații despre utilizarea sălilor de fitness și impactul programelor de wellness.

3. Cerințe non-funcționale:

- Timp de răspuns: Aplicația trebuie să răspundă la cererile utilizatorilor în maxim 2 secunde.
- Standard de dezvoltare: Codul sursă al aplicației trebuie să respecte standardele de codare impuse de comunitatea JavaScript și React.
- Termen de finalizare: Aplicație trebuie să fie finalizată până în data de 20 iunie 2024.

4. Cerințe calitative:

- Intuitivitate: Interfața grafică trebuie să fie intuitivă, facilitând utilizatorilor navigarea și realizarea de rezervări fără dificultăți.
- Fiabilitate: Sistemul trebuie să fie stabil și să funcționeze fără întreruperi sau erori.
- Utilizabilitate: Aplicația trebuie să fie ușor de utilizat pentru toate categoriile de utilizatori, inclusiv pentru cei fără cunoștințe tehnice avansate.
- Documentație: Trebuie să existe documentație completă pentru utilizatori și administratori, explicând toate funcționalitățile aplicației.
- Asistență: Trebuie să fie disponibil un serviciu de suport pentru utilizatori pentru a răspunde la întrebări și a rezolva problemele tehnice.

3. Model de dezvoltare

În dezvoltarea aplicației de gestiune a rezervărilor în sălile de fitness am utilizat metodologia Scrum. Această metodologie a fost pentru prima dată menționată de către Hirotaka Takeuchi și Ikujiro Nonaka în articolul „The New Product Development Game" (1986, p. 137). Ideea principală pe care acești autori au scos-o în evidență este aceea că echipele mici, care reușesc să transmită sarcinile de la una la alta, vor avea cele mai bune rezultate.

Prin diferitele metode de elicitație, menționate în capitolul anterior, am reușit să extrag și să grupez cerințele de sistem. Interacțiunea cu beneficiarii finali a fost esențială pentru a înțelege nevoile la care trebuie să răspundă aplicația. În acest caz, beneficiarii finali sunt persoanele angajate în cadrul firmei în care lucrează tatăl meu.

În contextul aplicației pe care am implementat-o, metodologia Scrum a fost cea mai potrivită prin prisma faptului că timpul de dezvoltare a fost unul relativ scurt. De asemenea, fiind singurul dezvoltator din cadrul proiectului, a fost mai ușor să îmi planific sprint-uri de una sau două săptămâni pentru realizarea funcționalităților necesare. Sprint-urile au fost planificate în colaborare cu tatăl meu, cel care m-a ajutat să interacționez cu utilizatorii finali ai aplicației. Împreună cu el am stabilit principalele funcționalități pe care aplicația trebuie să le conțină.

Astfel, sprint-urile au constat în identificarea entităților și proiectarea bazei de date. Mai apoi, am dezvoltat funcționalitățile de login și logout pentru utilizatori. Tot în acest pas, a trebuit să diferențiez nivelurile de acces ale aplicației, prin atribuirea de roluri diferite pentru persoanele care vor administra platforma și utilizatorii obișnuiți. De asemenea, am implementat un serviciu de recuperare a parolei pentru toți utilizatorii. În continuare, m-am ocupat de principala funcționalitate a aplicației, mai exact selectarea sălii de fitness dorite, alegerea datei calendaristice și a intervalului orar, iar apoi confirmarea rezervării. În plus, am adăugat câteva servicii suplimentare precum anularea rezervării curente sau editarea datelor personale.

La finalul fiecărui sprint, făceam o analiză alături de profesorul coordonator și tatăl meu asupra funcționalităților implementate . În această etapă, mă asiguram de îndeplinirea tuturor obiectivelor și evidențiam sarcinile care nu au fost finalizate, precum și anumite erori întâlnite pentru a nu le repeta. Revizuirea sprint-urilor anterioare m-a ajutat în planificarea următoarelor sarcini. Astfel, proiectul s-a dezvoltat treptat, prin adăugarea săptămânală a noi îmbunătățiri.

Majoritatea uneltelor folosite în dezvoltarea aplicației au fost descrise în subcapitolul Fațeta IT. Totuși, există unele tool-uri care nu au contribuit la procesul de implementarea al platformei, dar care m-au ajutat în proiectare acesteia. Astfel, uneltele pe care le-am folosit în realizarea diagramelor sunt:

- Bee-up este un tool pe care l-am utilizat pentru crearea diagramelor care respectă standardul UML Acesta a fost folosit pentru a modela și vizualiza structura aplicației. De asemenea, am facilitat comunicarea și înțelegerea componentelor, precum și relațiile dintre acestea.
- Microsoft Visio este un alt tool de diagramare care a fost folosit pentru a crea diagrama Business Model Canvas. Acesta a permis reprezentarea vizuală clară a modelului de afaceri și a componentelor sale.
- Archi este un tool pentru modelarea arhitecturii de afaceri utilizând limbajul Archimate. Acesta a fost folosit pentru a crea diagrama de motivație Archimate, prin care am evidențiat obiectivele și relațiile dintre actorii implicați.
- Lucidchart este un tool de diagramare online pe care l-am utilizat pentru a crea diagrama Fish-Bone. Acesta a facilitat vizualizarea cauzelor și efectelor într-un mod clar și organizat.
- Draw.io este un tool de diagramare online pe care l-am utilizat pentru a crea diagrama ERD. Astfel, am surprins modul în care sunt stocate datele și informațiile platformei în baza de date.
- Microsoft Excel a fost utilizat pentru proiectarea diagramei Pareto care surprinde problemele și prioritățile.

4. Proiectarea logică

Proiectarea logică este fundamentală în dezvoltarea oricărei aplicații software. Acest pas reprezintă puntea dintre documentarea cerințelor de sistem și proiectarea tehnică. Scopul capitolului este de a descrie în detaliu structura conceptuală a aplicației de gestionare a rezervărilor pentru sălile de fitness. De asemenea, voi oferi o imagine clară asupra modului în care procesele și activitățile sunt organizate, cum sunt gestionate datele și cum interacționează diferitele componente ale sistemului.

În contextul proiectului meu, partea de proiectarea logică a pus accent pe definirea precisă a fluxurilor de lucru și a interacțiunilor utilizatorilor cu aplicația. Am stabilit modelul de date care va susține funcționalitățile aplicației, iar cu ajutorul diagramelor am ilustrat aceste concepte într-un mod clar și concis.

4.1 Procese și activități

Aplicația pe care am dezvoltat-o gestionează accesul angajaților în sălile de fitness ale companiei. Cu ajutorul acestei platforme se facilitează procesul de rezervare a intervalelor orare. De asemenea, se asigură utilizarea optimă a resurselor și se înlocuiește metoda manuală de gestionare care era anevoioasă. Erorile sunt reduse, iar procesele administrative sunt eficientizate.

Această secțiune va detalia procesele esențiale implicate în utilizarea aplicației, evidențiind transformările aduse de digitalizarea și automatizarea sistemului de rezervări. Voi analiza fluxurile de lucru atât din perspectiva utilizatorilor finali, cât și a administratorilor platformei, prin prezentarea diagramelor de activități în format As-Is și To-Be. Diagramele oferă o vizualizare comparativă a proceselor curente și a celor optimizate prin utilizarea aplicației.

În contextul actual, angajații din compania cu care am interacționat se confruntă cu dificultăți în accesarea sălilor de fitness. Acest fapt se datorează lipsei unui sistem centralizat de rezervări care să reducă supraaglomerarea . Aplicația este concepută să rezolve aceste probleme prin implementarea unui sistem automatizat. Angajații vor accesa platforma cu ajutorul adresei de email corporative. Apoi, vor putea rezerva o sesiunea de antrenament în sala dorită, în funcție de capacitatea acesteia. În acest fel, se va evita aglomerația și se va asigura o experiență plăcută pentru utilizatori.

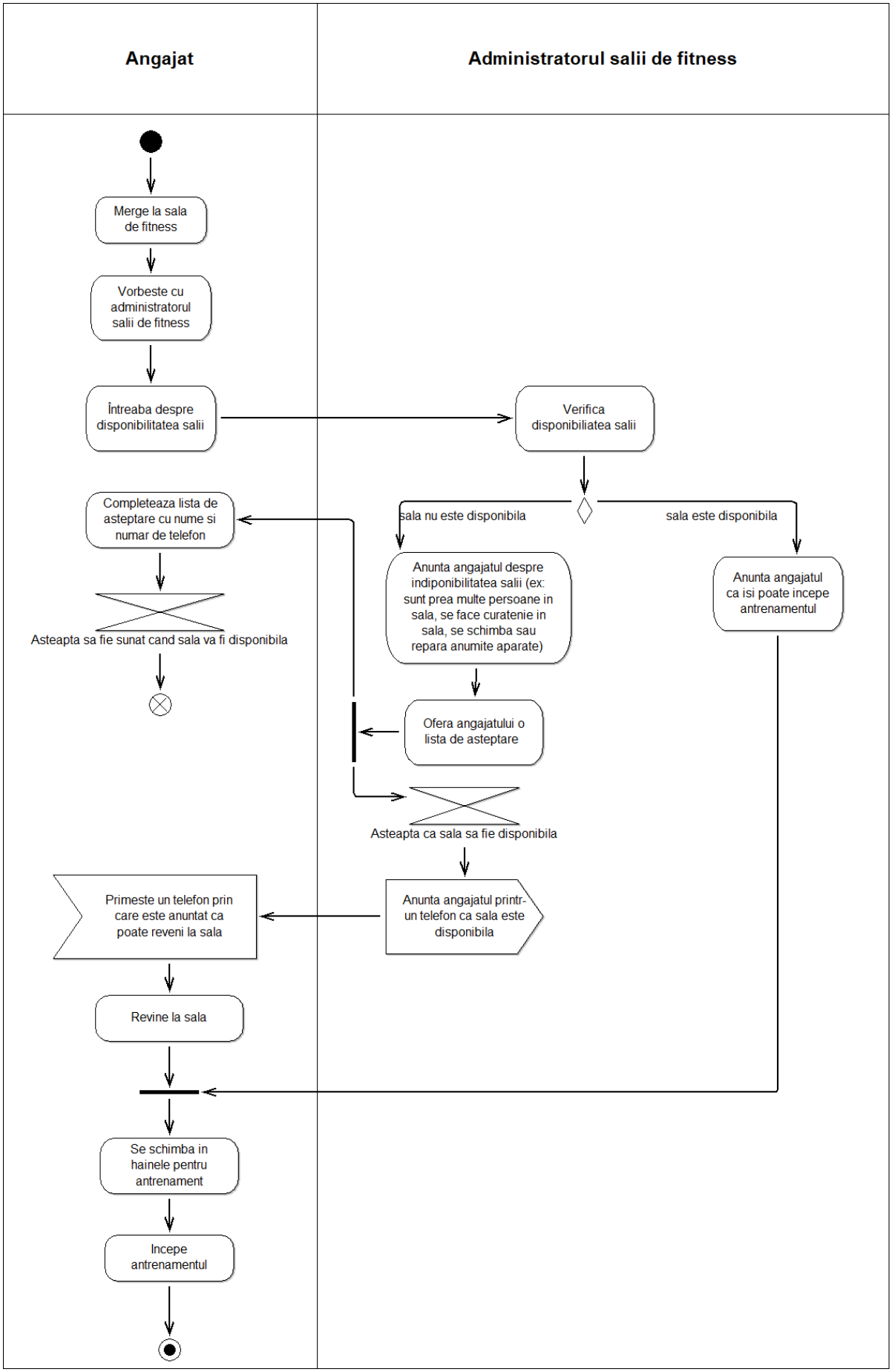


Figura 6 - Diagrama de activități As-Is

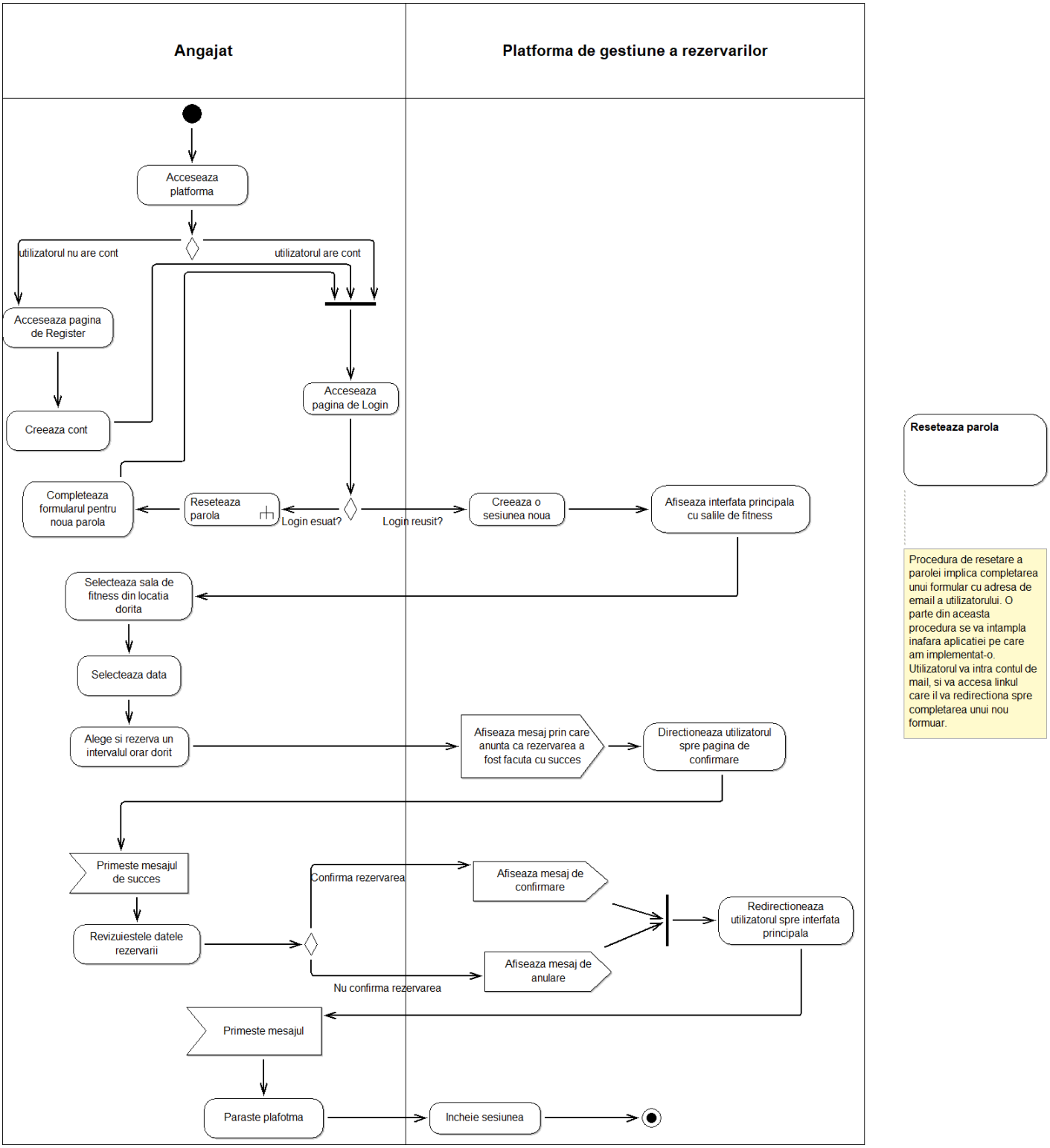


Figura 7 - Diagrama de activități To-Be din perspectiva utilizatorilor finali ai aplicației

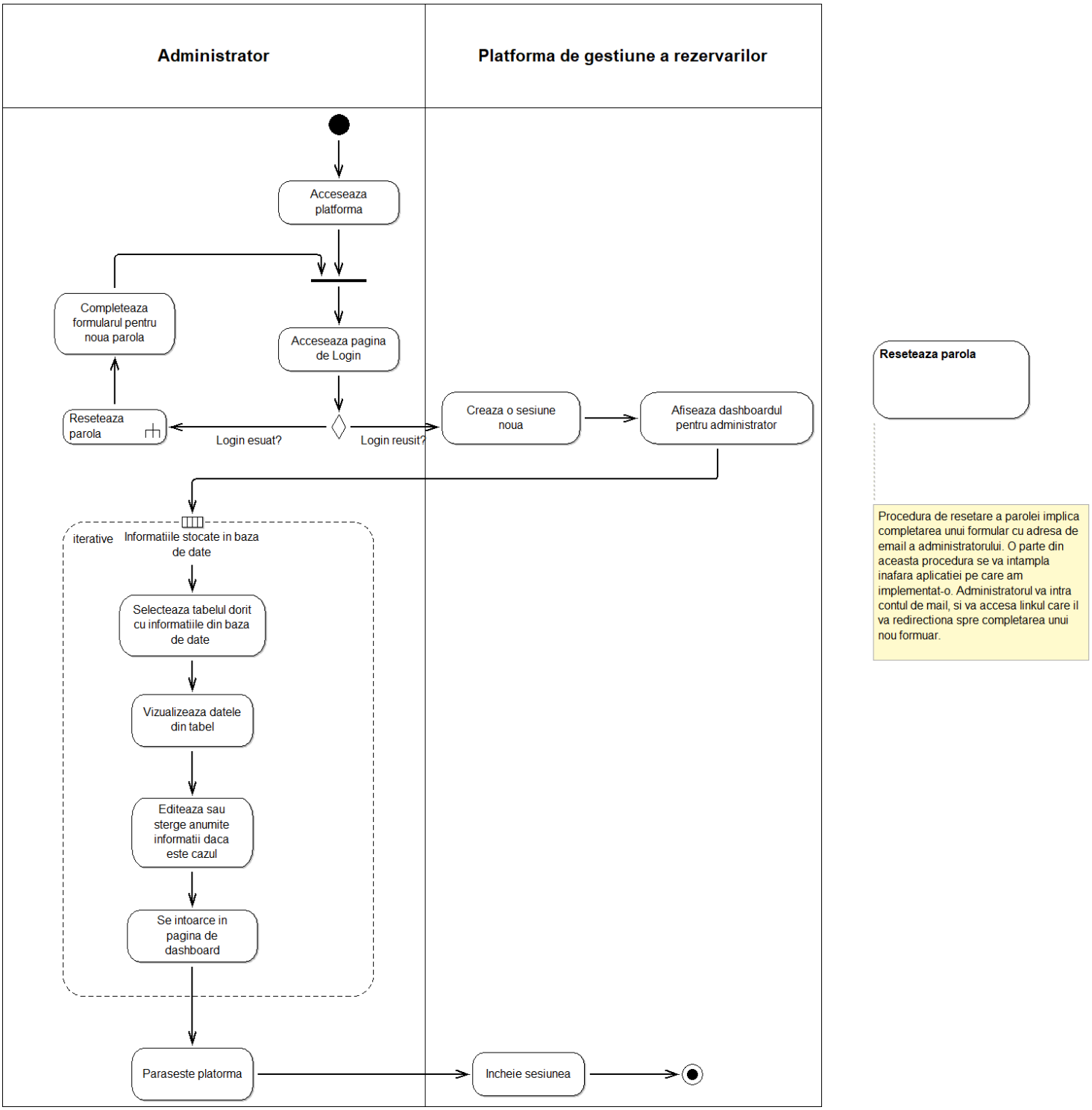


Figura 8 - Diagrama de activități To-Be din perspectiva administratorilor sălilor de fitness/platformei

Cu ajutorul standardului UML, am încercat să surprind acțiunile pe care le efectuează actorii implicați în acest proces. În primul rând, în Figura 6 am proiectat diagrama de activități As-Is prin care am arătat toate etapele pe care un angajat le parcurge pentru a folosi facilitățile de fitness. Această diagramă este menită să evidențieze procesul curent de gestionare a sălilor de fitness, înainte de implementarea platformei. Se poate observa că există timpi de așteptare și mult efort inutil. De exemplu, angajatul e nevoit să meargă fizic în sala de fitness pentru a afla dacă este disponibilă. Prin disponibilitate ne referim atât la capacitatea sălii, adică să nu fie deja prea multe persoane în sală, dar și la eventualele situații neprevăzute. Aceste situații neprevăzute pot fi: momentul în care este nevoie de repararea sau înlocuirea unor aparate de fitness, procesul de mentenanță sau curățenia periodică efectuată în sală.

În cazul în care angajatul primește un răspuns negativ în ceea ce privește disponibilitatea sălii, intervine o situație extrem de neplăcută. Acesta este nevoit să aștepte până când sala va fi disponibilă pentru a se evita supraaglomerarea. În momentul de față, la fel cum se poate vedea și în Figura 6, procesul este unul anevoios. Din acest motiv, implementarea unui platforme online prin care angajații să aibă posibilitatea să își rezerve un interval orar este esențială. În diagrama de activități To-Be din Figura 7, am creionat procesul prin care trece acum, un angajat, în momentul în care dorește să rezerve un loc într-o anumită sală de fitness. Acțiunile pe care le-am pus în această diagramă surprind exact etapele pe care le parcurge angajatul prin folosirea platformei.

Mai apoi, în Figura 8 am structurat interacțiunea dintre administratori și aplicație. Administratorii sunt acele persoane care se ocupă în totalitate de sălile de fitness. Aceștia încercau să mențină un număr decent de persoane care se aflau în sală la un anumit interval de timp. Însă, această încercare era de multe ori una aproximativă deoarece au fost destul de multe situații în care capacitatea recomandată a fost depășită. Aceste situații duceau de cele mai multe ori la supraaglomerare, timpi de așteptare pentru eliberarea anumitor aparate și nemulțumiri din partea angajaților. Cu ajutorul platformei online, administratorii nu se mai ocupă fizic de gestionarea rezervărilor, ci doar analizează datele despre fiecare rezervare din meniul dashboard. În această pagină dedicată exclusiv persoanelor cu rol de administrator, se pot face anumite modificări cu privire la conturile utilizatorilor, rezervările acestora sau informațiile despre sălile de fitness.

Astfel, prin aceste diagrame am dorit să surprind procesul descris în două faze, înainte de a implementa aplicația și după implementarea ei. Cu toate acestea, se poate observa lipsa a două cerințe identificate în procesul de elicitație: generarea de rapoarte pentru managerii de

resurse umane și notificarea utilizatorilor prin email cu privire la rezervarea făcută. Cum am menționat și anterior, aceste funcționalități vor fi dezbătute pe larg în capitolul de idei viitoare de dezvoltare.

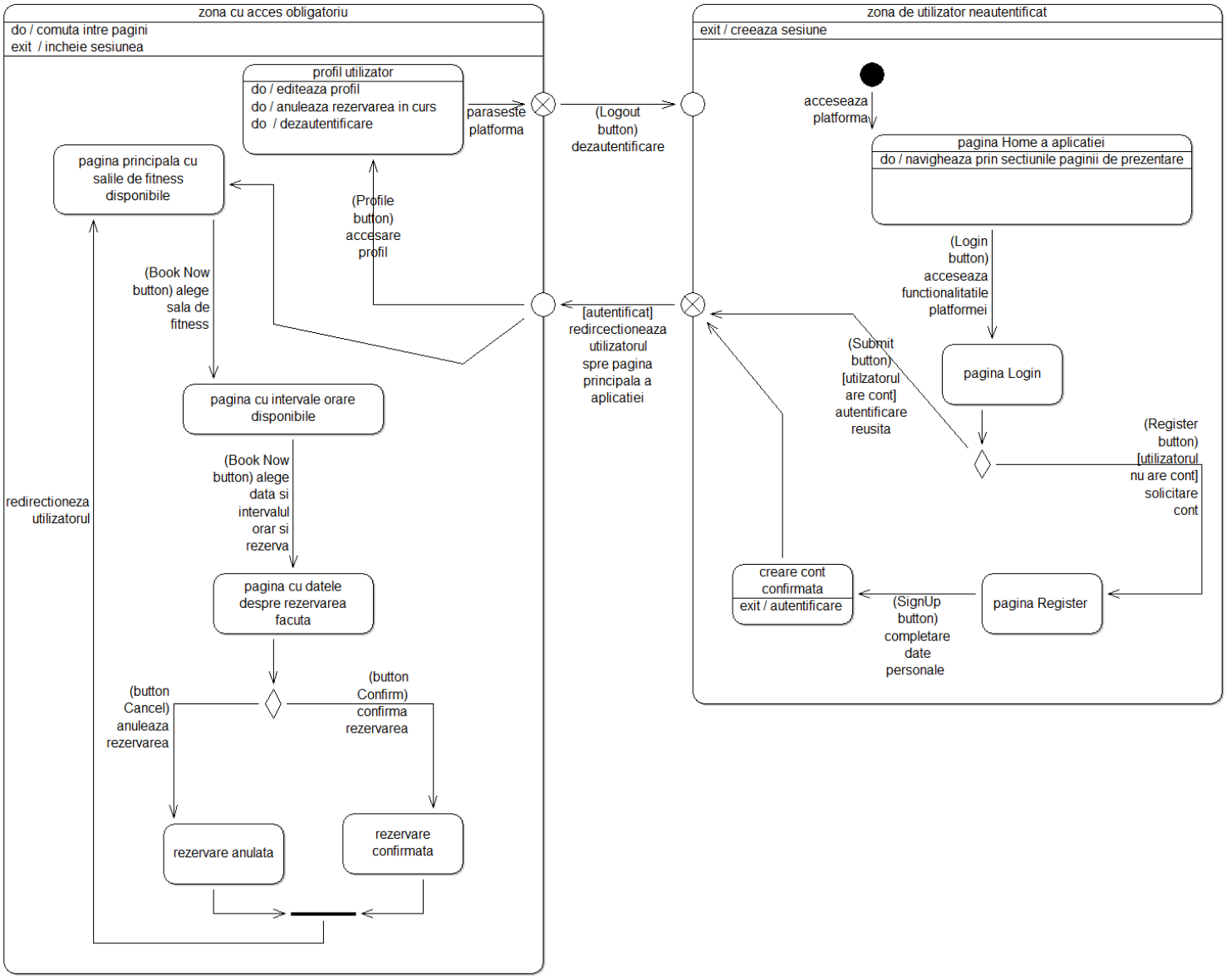


Figura 9 - Diagrama de stări

Diagrama din Figura 9 scoate în evidență fluxul logic de stări care sunt implicate în interacțiunea dintre utilizator și platformă. Stările reprezintă paginile web prin care utilizatorul navighează, cum ar fi pagina de autentificare, pagina principală, pagina de rezervări și pagina de confirmare. Fiecare stare este legată de acțiuni specifice pe care utilizatorul le poate efectua, iar diagrama ilustrează tranzițiile între aceste stări.

4.2 Managementul datelor

Managementul datelor reprezintă un aspect crucial în dezvoltarea aplicației de gestionare a rezervărilor în sălile de fitness. Printr-un management eficient al datelor, am asigurat stocarea informațiilor și accesarea lor într-o manieră coerentă și securizată. Astfel, se facilitează utilizarea optimă a resurselor și se oferă o experiență fără întreruperi sau erori pentru utilizatori.

Baza de date reprezintă inima aplicației, deoarece aici sunt stocate toate informațiile esențiale despre utilizatori, rezervări, săli de fitness și intervale orare. În acest sens, am adoptat o structură relațională bine definită, care să permită legături clare și eficiente între diferitele tabele. Prin asigurarea unui management eficient al datelor, am încercat să ofer o aplicație robustă și fiabilă, capabilă să gestioneze în mod optim rezervările și sălile de fitness. Un alt aspect pe care a trebui să îl iau în considerare a fost protejarea informațiilor sensibile ale utilizatorilor. Acest subcapitol descrie în detaliu modelul de date utilizat în aplicație, inclusiv descrierea tabelelor, relațiilor și tipurilor de date.

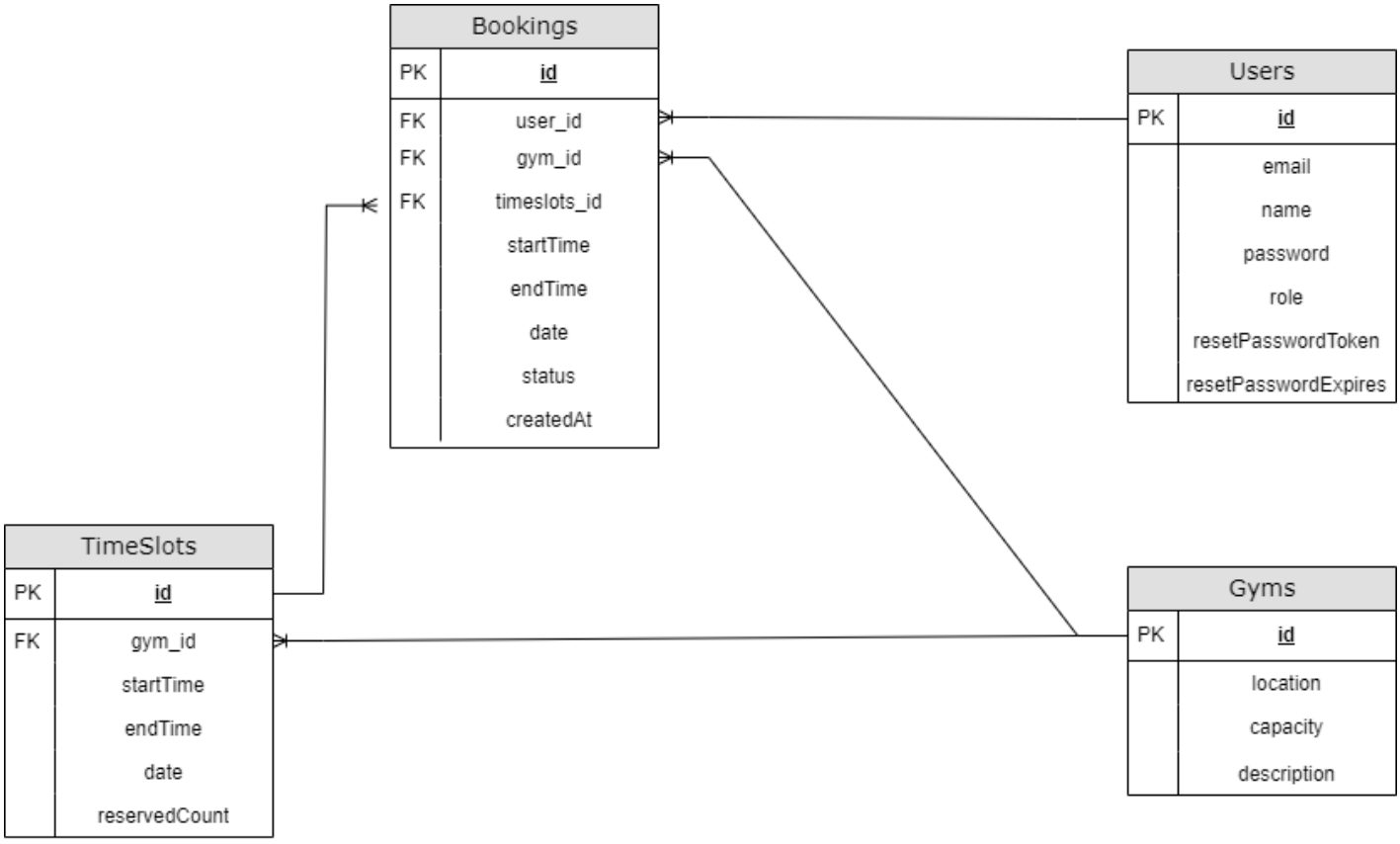


Figura 10 - Diagrama ERD

În Figura 10 se poate observa diagrama entitate-relație (ERD). Acest tip de diagramă oferă o reprezentare vizuală a modelului de date și evidențiază principalele entități, precum și relațiile dintre acestea. Aceasta este esențială pentru a înțelege modul în care datele sunt interconectate și pentru a facilita dezvoltarea ulterioară a aplicației. În continuare voi descrie informațiile care sunt stocate în fiecare tabel din baza de date.

Tabelul „Users" stochează informațiile despre utilizatorii platformei. Fiecare utilizator are un identificator unic și alte atribute relevante pentru gestionarea conturilor, precum:

- id: int(11), not null, auto_increment - Cheia primară a tabelului. Identifică unic fiecare utilizator.
- email: varchar(100), not null - Adresa de email a utilizatorului. Utilizată pentru autentificare și comunicare.
- name: varchar(100), not null - Numele complet al utilizatorului.
- password: varchar(100), not null - Parola utilizatorului, stocată într-un format hashuit pentru securitate.
- role: int(2), not null - Rolul utilizatorului în sistem (1 pentru administrator, 2 pentru utilizator obișnuit).
- resetPasswordToken: varchar(256), null - Token-ul utilizat pentru resetarea parolei.
- resetPasswordExpires: datetime, null - Data și ora expirării token-ului de resetare a parolei.

Tabelul „Gyms" stochează informațiile despre sălile de fitness disponibile pentru rezervare:

- id: int(11), not null, auto_increment - Cheia primară a tabelului. Identifică unic fiecare sală de fitness.
- location: varchar(100), not null - Locația sălii de fitness.
- capacity: int(3), not null - Capacitatea maximă a sălii de fitness (numărul maxim de persoane).
- description: text, not null - Descrierea detaliată a sălii de fitness.

Tabelul „TimeSlots" stochează informațiile despre intervalele orare disponibile pentru rezervare în fiecare sală de fitness:

- id: int(11), not null, auto_increment - Cheia primară a tabelului. Identifică unic fiecare interval orar.

- gym_id: int(11) , not null - Cheia străină care referențiază tabela „Gyms”. Indică sala de fitness asociată intervalului orar.
- startTime: time, not null - Ora de început a intervalului orar.
- endTime: time, not null - Ora de sfârșit a intervalului orar.
- date: date, not null - Data specifică a intervalului orar.
- reservedCount: int(11) , not null – Numărul total de rezervări disponibile pentru acest interval orar.

Tabelul „Bookings” stochează informațiile despre rezervările efectuate de utilizatori:

- id: int(11), not null, auto_increment - Cheia primară a tabelului. Identifică unic fiecare rezervare.
- user_id: int(11), not null - Cheia străină care referențiază tabela „Users”. Indică utilizatorul care a efectuat rezervarea.
- gym_id: int(11), not null - Cheia străină care referențiază tabela „Gyms”. Indică sala de fitness rezervată.
- timeslots_id: int(11), not null - Cheia străină care referențiază tabela „TimeSlots”. Indică intervalul orar rezervat.
- startTime: time, not null - Ora de început a rezervării (preluată din „TimeSlots”).
- endTime: time, not null - Ora de sfârșit a rezervării (preluată din „TimeSlots”).
- date: date, not null - Data rezervării (preluată din „TimeSlots”).
- status: enum('pending', 'confirmed', 'cancelled'), not null - Starea rezervării.
- createdAt: timestamp, not null, current_timestamp() - Data și ora la care a fost creată rezervarea.

În ceea ce privește relațiile care conectează entitățile prezentate mai sus, explicația este următoarea:

- Relația „One-to-Many” între „Users” și „Bookings”: Un utilizator poate face mai multe rezervări (desigur, un utilizator poate avea doar o rezervare curentă)
- Relația „One-to-Many” între „Gyms” și „Bookings”: O sală de fitness poate fi rezervată de mai multe ori.
- Relația „One-to-Many” între „Gyms” și „TimeSlots”: O sală de fitness poate avea multiple intervale orare disponibile pentru rezervare.
- Relația „One-to-Many” între „TimeSlots” și „Bookings”: Un interval orar poate fi rezervat de mai multe ori.

5. Proiectarea tehnică

Proiectarea tehnică reprezintă o etapă esențială în dezvoltarea oricărui sistem software. Acesta este momentul în care cerințele funcționale și nefuncționale sunt transformate într-o arhitectură concretă și detaliată. În cadrul acestui capitol, voi explora în profunzime arhitectura tehnică a aplicației.

WellnessWork Hub este concepută pentru a facilita gestionarea rezervărilor în sălile de fitness și pentru a preveni supraaglomerarea. Pentru a îndeplini aceste obiective, aplicația utilizează o arhitectură bazată pe microservicii, cu un front-end dezvoltat în React și un back-end construit pe Node.js. Aceste componente sunt interconectate prin intermediul API-urilor RESTful. Baza de date utilizată este MySQL, gestionată local prin Xampp, iar sarcinile automate sunt programate cu ajutorul Task Scheduler.

În continuare, voi prezenta arhitectura generală a sistemului. Apoi voi detalia diagrama de deployment a aplicației, care ilustrează modul în care componentele sunt distribuite pe diferite platforme de hardware și software. Aceasta va fi urmată de o diagrama de componente, care va prezenta structura internă a fiecărei componente și interacțiunile dintre ele. De asemenea, voi menționa tehnologiile și bibliotecile folosite, precum și modul în care acestea contribuie la funcționarea generală a sistemului. Acest capitol va servi ca bază pentru înțelegerea implementării tehnice și a strategiilor de testare prezentate în capitolele următoare.

5.1 Arhitectura sistemului

Aplicația WellnessWork Hub urmează o arhitectură de tip MVC (Model-View-Controller), adaptată specificului tehnologiilor folosite. Arhitectura sistemului este proiectată pentru a separa logica aplicației de interfața utilizatorului și de gestionarea datelor. Astfel, se asigură o mai bună organizare și scalabilitate a codului.

- Modelul (Model): Acesta include toate fișierele care definesc structura datelor și interacțiunea cu baza de date. În proiectul meu, aceste modele sunt grupate în folderul models.
- Controlerul (Controller): Controlerele gestionează logica aplicației și sunt situate în folderul controllers. Acestea primesc solicitările HTTP, procesează datele folosind modelele și returnează răspunsurile adecvate.

- Vederea (View): Deși nu avem fișiere view tradiționale în back-end, partea de front-end, dezvoltată în React, servește acestui scop. Componentele React din folderul pages sunt responsabile pentru prezentarea datelor utilizatorului prin interacțiunea cu API-urile din back-end.

5.2 Diagrama de deployment

Diagrama de deployment descrie modul în care componentele aplicației sunt distribuite și cum interacționează acestea pe diferite platforme de hardware și software. Serverul este infrastructura unde este instalat sistemul de operare Windows OS, care oferă mediul de execuție pentru diverse componente. Pe sistemul de operare Windows OS este instalat serverul local Xampp, care gestionează baza de date. Baza de date utilizată pentru stocarea datelor aplicației este MySQL. De asemenea, serviciul Task Scheduler, care rulează pe același sistem de operare, este utilizat pentru programarea și executarea scripturilor automate.

Mediul de execuție pentru back-end-ul aplicației WellnessWork Hub este Node.js. În cadrul acestui mediu, sunt utilizate mai multe biblioteci. Express este framework-ul web folosit pentru a construi API-ul serverului. Bcrypt și crypto sunt biblioteci utilizate pentru hashingul parolelor, respectiv pentru generarea și manipularea datelor criptografice. Biblioteca cors este folosită pentru gestionarea cererilor CORS, iar dotenv pentru încărcarea variabilelor de mediu dintr-un fișier cu extensia .env. Mysql2 asigură conexiunea la baza de date MySQL. Biblioteca node-cron permite programarea taskurilor cron. Express-session este biblioteca pentru gestionarea sesiunilor, iar express-mysql-session este biblioteca pentru stocarea sesiunilor în baza de date MySQL. Biblioteca @sendgrid/mail ajută la trimiterea emailurilor.

Dispozitivul utilizatorului este locul unde se execută interfața aplicației în browser. Mediul de execuție pentru front-end-ul aplicației WellnessWork Hub este browserul. Pentru partea de front-end, la fel ca în back-end, am instalat anumite biblioteci. React este biblioteca JavaScript utilizată pentru construirea interfeței de utilizator. Biblioteca axios e folosită pentru a face cereri HTTP către server. React-bootstrap ajută la stilizarea componentelor din interfața utilizatorului. React-datepicker și moment sunt biblioteci utilizate pentru selectarea, respectiv manipularea și formatarea datelor. React-icons este o biblioteca pentru utilizarea de pictograme în aplicație, în timp ce react-router-dom gestionează rutele între paginile interfeței.

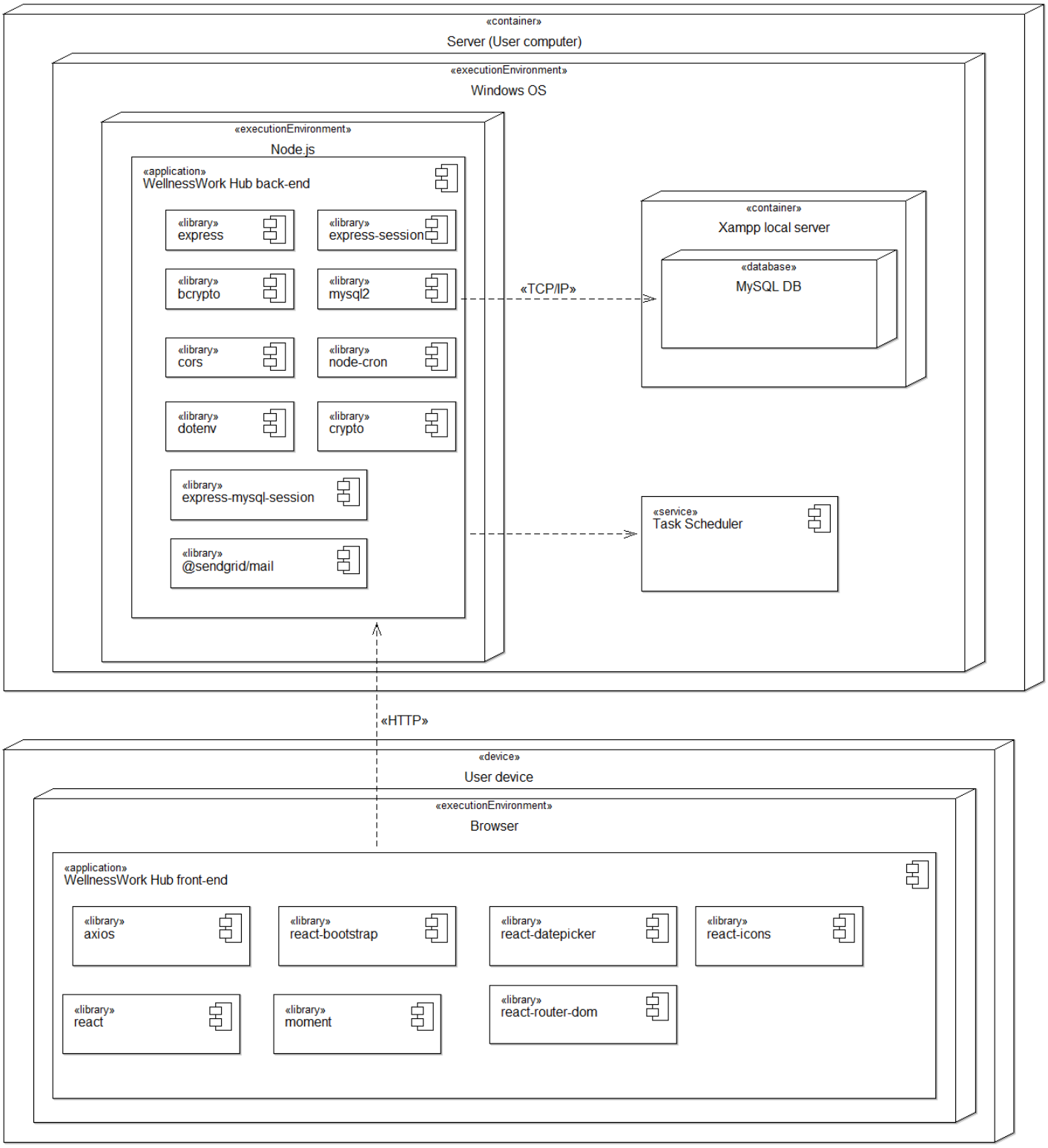


Figura 11 - Diagrama de deployment

5.3 Diagrama de componente

Diagrama de componente descrie structura internă a aplicației WellnessWork Hub. Aceasta oferă o vedere detaliată a modului în care sunt organizate și interacționează diversele componente ale sistemului. Diagrama de componente este esențială pentru înțelegerea arhitecturii sistemului deoarece facilitează identificarea și gestionarea responsabilităților fiecărei componente în parte.

În partea de back-end, aplicația utilizează Node.js, un mediu de execuție JavaScript care permite dezvoltarea de aplicații scalabile și rapide. Structura back-end-ului este organizată în mai multe componente. Modelele reprezintă structura de date și logica de acces la baza de date. Acestea sunt utilizate pentru a defini schema datelor și pentru a executa operațiuni CRUD (Create, Read, Update, Delete) asupra bazei de date. Controlerele gestionează logica aplicației și interacțiunile cu utilizatorul. Ele procesează cererile primite de la front-end, apelează funcțiile modelelor pentru a accesa sau modifica datele și returnează răspunsurile adecvate. Rutele definesc endpoint-urile API-ului și mapează cererile HTTP către funcțiile corespunzătoare din controlere. Scripturile sunt utilizate pentru sarcini automate și programate. Componentele principale ale back-end-ului interacționează cu baza de date MySQL prin intermediul fișierul database.js. Acest fișier asigură conexiunea cu baza de date, unde sunt stocate toate datele necesare pentru funcționarea aplicației. Fișierul app.js este responsabil pentru configurarea inițială a aplicației și pentru definirea punctelor de intrare principale.

În partea de front-end, aplicația utilizează React, o bibliotecă JavaScript pentru construirea interfețelor de utilizator. Fiecare pagină reprezintă o componentă UI specifică și conține logica și stilurile necesare pentru afișarea informațiilor utilizatorului. Stilurile (CSS) sunt utilizate pentru a defini aspectul vizual al componentelor UI. Fișiere publice includ active statice, cum ar fi imagini și alte resurse necesare pentru interfața utilizatorului. Fișierele principale, cum ar fi index.js și App.js, gestionează inițializarea aplicației și configurarea rutelor pentru front-end. Componentele front-end comunică cu back-end-ul prin cereri HTTP gestionate de axios, o bibliotecă JavaScript pentru efectuarea cererilor HTTP. Această interacțiune permite aplicației să preia și să trimită date către server, actualizând interfața utilizatorului în timp real.

Pentru a ușura vizualizarea diagramei de componente, am prezentat doar componentele esențiale, atât din partea de back-end, cât și din partea de front-end. De asemenea, am simplificat reprezentarea rutelor API prin includerea acestora direct pe porturile controlerelor.

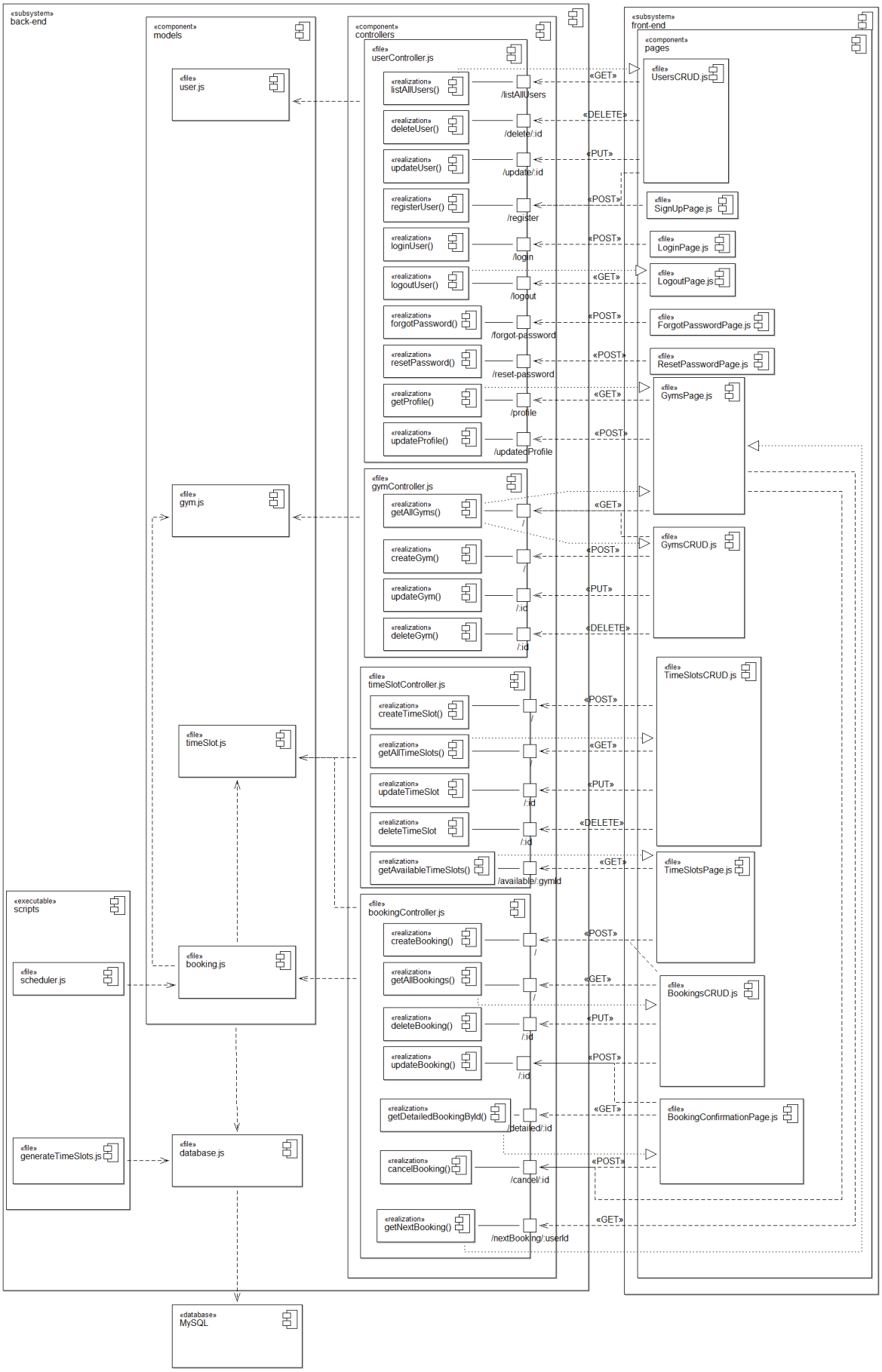


Figura 12 - Diagrama de componente

6. Implementarea

Capitolul de implementare al lucrării descrie procesul prin care aplicația WellnessWork Hub a fost transpusă din faza de proiectare tehnică în cod executabil. În acest capitol, voi detalia modul în care am realizat componentele esențiale ale aplicației, atât pe partea de back-end cât și pe front-end. De asemenea, voi include fragmente de cod elocvente care ilustrează funcționalitățile cheie. Implementarea a fost realizată cu atenție la detalii pentru a asigura funcționarea corectă și eficientă a sistemului, respectând bunele practici în dezvoltarea software.

Pentru început, voi prezenta structura generale a proiectului, urmată de detalii despre părțile componente. Apoi, voi evidenția utilizarea tehnologiilor și bibliotecilor specifice, care au contribuit la funcționalitatea generală a aplicației. De asemenea, includerea fragmentelor de cod relevante va exemplifica modul în care funcționează sistemul.

Proiectul este structurat în două părți principale. Partea de back-end include toate componentele serverului, cum ar fi modelele de date, controlerele, rutele, middleware-uri și scripturile automate. Front-end-ul include interfața pentru utilizator, construită folosind React și diverse biblioteci pentru a crea o experiență interactivă și intuitivă.

6.1 Configurarea conexiunii la baza de date

În primul rând, configurarea conexiunii la baza de date este esențială pentru a asigura interacțiunea corectă cu baza de date MySQL. Fișierul care se ocupă de acest aspect este database.js și are următorul conținut:

```
const mysql = require("mysql2/promise");
const pool = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "",
  database: "gym_booking_system",
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0,
});
module.exports = pool;
```

Linia de cod „const mysql = require("mysql2/promise");” importă biblioteca mysql2/promise, care este o versiune modernă a clientului MySQL pentru Node.js. Aceasta oferă suport pentru utilizarea async/await pentru operațiuni asincrone. Biblioteca

mysql2/promise permite gestionarea conexiunilor la baza de date într-un mod asincron, ceea ce îmbunătățește performanța și scalabilitatea aplicației prin evitarea blocării execuției.

Apoi, linia „const pool = mysql.createPool({...});” creează un pool de conexiuni la baza de date MySQL, folosind configurațiile specificate în obiectul dat ca parametru. Utilizarea unui pool de conexiuni este o practică obișnuită pentru aplicațiile care necesită acces frecvent la baza de date. Pool-ul permite reutilizarea conexiunilor care ajută la reducerea timpului și resurselor necesare pentru a stabili conexiuni noi de fiecare dată când aplicația interacționează cu baza de date. Parametrii specificați sunt:

- host: Adresa serverului MySQL (în acest caz, localhost indică că baza de date rulează pe aceeași mașină cu aplicația).
- user: Numele de utilizator pentru autentificarea la baza de date (root este utilizatorul implicit în MySQL).
- password: Parola asociată utilizatorului (în acest caz lipsește).
- database: Numele bazei de date utilizate (gym_booking_system).
- waitForConnections: Dacă este true, atunci cererile de conexiune vor aștepta până când una devine disponibilă în loc să eșueze imediat.
- connectionLimit: Numărul maxim de conexiuni active în pool (setat la 10 în acest caz).
- queueLimit: Numărul maxim de cereri de conexiune care pot fi puse în coadă (setat la 0 pentru a permite un număr nelimitat).

În final, secvența „module.exports = pool;” exportă pool-ul de conexiuni pentru a fi utilizat în alte module ale aplicației. Exportarea pool-ului permite altor fișiere din aplicație să importe și să utilizeze același pool de conexiuni. Astfel, se asigură o gestionare eficientă și centralizată a accesului la baza de date. Desigur, înainte de a configura conexiunea, a fost necesar să creez, pe serverul local, baza de date cu numele „gym_booking_system”.

6.2 Modele (Models)

Modelele sunt componentele care se ocupă de interacțiunea cu baza de date. În cadrul aplicației WellnessWork Hub, am implementat mai multe modele pentru a gestiona diferite entități ale aplicației, mai exact: user.js, gym.js, timeslot.js și booking.js.

Fișierul user.js definește modelul pentru utilizatori și include funcții pentru crearea, găsirea, actualizarea și ștergerea utilizatorilor. În plus, conține funcțiile care ajută la resetarea parolelor pentru utilizatori.

```
const bcrypt = require("bcrypt");
const pool = require("../lib/database");

const createUser = async ({ email, name, password }) => {
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const defaultRole = 2; // 2 reprezintă rolul inițial pentru un utilizator nou
    const sql =
      "INSERT INTO users (email, name, password, role) VALUES (?, ?, ?, ?)";
    const [result] = await pool.query(sql, [
      email,
      name,
      hashedPassword,
      defaultRole,
    ]);
    return { id: result.insertId, email, name, role: defaultRole };
  } catch (error) {
    throw error;}
};
const setResetPasswordToken = async (email, token) => {
  const expires = new Date(Date.now() + 3600000); // 1 oră până la expirare
  try {
    const sql =
      "UPDATE users SET resetPasswordToken = ?, resetPasswordExpires = ? WHERE email
= ?";
    const [result] = await pool.query(sql, [token, expires, email]);
    return result.affectedRows > 0;
  } catch (error) {
    throw error;}
};
const getResetPasswordToken = async (token) => {
  try {
    const sql =
      "SELECT * FROM users WHERE resetPasswordToken = ? AND resetPasswordExpires
> ?";
    const [rows] = await pool.query(sql, [token, Date.now()]);
    if (rows.length > 0) {
      return rows[0];
    } else {
      return null;
    }
  } catch (error) {
    throw error;}
};
const resetPassword = async (userId, newPassword) => {
  try {
    const hashedPassword = await bcrypt.hash(newPassword, 10);
    const sql =
```

```
    "UPDATE users SET password = ?, resetPasswordToken = NULL, resetPasswordExpires
= NULL WHERE id = ?";
   const [result] = await pool.query(sql, [hashedPassword, userId]);
   return result.affectedRows > 0;
  } catch (error) {
   throw error;}
};
```

Primele două linii de cod definesc importarea unor biblioteci care vor fi folosite în cadrul fișierului. Mai exact, bcrypt este utilizat pentru hashing-ul parolelor, asigurând astfel securitatea datelor utilizatorilor, iar pool este creat din configurația bazei de date pentru a facilita conexiunile asincrone.

Funcția createUser este responsabilă pentru crearea unui utilizator nou în baza de date. Parola utilizatorului este hash-uită folosind bcrypt pentru securitate. Fiecare utilizator nou creat are rolul implicit de 2, care reprezintă un utilizator standard. Dacă un utilizator trebuie să aibă rolul de administrator (rol 1), administratorul bazei de date poate să modifice manual rolul utilizatorului direct în baza de date sau din dashboard-ul administratorului.

Funcția setResetPasswordToken setează un token pentru resetarea parolei pentru un utilizator specificat prin email. Token-ul are o valabilitate de 1 oră, după care expiră. Aceasta funcționează prin generarea unui token și a unei date de expirare și actualizează utilizatorul din baza de date cu aceste valori. Token-ul generat și data de expirare sunt stocate în câmpurile resetPasswordToken și resetPasswordExpires din tabelul utilizatorilor. În cazul în care există o eroare în timpul actualizării bazei de date, aceasta este prinsă și aruncată.

Funcția getResetPasswordToken obține un utilizator pe baza token-ului de resetare a parolei și verifică dacă token-ul este încă valid. Aceasta funcționează prin căutarea unui utilizator în baza de date care are token-ul specificat și o dată de expirare mai mare decât data curentă. Dacă este găsit un utilizator valid, acesta este returnat, altfel funcția returnează null. Aceasta asigură că numai token-urile valide pot fi utilizate pentru resetarea parolelor.

Funcția resetPassword resetează parola utilizatorului cu un id specificat, hash-uind noua parolă și setând token-ul de resetare și expirarea token-ului la null. Aceasta funcționează prin generarea unei parole hash-uite folosind bcrypt și actualizează utilizatorul din baza de date cu noua parolă, ștergând în același timp token-ul de resetare și data de expirare. Aceasta asigură că după resetarea parolei, token-ul și data de expirare sunt eliminate, prevenind utilizarea ulterioară a aceluiași token.

Fișierul timeSlot.js gestionează intervalele de timp disponibile pentru rezervări în cadrul sălilor de sport. Funcții pe care le voi prezenta în continuare le-am considerat esențiale pentru gestionarea corectă a intervalelor.

```
const getAvailableTimeSlotsForGym = async (gymId, date) => {
  const sql = `
    SELECT ts.id, ts.startTime, ts.endTime, ts.date,
          IFNULL(b.reservedCount, 0) as reservedCount, g.capacity
    FROM timeslots ts
    LEFT JOIN (
      SELECT timeslots_id, COUNT(*) as reservedCount
      FROM bookings
      WHERE date = ?
      GROUP BY timeslots_id
    ) b ON ts.id = b.timeslots_id
    JOIN gyms g ON ts.gym_id = g.id
    WHERE ts.gym_id = ? AND ts.date = ? AND g.capacity > IFNULL(b.reservedCount, 0)
  `;
  const [rows] = await pool.query(sql, [date, gymId, date]);
  return rows;
};

const incrementReservedCount = async (timeSlotId) => {
  const sql = `UPDATE timeslots SET reservedCount = reservedCount + 1 WHERE id = ?`;
  const [result] = await pool.query(sql, [timeSlotId]);
  return result.affectedRows === 1;
};

const decrementReservedCount = async (timeSlotId) => {
  const sql = `UPDATE timeslots SET reservedCount = reservedCount - 1 WHERE id = ?
AND reservedCount > 0`;
  const [result] = await pool.query(sql, [timeSlotId]);
  return result.affectedRows === 1;
};
```

Funcția getAvailableTimeSlotsForGym obține intervalele de timp disponibile pentru o anumită sală de fitness și o anumită dată. Aceasta funcționează prin execuția unei interogări SQL complexe care selectează intervalele de timp (timeslots) pentru sala specificată (gym_id) și data specificată. Interogarea folosește o subinterogare pentru a calcula numărul de rezervări pentru fiecare interval de timp și se asigură că intervalele de timp returnate au încă locuri disponibile, comparând capacitatea sălii de sport cu numărul de rezervări existente. Funcția returnează o listă de intervale de timp disponibile.

Funcția incrementReservedCount incrementează numărul de rezervări pentru un interval de timp specificat. Aceasta funcționează prin execuția unei interogări SQL UPDATE care crește valoarea reservedCount pentru intervalul de timp cu id specificat.

Inversul funcției de mai sus este funcția decrementReservedCount. Aici se decrementează numărul de rezervări pentru un interval de timp specificat. Aceasta funcționează prin execuția unei interogări SQL UPDATE care scade valoarea reservedCount pentru intervalul de timp cu id specificat, cu condiția ca reservedCount să fie mai mare de 0.

Fișierul booking.js gestionează rezervările făcute de utilizatori pentru diverse intervale de timp în sălile de sport. Mai jos sunt explicate câteva din funcțiile prezente în acest fișier, care sunt esențiale pentru funcționarea corectă a sistemului de rezervări.

```
const pool = require("../lib/database");
const {
  getTimeSlotById,
  decrementReservedCount,
  incrementReservedCount,
} = require("./timeSlot");
const { getGymById } = require("./gym");

const createBooking = async (booking) => {
  const { user_id: userId, timeslots_id: timeSlotId, status } = booking;

  await pool.query("START TRANSACTION");

  try {
    const existingActiveBooking = await hasActiveBooking(userId);
    if (existingActiveBooking) {
      await pool.query("ROLLBACK");
      throw new Error("You already have an active booking.");
    }
    const timeslot = await getTimeSlotById(timeSlotId);
    if (!timeslot) {
      await pool.query("ROLLBACK");
      throw new Error("The timeslot is not available or does not exist.");
    }
    const gym = await getGymById(timeslot.gym_id);

    if (timeslot.reservedCount >= gym.capacity) {
      await pool.query("ROLLBACK");
      throw new Error("There are no more available spots in this timeslot.");
    }

    const [result] = await pool.query(
      "INSERT INTO bookings (user_id, gym_id, timeslots_id, startTime, endTime, date,
status) VALUES (?, ?, ?, ?, ?, ?, 'pending')",
      [
        userId,
        timeslot.gym_id,
        timeSlotId,
```

```
      timeslot.startTime,
      timeslot.endTime,
      timeslot.date,
      status,
    ]
  );
  await incrementReservedCount(timeSlotId);
  await pool.query("COMMIT");

  return result.insertId;
 } catch (error) {
  await pool.query("ROLLBACK");
  throw error;
 }
};

const getOldPendingBookings = async () => {
 const fiveMinutesAgo = new Date(Date.now() - 5 * 60 * 1000);
 const [oldBookings] = await pool.query(
  "SELECT * FROM bookings WHERE status = 'pending' AND createdAt < ?",
  [fiveMinutesAgo]
 );
 return oldBookings;
};

const getNextBookingForUser = async (userId) => {
 const [rows] = await pool.query(
  `SELECT b.*, g.location
  FROM bookings AS b
  JOIN gyms AS g ON b.gym_id = g.id
  WHERE b.user_id = ? AND b.status = 'confirmed' AND (b.date > CURDATE() OR
(b.date = CURDATE() AND b.startTime > CURRENT_TIME()))
  ORDER BY b.date ASC, b.startTime ASC
  LIMIT 1`,
  [userId]
 );
 return rows[0];
};

const cancelBookingById = async (bookingId) => {
 const booking = await getBookingById(bookingId);
 if (!booking) throw new Error("Booking not found");
 if (booking.status !== "cancelled") {
  await decrementReservedCount(booking.timeslots_id);
 }

 await updateBookingStatus(bookingId, "cancelled");
};
```

Pentru început, am important modulele necesar pentru conexiunea la baza de date (pool), dar și alte funcții necesare din fișierele timeSlot.js și gym.js.

Funcția createBooking creează o rezervare nouă pentru un utilizator într-o sală de fitness și un interval de timp specificat. Funcția verifică mai întâi dacă utilizatorul are deja o rezervare activă. Dacă există deja o rezervare activă, tranzacția este anulată și se aruncă o eroare. Apoi, se verifică disponibilitatea intervalului de timp și capacitatea sălii de sport. Dacă intervalul de timp nu este disponibil sau capacitatea sălii este atinsă, tranzacția este anulată și se aruncă o eroare. Dacă toate verificările trec cu succes, se creează o nouă rezervare cu statusul pending, se incrementează contorul de rezervări pentru intervalul de timp și tranzacția este finalizată cu succes. Ulterior, utilizator va avea posibilitatea de a confirma sau anula rezervarea, după ce revizuiește detaliile rezervării. Astfel, statusul se va modifica în confirmed sau cancelled, depinde de caz.

În cazul în care utilizatorul părăsește aplicația fără a confirma sau a anula o rezervare după ce a revizuit detaliile, statusul rezervării va fi pending. Pentru a evita blocarea sloturilor de timp în mod nejustificat, funcția getOldPendingBookings obține rezervările cu statusul pending care au fost create cu mai mult de 5 minute în urmă. Această funcție este utilă pentru curățarea rezervărilor neconfirmate după un anumit timp.

Funcția getNextBookingForUser returnează următoarea rezervare confirmată pentru un utilizator, ordonată după dată și ora de începere. Aceasta ajută utilizatorii să vadă care este următoarea lor rezervare activă și unde se va desfășura.

Funcția cancelBookingById anulează o rezervare specificată prin ID. Dacă rezervarea nu există, se aruncă o eroare. Dacă rezervarea nu este deja anulată, se decrementează contorul de rezervări pentru intervalul de timp asociat și se actualizează statusul rezervării la "cancelled".

Cum am precizat și înainte, fișierele modele se conectează la baza de date prin intermediul fișierului database.js și efectuează operațiile necesare pentru toate entitățile care se regăsesc în baza de date. Pe lângă fișierele pe care le-am prezentat mai sus, în folderul models mai există și fișierul gym.js. Cu toate acestea, acest fișier conține doar funcții simple cum ar fi: create, get, delete sau update, astfel că nu voi intra în amănunte.

6.3 Controlere (Controllers)

Controlerele sunt responsabile pentru gestionarea logicii aplicației și pentru manipularea cererilor HTTP. Acestea primesc cererile din partea utilizatorului, interacționează cu modelele pentru a obține sau modifica date și returnează răspunsurile adecvate.

Fișierul userController.js gestionează operațiunile legate de utilizatori, cum ar fi înregistrarea, autentificarea, resetarea parolei și altele. Voi prezenta câteva din funcțiile mai interesante care se regăsesc în acest fișier.

```
const User = require("../models/user");
const bcrypt = require("bcrypt");
const sgMail = require("@sendgrid/mail");
sgMail.setApiKey(process.env.SENDGRID_API_KEY);
const crypto = require("crypto");

const registerUser = async (req, res) => {
  const { email, name, password } = req.body;
  try {
    const existingUser = await User.findUserByEmail(email);
    if (existingUser) {
      return res.status(400).json({ message: "Email is already registered." });
    }

    const userId = await User.createUser({ email, name, password });
    res.status(201).json({ message: "User successfully created!", userId });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

const generateResetToken = () => {
  return crypto.randomBytes(20).toString("hex");
};

const forgotPassword = async (req, res) => {
  const { email } = req.body;
  const user = await User.findUserByEmail(email);
  if (!user) {
    return res.status(400).send("User does not exist.");
  }

  const resetToken = generateResetToken();
  await User.setResetPasswordToken(email, resetToken, Date.now() + 3600000);

  const resetUrl = `http://localhost:3000/reset-password/${resetToken}`;

  const msg = {
    to: email,
    from: "ilies.dragos2002@gmail.com",
    subject: "Password reset",
    text: `Please use the following link to reset your password: ${resetUrl}`,
    html: `<strong>Please use the following link to reset your password:</strong> <a
href="${resetUrl}">Reset Password</a>`,
```

```
  };

  sgMail
    .send(msg)
    .then(() => res.status(200).send("The password reset email has been sent."))
    .catch((error) => {
      console.error("Error sending email:", error);
      res.status(500).send("The password reset email could not be sent.");
    });
};

const resetPassword = async (req, res) => {
  const { token, password } = req.body;
  try {
    const user = await User.getResetPasswordToken(token);
    if (!user) {
      return res.status(400).send("Invalid or expired token.");
    }
    const success = await User.resetPassword(user.id, password);
    if (success) {
      res.send("Password has been successfully reset.");
    } else {
      res.status(500).send("Password reset failed.");
    }
  } catch (error) {
    res.status(500).send("Server error.");
  }
};
```

La începutul fișierului, sunt importate librăriile și modulele necesare cum ar fi:

- User este modelul pentru utilizatori, care conține funcțiile pentru interacțiunea cu baza de date.
- bcrypt este utilizat pentru hashing-ul parolelor.
- sgMail este folosit pentru trimiterea emailurilor prin intermediul serviciului SendGrid.
- Linia sgMail.setApiKey(process.env.SENDGRID_API_KEY) încarcă cheia API din fișierul .env, asigurând astfel securitatea și confidențialitatea acestei informații sensibile. Cheia API este utilizată pentru autentificarea cererilor de trimitere a emailurilor prin SendGrid.
- crypto este utilizat pentru generarea tokenurilor de resetare a parolei.

Funcția registerUser gestionează înregistrarea unui nou utilizator. Mai întâi verifică dacă adresa de email este deja înregistrată în baza de date. Dacă email-ul este unic, creează

un nou utilizator și returnează un mesaj de succes împreună cu ID-ul utilizatorului nou creat. Dacă apare o eroare în timpul procesului, returnează un mesaj de eroare.

Funcția generateResetToken generează un token alfanumeric aleator de 20 de caractere, utilizat pentru resetarea parolei. Aceasta funcție folosește biblioteca crypto pentru a crea un token sigur.

Funcția forgotPassword gestionează procesul de inițiere a resetării parolei. Primește adresa de email din corpul cererii și verifică dacă există un utilizator asociat cu această adresă. Dacă utilizatorul nu există, returnează un mesaj de eroare. Dacă utilizatorul este găsit, generează un token de resetare a parolei și setează termenul de expirare pentru acest token la 1 oră. Token-ul și data de expirare sunt stocate în baza de date. Un email este apoi trimis utilizatorului cu un link pentru resetarea parolei. Linkul conține token-ul de resetare. Dacă emailul nu poate fi trimis, se returnează un mesaj de eroare.

Funcția resetPassword gestionează procesul de resetare a parolei. Primește token-ul și noua parolă din corpul cererii. Verifică dacă token-ul este valid și găsește utilizatorul asociat. Dacă token-ul este invalid sau expirat, returnează un mesaj de eroare. Dacă token-ul este valid, resetează parola utilizatorului și returnează un mesaj de succes. În caz de eroare, returnează un mesaj corespunzător.

Fișierul bookingController.js gestionează operațiunile legate de rezervările utilizatorilor, cum ar fi crearea, vizualizarea detaliilor, anularea și altele. Am selectat doar câteva fragmente de cod mai elocvente.

```
const Booking = require("../models/booking");
const TimeSlot = require("../models/timeSlot");
const createBooking = async (req, res) => {
  try {
    const { user_id, timeslots_id, status } = req.body;
    const newBookingData = {
      user_id,
      timeslots_id,
      status,
    };
    const existingActiveBooking = await Booking.hasActiveBooking(user_id);
    if (existingActiveBooking) {
      return res.status(400).json({
        message: "You already have an active booking.",
      });
    }
    const bookingId = await Booking.createBooking(newBookingData);

    const newBooking = await Booking.getBookingById(bookingId);
```

```
    return res
      .status(201)
      .json({ message: "Booking created successfully!", booking: newBooking });
  } catch (error) {
    return res.status(500).json({
      message: "An error occurred while creating the booking.",
      error: error.message,
    });
  }
};
const cancelBooking = async (req, res) => {
  try {
    const bookingId = req.params.id;
    const booking = await Booking.getBookingById(bookingId);
    if (!booking) {
      return res.status(404).json({ message: "The booking was not found." });
    }
    if (booking.status === "cancelled") {
      return res
        .status(400)
        .json({ message: "The booking is already cancelled." });
    }
    const success = await Booking.updateBookingStatus(bookingId, "cancelled");
    if (success) {
      if (booking.status === "confirmed" || booking.status == "pending") {
        await TimeSlot.decrementReservedCount(booking.timeslots_id);
      }
      return res
        .status(200)
        .json({ message: "The booking was cancelled successfully!" });
    } else {
      return res.status(404).json({
        message: "The booking was not found or the status was not changed.",
      });
    }
  } catch (error) {
    return res.status(500).json({ message: "Server error: " + error.message });
  }
};
```

La începutul fișierului, sunt importate modulele necesare: Booking este modelul pentru rezervări, care conține funcțiile pentru interacțiunea cu baza de date, iar TimeSlot este modelul pentru intervalele de timp, care gestionează disponibilitatea acestora.

Funcția createBooking gestionează cererile pentru crearea unei noi rezervări. Funcția începe prin extragerea datelor necesare (user_id, timeslots_id și status) din corpul cererii (req.body). Datele extrase sunt plasate într-un obiect newBookingData. Funcția utilizează

Booking.hasActiveBooking pentru a verifica dacă utilizatorul are deja o rezervare activă. Dacă există deja o astfel de rezervare, se returnează un răspuns cu statusul 400 și un mesaj informativ. Dacă utilizatorul nu are o rezervare activă, funcția Booking.createBooking este apelată pentru a crea o nouă rezervare în baza de date. După ce rezervarea este creată, Booking.getBookingById este utilizat pentru a obține detaliile rezervării nou create. În caz de succes, se returnează un răspuns cu statusul 201 și detaliile rezervării. În caz de eroare, se returnează un răspuns cu statusul 500 și un mesaj de eroare.

Funcția cancelBooking gestionează cererile pentru anularea unei rezervări existente. Funcția începe prin extragerea id-ului rezervării din parametrii cererii (req.params.id). Booking.getBookingById este utilizat pentru a obține detaliile rezervării specificate prin id. Dacă rezervarea nu este găsită, se returnează un răspuns cu statusul 404 și un mesaj informativ. Se verifică dacă rezervarea este deja anulată. Dacă da, se returnează un răspuns cu statusul 400 și un mesaj informativ pentru a evita actualizările inutile. Dacă rezervarea nu este deja anulată, Booking.updateBookingStatus este utilizat pentru a actualiza statusul la "cancelled". Dacă rezervarea avea statusul "confirmed" sau "pending", contorul de rezervări pentru intervalul de timp asociat este decrementat utilizând TimeSlot.decrementReservedCount. În caz de succes, se returnează un răspuns cu statusul 200 și un mesaj de confirmare. În caz de eroare, se returnează un răspuns cu statusul 500 și un mesaj de eroare.

6.4 Scripturi (Scripts)

În cadrul aplicației WellnessWork Hub, am implementat două scripturi esențiale pentru gestionarea intervalelor de timp și pentru curățarea rezervărilor vechi. Aceste scripturi rulează automat și contribuie la menținerea bazei de date curate și actualizate.

Fișierul generateTimeSlots.js se ocupă de generarea automată a intervalelor de timp pentru toate sălile de sport pentru următoarele trei luni. Acest script se conectează la baza de date prin intermediul fișierului database.js și este programat să ruleze săptămânal cu ajutorul TaskScheduler. Scriptul se conectează la baza de date utilizând pool din database.js. Se obțin toate sălile de sport din baza de date. Pentru fiecare sală de sport, se generează intervale de timp pentru următoarele trei luni. Intervalele sunt generate pentru fiecare oră între 08:00 și 22:00, în fiecare zi. Înainte de a genera intervalele, se verifică dacă acestea există deja în baza de date pentru a evita duplicarea. Dacă intervalele nu există deja, acestea sunt inserate în baza de date.

```
const pool = require("../lib/database");
async function generateTimeSlotsForAllGyms() {
 try {
  const [gyms] = await pool.execute("SELECT * FROM gyms");
  for (let monthOffset = 1; monthOffset <= 3; monthOffset++) {
   const targetMonth = new Date();
   targetMonth.setMonth(targetMonth.getMonth() + monthOffset);
   targetMonth.setDate(1);
   const lastDayOfTargetMonth = new Date(targetMonth.getFullYear(),
    targetMonth.getMonth() + 1,
    0);
   const formatDateForMySQL = (date) => {
    const offset = date.getTimezoneOffset();
    const adjustedDate = new Date(date.getTime() - offset * 60 * 1000);
    return adjustedDate.toISOString().split("T")[0];
   };
   const dateStringStart = formatDateForMySQL(
    new Date(targetMonth.getFullYear(), targetMonth.getMonth(), 1)
   );
   const dateStringEnd = formatDateForMySQL(lastDayOfTargetMonth);
   const [existingTimeSlots] = await pool.execute(
    "SELECT COUNT(*) AS count FROM timeslots WHERE date BETWEEN ? AND ?",
    [dateStringStart, dateStringEnd]
   );
   if (existingTimeSlots[0].count > 0) {
    console.log(
     `Timeslots for month starting ${dateStringStart} already generated.`
    );
    continue;
   }
   for (let gym of gyms) {
    for (
     let day = targetMonth.getDate();
     day <= lastDayOfTargetMonth.getDate();
     day++
    ) {
     for (let hour = 8; hour < 22; hour++) {
      const startTime = `${hour}:00:00`;
      const endTime = `${hour + 1}:00:00`;
      const date = new Date(
       targetMonth.getFullYear(),
       targetMonth.getMonth(),
       day
      );
      const dateString = formatDateForMySQL(date);
      await pool.execute(
       "INSERT INTO timeslots (gym_id, startTime, endTime, date, reservedCount)
VALUES (?, ?, ?, ?, 0)",
```

```
            [gym.id, startTime, endTime, dateString]
          );
        }
      }
    }
  }
  console.log("Timeslots generated for the next 3 months.");
 } catch (error) {
  console.error("Error generating timeslots:", error.message);
 } finally {
  await pool.end();}
}
```

Fișierul scheduler.js se ocupă de curățarea rezervărilor cu statusul pending care sunt mai vechi de cinci minute. Acest script utilizează librăria node-cron pentru a rula periodic la fiecare cinci minute și este configurat în fișierul principal app.js. Funcția getOldPendingBookings din modelul booking.js este utilizată pentru a obține toate rezervările cu statusul pending care sunt mai vechi de cinci minute. Pentru fiecare rezervare veche, funcția cancelBookingById din booking.js este utilizată pentru a anula rezervarea și a decrementa contorul de rezervări pentru intervalul de timp asociat.

```
const cron = require("node-cron");
const bookingModel = require("../models/booking");
const startBookingCleanupTask = () => {
 cron.schedule("*/5 * * * *", async () => {
  console.log("Running booking cleanup task.");
  try {
   const oldPendingBookings = await bookingModel.getOldPendingBookings();
   for (const booking of oldPendingBookings) {
    await bookingModel.cancelBookingById(booking.id);
   }
  } catch (error) {
   console.error("Error during booking cleanup: ", error);
  }
 });
};
```

6.5 Interfața utilizatorului (Pages)

În aplicația WellnessWork Hub, interfața utilizatorului este construită cu ajutorul React, care asigură o experiență de utilizare modernă și interactivă. Fiecare pagină a aplicației este proiectată pentru a îndeplini anumite funcționalități și a ghida utilizatorii prin diverse fluxuri de activități, cum ar fi: înregistrarea, autentificarea, rezervarea de intervale orare și gestionarea profilului.

HomePage.js este pagina principală a aplicației, unde utilizatorii sunt întâmpinați cu o descriere a platformei și a facilităților oferite. Pagina include opțiuni pentru înregistrare și autentificare, precum și o secțiune dedicată antrenorilor și testimoniale de la utilizatori.

Scopul următoarelor pagini este de a colecta informații despre utilizatori prin intermediul formularelor. Astfel, SignUpPage.js este pagina de înregistrare unde utilizatorii noi pot crea un cont prin completarea unui formular cu numele, adresa de email și parola. După completare, utilizatorii sunt redirecționați către pagina de autentificare. LoginPage.js permite utilizatorilor să se autentifice în platformă prin completarea unui formular cu adresa de email și parola. De asemenea, pagina oferă un link către pagina de înregistrare și către pagina de resetare a parolei. ForgotPasswordPage.js oferă utilizatorilor posibilitatea de a-și reseta parola în cazul în care au uitat-o. Utilizatorii trebuie să introducă adresa de email pentru a primi un link de resetare a parolei. ResetPasswordPage.js este pagina unde utilizatorii pot introduce noua parolă folosind linkul de resetare primit pe email. După ce parola este resetată, utilizatorii sunt redirecționați către pagina de autentificare.

AdminDashboard.js este pagina principală unde sunt redirecționați administratori. De aici pot accesa paginile UsersCRUD.js, TimeSlotsCRUD.js, BookingsCRUD.js sau GymsCRUD.js. În cadrul acestor pagini pot efectua operații de vizualizare, creare, ștergere sau editare asupra utilizatorilor, sălilor de fitness, intervalelor orare și rezervărilor. Cu ajutorul acestor pagini, administratorii pot gestiona eficient informațiile din baza de date.

GymsPage.js este pagina unde sunt redirecționați utilizatorii după autentificarea pe platformă. Aici se afișează toate sălile de fitness disponibile și oferă utilizatorilor posibilitatea de a face rezervări. Pagina include și un profil al utilizatorului, de unde acesta poate accesa și gestiona detaliile contului său și rezervările viitoare. Această pagină este destul de importantă pentru platforma pe care am dezvoltat-o, astfel că voi prezenta și explica fragmentele de cod care se regăsesc în acest fișier. Pentru început am importat modulele și componentele necesare din React, axios (pentru cereri HTTP), react-bootstrap (pentru stilizarea componentelor) și react-router-dom (pentru navigație). De asemenea, am definit URL-ul API-ului pentru a prelua datele despre sălile de fitness.

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import {Button, Card, Container, Row, Col, Modal, Dropdown, Form, } from "react-
bootstrap";
import { useNavigate } from "react-router-dom";
const API_URL = "http://localhost:3001/api/gyms";
```

Apoi, funcția asincronă fetchGyms face o cerere GET la API pentru a prelua datele despre sălile de fitness și returnează aceste date. În caz de eroare, aceasta este gestionată și afișată în consolă.

```
const fetchGyms = async () => {
  try {
    const response = await axios.get(API_URL);
    return response.data;
  } catch (error) {
    console.error("Eroare la preluarea sălilor de fitness:", error);
    throw error;
  }
};
```

Funcția GymsPage este componenta principală a paginii și gestionează afișarea sălilor de fitness disponibile, profilul utilizatorului și rezervările acestuia. În această funcție, am definit mai multe stări folosind hook-ul useState și am folosit useEffect pentru a prelua datele despre sălile de fitness atunci când componenta este montată. De asemenea, am creat mai multe funcții de manipulare pentru a gestiona diferite acțiuni ale utilizatorului.

State-uri (useState) utilizate sunt :

- gyms: stochează lista sălilor de fitness.
- showProfileModal: controlează afișarea modalului pentru profilul utilizatorului.
- profileData: stochează datele de profil ale utilizatorului.
- nextBooking: stochează următoarea rezervare a utilizatorului.
- showNoBookingModal: controlează afișarea modalului pentru cazul în care nu există rezervări viitoare.
- userId: obține ID-ul utilizatorului din localStorage.

Ca și Hooks avem useEffect care preia datele despre sălile de fitness când componenta este montată.

```
function GymsPage() {
  const [gyms, setGyms] = useState([]);
  const [showProfileModal, setShowProfileModal] = useState(false);
  const [profileData, setProfileData] = useState({
    email: "",
    name: "",
    newPassword: "",
  });

  const navigate = useNavigate();
  const [nextBooking, setNextBooking] = useState(null);
  const [showNoBookingModal, setShowNoBookingModal] = useState(false);
```

```
const userId = localStorage.getItem("userId");
const formatDate = (dateString) => {
  const options = {
    year: "numeric",
    month: "long",
    day: "numeric",
  };
  return new Date(dateString).toLocaleDateString(undefined, options);
};

useEffect(() => {
  const getGyms = async () => {
    try {
      const data = await fetchGyms();
      setGyms(data);
    } catch (error) {
      console.log(error);
    }
  };
  getGyms();
}, []);
```

Funcții de manipulare sunt:

- handleGymSelect: navighează la pagina cu intervalele orare disponibile pentru sala de fitness selectată.
- fetchProfileData: face cererea pentru a prelua datele de profil ale utilizatorului și afișează modalul pentru profil.
- handleProfileUpdate: trimite datele actualizate ale profilului utilizatorului către server.
- handleLogout: navighează la pagina de logout.
- handleMyBookingsClick: face cererea pentru a prelua următoarea rezervare a utilizatorului și afișează modalul corespunzător.
- handleCancelBooking: anulează rezervarea selectată și actualizează starea.

```
const handleGymSelect = (gymId) => {
  navigate(`/gyms/${gymId}/timeslots`);
};
const fetchProfileData = async () => {
  try {
    const response = await axios.get("/api/users/profile", {
      withCredentials: true,
    });
    setProfileData(response.data);
    setShowProfileModal(true);
```

```
  } catch (error) {
    console.error("Error fetching profile data", error);
  }
};
const handleProfileUpdate = async (event) => {
  event.preventDefault();
  try {
    const response = await axios.post(
      "/api/users/updatedProfile",
      profileData,
      { withCredentials: true }
    );
    alert(response.data.message);
    setShowProfileModal(false);
  } catch (error) {
    console.error("Error updating profile", error);
  }
};
const handleLogout = () => {
  navigate("/logout");
};

const handleMyBookingsClick = async () => {
  try {
    const response = await axios.get(`/api/users/nextBooking/${userId}`, {
      withCredentials: true,
    });
    if (response.data) {
      setNextBooking(response.data);
      setShowNoBookingModal(false);
    } else {
      setShowNoBookingModal(true);
    }
  } catch (error) {
    console.error("Error fetching next booking", error);
    setShowNoBookingModal(true);
  }
};

const handleCancelBooking = async (bookingId) => {
  try {
    await axios.post(
      `/api/bookings/cancel/${bookingId}`,
      {},
      { withCredentials: true }
    );
    alert("Booking successfully cancelled!");
    setNextBooking(null);
```

```
  } catch (error) {
    console.error("Error cancelling booking", error);
  }
};
```

Ultima parte a funcției reprezintă returnarea JSX. JSX (JavaScript XML) este o caracteristică esențială a React care facilitează scrierea de cod ușor de înțeles și de întreținut, combinând marcajul HTML cu logica JavaScript într-un mod natural și eficient. Aceasta face ca dezvoltarea componentelor de interfață să fie mai simplă și mai rapidă. Aici se construiește interfața utilizatorului folosind componente din react-bootstrap. Conținutul este împărțit în mai multe secțiuni: dropdown-ul pentru profilul utilizatorului, rezervări și logout, lista sălilor de fitness, afișată folosind componentele Row, Col și Card și modalurile pentru gestionarea profilului și rezervărilor.

După ce utilizatorul a decis care este sala în care dorește să se antreneze, apasă pe butonul „Book Now” iar acesta îl redirecționează în pagina TimeSlotsPage.js. Aici, utilizatorul poate selecta o dată specifică cu ajutorul unui calendar. Pagina va afișa intervalele orare disponibile pentru data selectată. Utilizatorul poate face o rezervare pentru un interval orar disponibil. După ce rezervarea este efectuată, utilizatorul primește un mesaj de confirmare și este redirecționat către BookingConfirmationPage.js.

BookingConfirmationPage.js. afișează detaliile rezervării (locație, dată, interval orar) după ce utilizatorul a făcut o rezervare în TimeSlotsPage.js. Utilizatorul poate revizui detaliile rezervării. Dacă utilizatorul apasă pe butonul „Confirm Booking”, rezervarea este confirmată și utilizatorul primește un mesajul de confirmare. Însă, dacă utilizatorul apasă pe butonul „Cancel Booking”, rezervarea este anulată și utilizatorul primește un mesaj de anulare. În ambele cazuri, utilizatorul este redirecționat înapoi la GymsPage.js.

Pagina LogoutPage.js este defapt un modal care poate fi accesat atât de utilizatori normali, din meniul dropdown din GymsPage.js, cât și de administratori din AdminDashboard. Dacă utilizatorul apasă pe „Close”, modalul se închide și utilizatorul rămâne conectat pe platformă. Dacă utilizatorul apasă pe „Logout”, acesta este deconectat și redirecționat către HomePage.js.

7. Testarea

Testarea software este un pas esențial în procesul de dezvoltare al unei aplicații. Prin acest mod, se asigură faptul că funcționalitățile implementate îndeplinesc cerințele inițiale și că aplicația funcționează conform așteptărilor. În cadrul aplicației WellnessWork Hub, testarea a fost efectuată în mod riguros cu ajutorul a diverse instrumente și tehnici pentru a acoperi toate aspectele critice ale funcționalității, performanței și uzabilității.

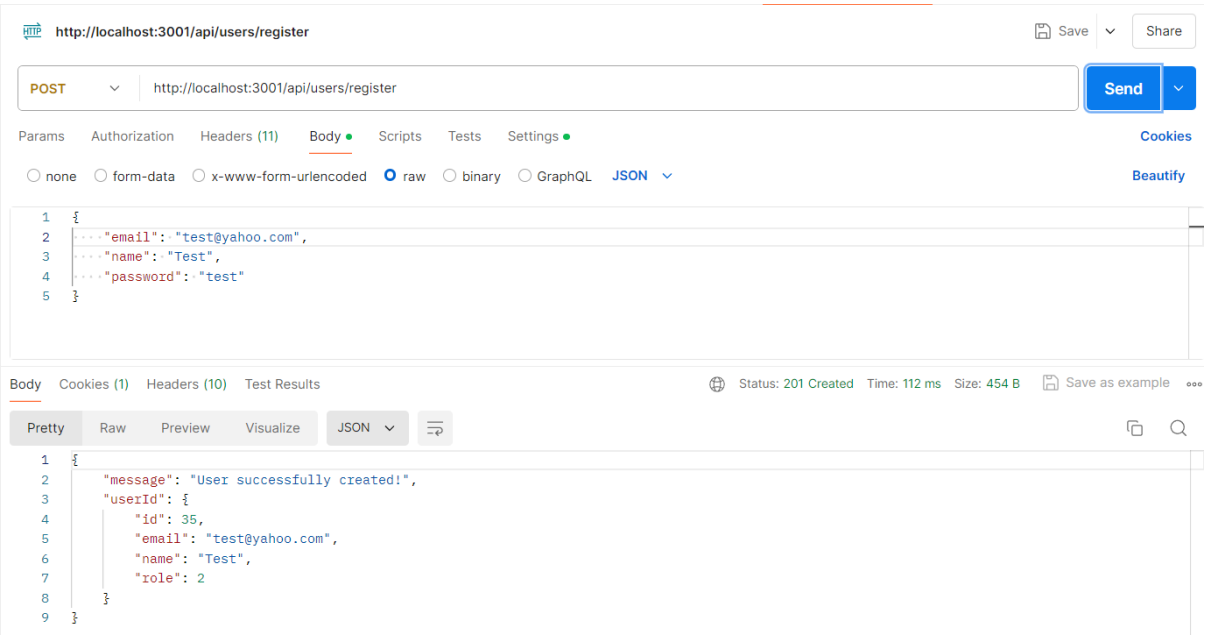
În această secțiune, voi descrie metodele și instrumentele utilizate pentru a testa aplicația, precum și rezultatele obținute. Testarea s-a concentrat pe verificarea corectitudinii cererilor HTTP folosind Postman, automatizarea testelor de interfață cu Selenium și evaluarea performanței și uzabilității aplicației prin măsurători specifice.

7.1 Verificarea funcționalităților back-end cu Postman

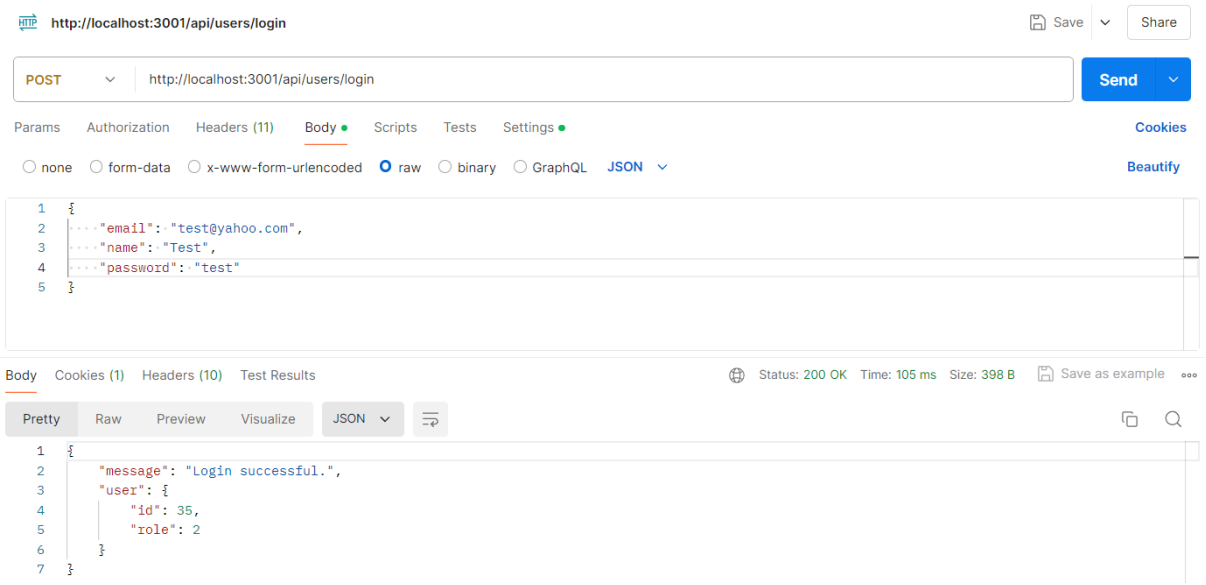
Postman este un instrument folosit pentru dezvoltarea și testarea API-urilor, care mi-a permis să trimit cereri HTTP și să analizez răspunsurile serverului. Testarea cererilor HTTP, înainte de a implementa interfața utilizatorului, a fost crucială pentru a asigura comunicarea corectă între frontend și backend, dar și pentru a verifica funcționalitatea logicii de business.

În continuare, voi prezenta fluxul cererilor trimise de aplicație către server prin acțiunea utilizatorului din momentul în care acesta se înregistrează și se autentifică pe platformă, până când face o rezervare și părăsește platforma. De asemenea, voi furniza și poze din Postman pentru a scoate în evidență timpul de răspuns al cererilor. Astfel, prima cerere pe care o trimite un client nou către server este cea de înregistrare. Cererea este una de tipul POST și permite crearea unui nou cont de utilizator.

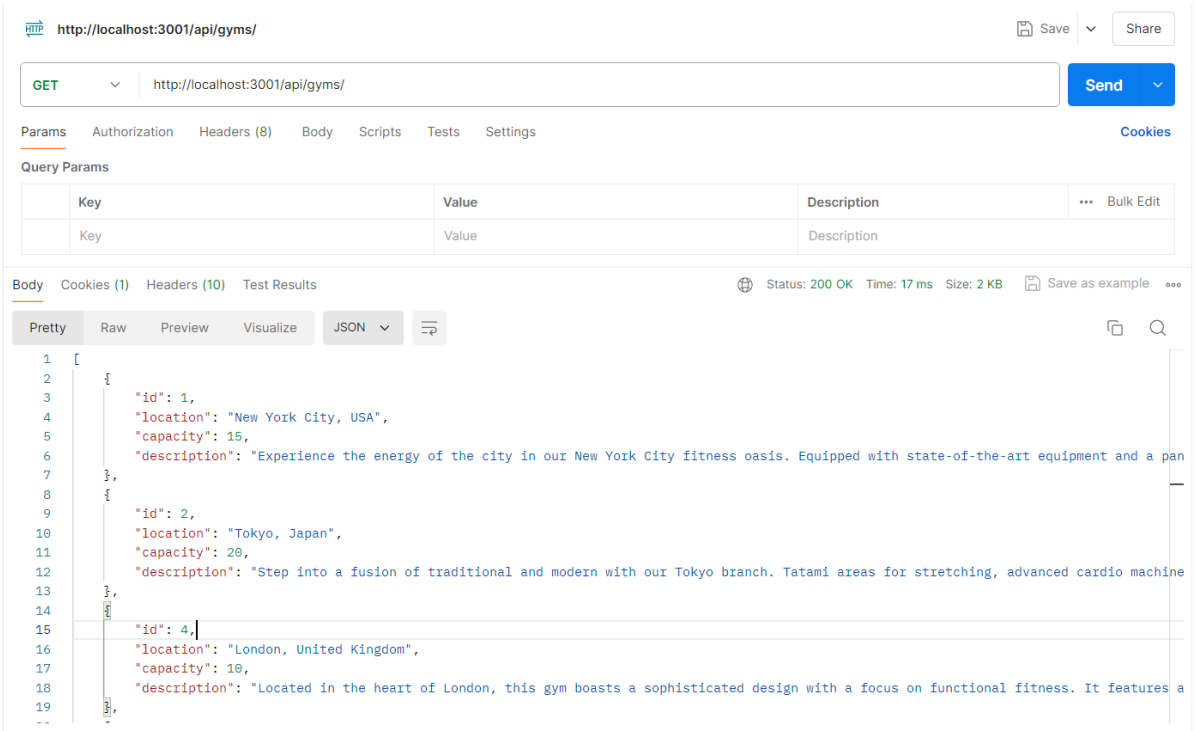
În exemplul de mai jos, corpul cererii conține informațiile necesare pentru înregistrarea unui nou utilizator: adresa de email, nume și parolă. Aceste informații sunt trimise către endpoint-ul „/api/users/register”. După trimiterea cererii, serverul răspunde cu un mesaj care confirmă crearea contului. Mesajul de răspuns include detalii despre utilizatorul creat, cum ar fi id-ul utilizatorului, adresa de email și rolul acestuia. Toți utilizatorii vor avea implicit rolul egal cu valoarea 2 (utilizatori simpli), însă acesta poate fii schimbat apoi de către administrator. În plus, răspunsul serverului include un cod de stare 201 (Created), care evidențiază că utilizatorul a fost creat cu succes. Timpul de răspuns pentru această cerere a fost de 112 ms, ceea ce demonstrează o performanță bună a serverului în gestionarea cererii de înregistrare.



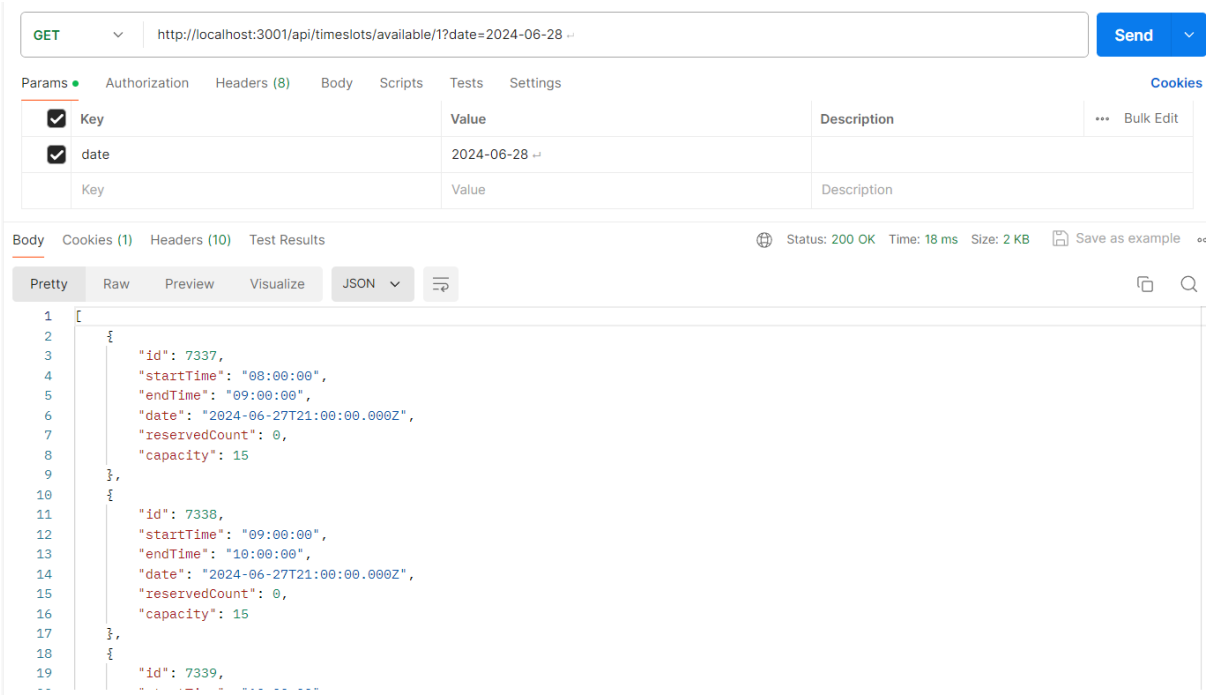
După înregistrarea unui cont nou, următoarea cerere importantă este cea de autentificare. Cererea este una de tipul POST și permite utilizatorului să se autentifice pe platformă. Și aici, corpul cererii conține informațiile necesare pentru autentificare: adresa de email și parola. Aceste informații sunt trimise către endpoint-ul „/api/users/login”. După trimiterea cererii, serverul răspunde cu un mesaj care confirmă autentificarea reușită. Răspunsul serverului include un cod de stare 200 (OK), indicând că autentificarea a fost efectuată cu succes. Timpul de răspuns pentru această cerere a fost de 105 ms, ceea ce reflectă o performanță eficientă a serverului în gestionarea cererii de autentificare.



După ce utilizatorul s-a autentificat pe platformă, acesta este redirecționat către pagina GymsPage. Aici sunt afișate toate informațiile despre sălile de fitness disponibile, inclusiv locația, capacitatea și descrierea fiecărei săli. Această pagină trimite la server cererea din poza de mai jos pentru a obține informațiile despre sălile de fitness. Cererea este una de tipul GET și este trimisă către endpoint-ul „/api/gyms". Această cerere returnează o listă cu toate sălile de fitness disponibile pe platformă. Serverul răspunde cu un mesaj care conține detalii despre fiecare sală de fitness. Din nou, răspunsul serverului include un cod de stare 200 (OK), iar timpul de răspuns pentru această cerere a fost de 17 ms. Se poate observa o performanță excelentă a serverului în gestionarea cererii de obținere a informațiilor despre sălile de fitness.



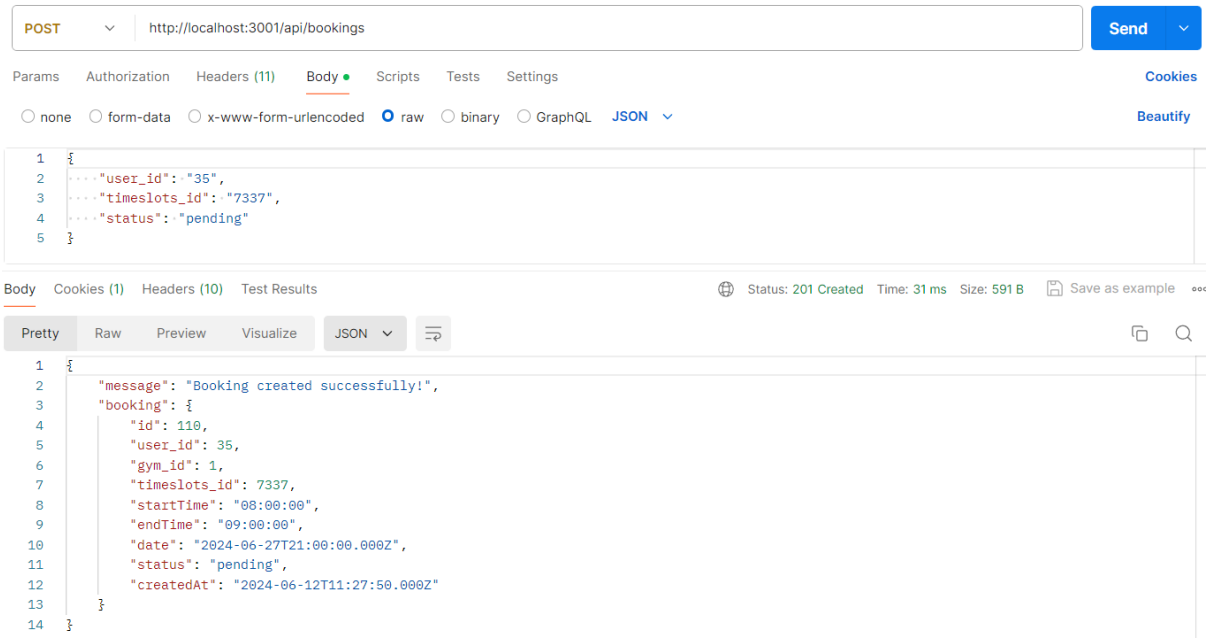
După ce utilizatorul selectează o sală de fitness din pagina GymsPage și apasă pe butonul "Book Now", acesta este redirecționat către pagina TimeSlotsPage, unde poate selecta o dată și vizualiza intervalele orare disponibile pentru sala respectivă. Pentru a obține aceste informații, aplicația trimite o cerere GET la server. Cererea este realizată către endpoint-ul „/api/timeslots/available/:gymId" și include un parametru de query pentru dată. În poza de mai jos, se poate observa răspunsul serverului care conține detalii despre intervalele orare disponibile pentru sala de fitness specificată (gymId=1) la data indicată (2024-06-28). Timpul de răspuns pentru această cerere a fost de 18 ms, ceea ce demonstrează, din nou, o performanță excelentă a serverului în gestionarea cererii.



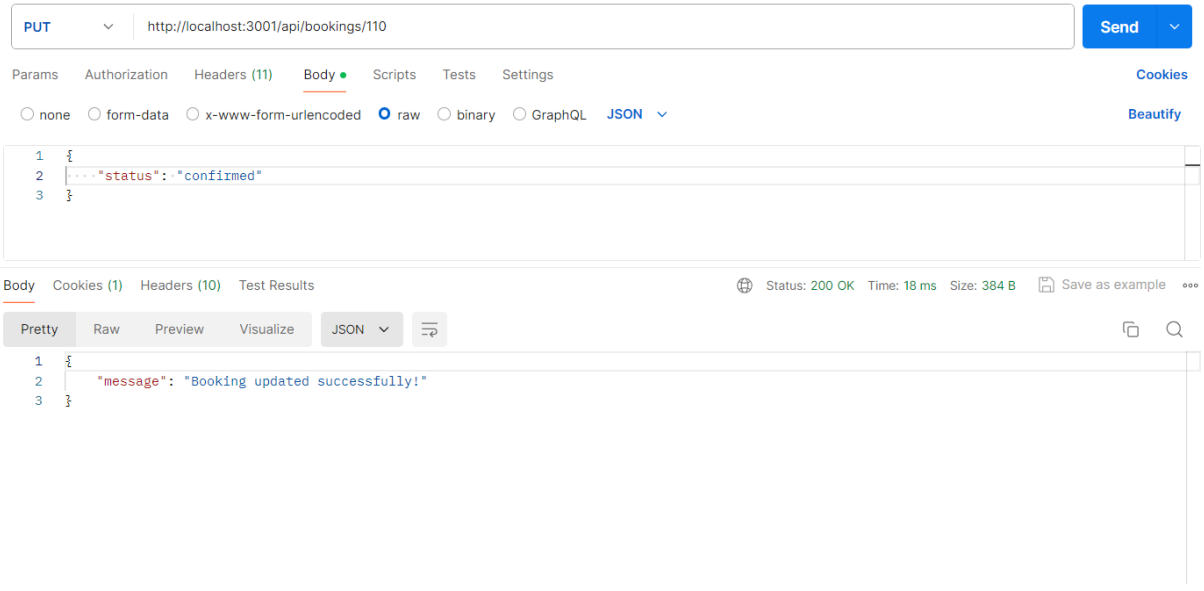
Dacă utilizatorul a selectat un interval orar disponibil și a apăsat pe butonul "Book Now" în pagina TimeSlotsPage.js, aplicația trimite o cerere de tip POST către server pentru a crea o rezervare. Această cerere este trimisă către endpoint-ul „/api/bookings” și include informațiile necesare pentru crearea unei rezervări. Cererea este configurată să trimită datele în format JSON și conține următoarele câmpuri: user_id, care reprezintă id-ul utilizatorului care face rezervarea, timeslots_id, care este id-ul intervalului orar selectat, și status, care este inițial setat la "pending". Aceste date sunt esențiale pentru a asocia rezervarea cu utilizatorul și intervalul orar specificat.

Odată trimisă cererea, serverul o procesează și răspunde cu un mesaj de confirmare, indicând că rezervarea a fost creată cu succes. Răspunsul serverului include un obiect JSON care conține detalii despre rezervarea nou creată, cum ar fi id-ul rezervării, id-ul utilizatorului, id-ul sălii de fitness asociate, id-ul intervalului orar rezervat, orele de început și sfârșit ale intervalului, data rezervării, starea rezervării (care rămâne "pending" până la confirmare) și data și ora la care a fost creată rezervarea.

În exemplul de mai jos, cererea a primit un răspuns cu codul de status 201 (Created), confirmând că rezervarea a fost creată cu succes. Mesajul de răspuns include timpul de răspuns, care a fost de 21 ms, demonstrând o performanță eficientă a serverului în gestionarea acestei cereri.

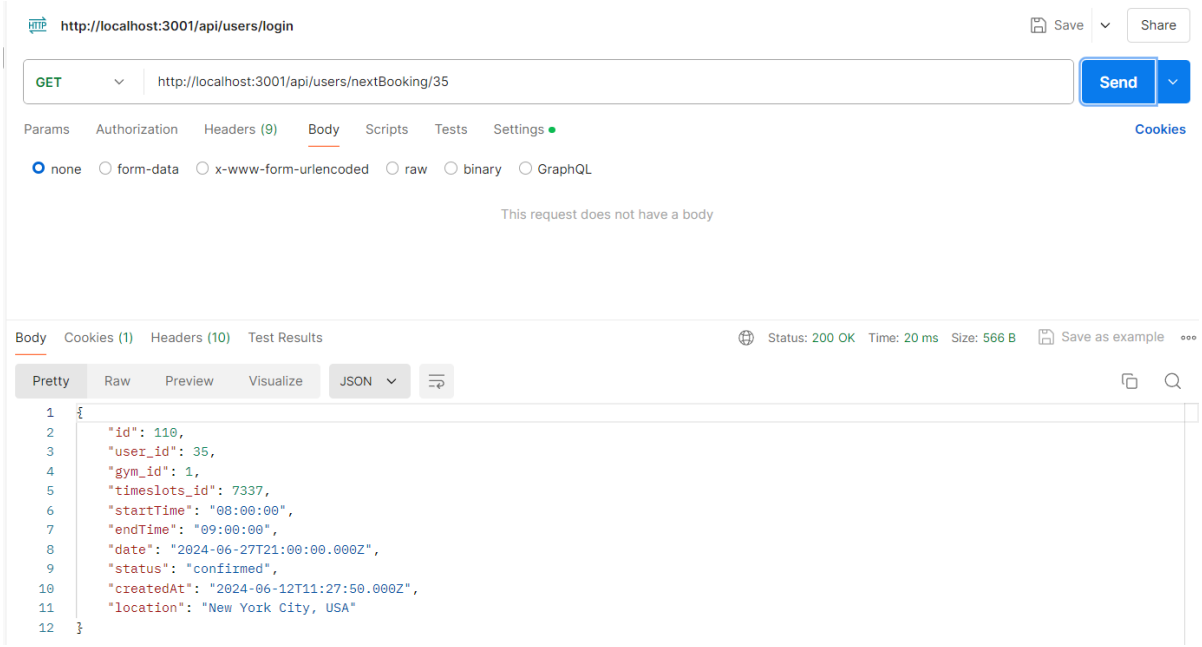


După crearea rezervării, utilizatorul este redirecționat către pagina BookingConfirmationPage.js. Aici, utilizatorul poate vedea detaliile rezervării, inclusiv locația sălii, data și intervalul orar ales. Pentru a finaliza rezervarea, utilizatorul trebuie să apese pe butonul "Confirm Booking". Apăsarea acestui buton trimite o cerere de tip PUT către server pentru a actualiza statusul rezervării la "confirmed". În poza de mai jos, se observă că cererea este trimisă către endpoint-ul „/api/bookings/110” (unde 110 este id-ul rezervării) și are corpul "status": "confirmed". Serverul răspunde cu un mesaj care confirmă actualizarea rezervării. Timpul de răspuns pentru această cerere a fost de 18 ms.

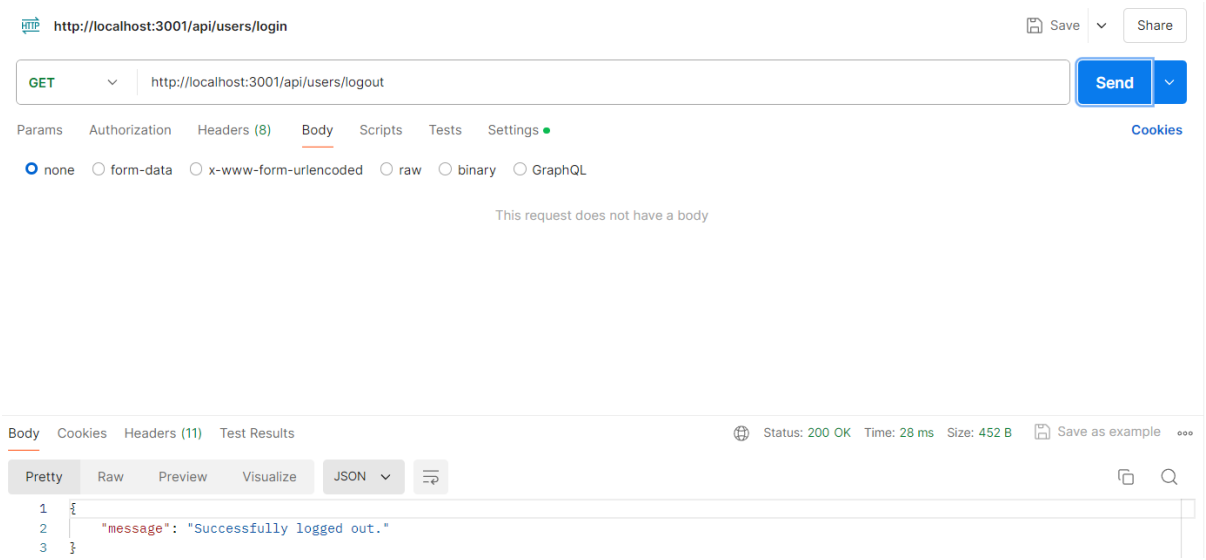


Apoi, după confirmarea rezervării, utilizatorul este redirecționat la pagina GymsPage.js. Aici, utilizatorul poate accesa și meniul dropdown Profile. Acest meniu are mai multe opțiuni precum: My Profile, My Bookings și Logout. Pentru a vizualiza detaliile următoarei rezervării confirmate, trebuie sa acceseze meniul My Bookings. Accesarea opțiunii "My Bookings" trimite o cerere de tip GET către server pentru a obține detaliile următoarei rezervări a utilizatorului. Cererea este trimisă către endpoint-ul „/api/users/nextBooking/35”, unde 35 este id-ul utilizatorului. Cererea nu are un corp, ci doar parametrii de autentificare și identificare ai utilizatorului.

Răspunsul serverului conține detaliile complete ale rezervării confirmate, precum: id-ul rezervării, id-ul utilizatorului, id-ul sălii, id-ul intervalului orar, ora de început și de sfârșit, data, statusul rezervării, momentul creării rezervării și locația sălii. În imaginea de mai jos se pot observa aceste aspecte. Răspunsul serverului are statusul 200 OK, iar timpul de răspuns a fost de 20 ms, demonstrând o performanță rapidă.



În final, după ce utilizatorul a confirmat rezervarea și a vizualizat detaliile acesteia, acesta poate decide să se delogheze din aplicație. Pentru a face acest lucru, utilizatorul trebuie să acceseze opțiunea Logout din meniul dropdown. Accesarea opțiunii Logout trimite o cerere de tip GET către server pentru a închide sesiunea curentă a utilizatorului. Cererea este trimisă către endpoint-ul „/api/users/logout”. Cererea nu are un corp de date. Serverul răspunde cu un mesaj care confirmă delogarea utilizatorului. În exemplul prezentat, timpul de răspuns pentru această cerere a fost de 28 ms.



Testarea API-urilor din back-end utilizând Postman a demonstrat că serverul gestionează cererile într-un mod eficient și rapid. Fiecare cerere, de la înregistrarea unui nou utilizator și autentificarea acestuia, până la realizarea și confirmarea unei rezervări, a avut timpi de răspuns foarte buni. Timpii de răspuns prezentați au avut valori sub 150ms ceea ce evidențiază performanța bună a serverului și capacitatea acestuia de a răspunde rapid la cererile utilizatorilor. Astfel, am încercat să asigur o experiență fluidă și eficientă.

7.2 Testarea interfeței cu Selenium

În acest subcapitol, voi descrie procesul de testare a interfeței aplicației web pe care l-am realizat cu ajutorul Selenium WebDriver. Scopul acestei testări este de a asigura funcționalitatea corectă a aplicației și de a detecta eventualele erori care pot apărea în interacțiunea utilizatorilor cu aplicația.

Pentru a utiliza Selenium WebDriver în Python a trebuit să instalez pachetul Selenium. Acest lucru l-am realizat folosind pip, managerul de pachete pentru Python. Comanda pentru instalare este următoarea: „pip install selenium”. Selenium WebDriver necesită un driver specific pentru browserul utilizat, astfel că am descărcat driverul ChromeDriver de pe site-ul oficial. După descărcarea arhivei ChromeDriver, aceasta trebuie extrasă într-un director accesibil. În final, am implementat și rulat scriptul necesar pentru testare, în care am specificat și calea către executabilul ChromeDriver în variabila service.

În continuare voi prezenta codul pe care l-am implementat pentru a putea testa interfața aplicației web. De asemenea, voi furniza și poze cu rezultatul fiecărei acțiuni realizat de script în browser. Prima parte a codului este reprezentată de configurarea inițială. În această secțiune, am configurat opțiunile pentru browserul Chrome și am setat calea către

executabilul ChromeDriver. De asemenea, am definit URL-ul de bază al aplicației și credențialele utilizatorului pentru testare.

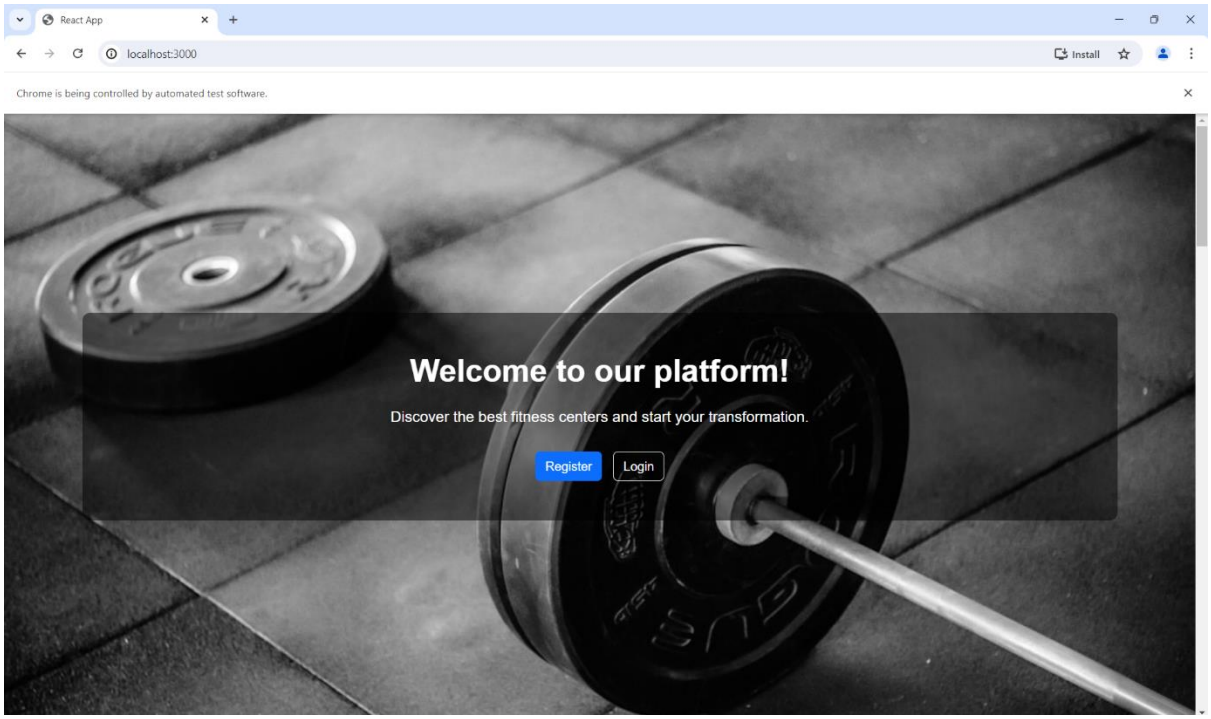
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from datetime import datetime, timedelta
import time

# Configurarea opțiunilor Chrome
chrome_options = Options()
chrome_options.add_argument("--start-maximized")

# Configurarea driverului Chrome
service = Service(r"C:\Users\Dragos\Downloads\chromedriver-win64\chromedriver-
win64\chromedriver.exe")
driver = webdriver.Chrome(service=service, options=chrome_options)

# Definirea URL-ului și a credențialelor
base_url = "http://localhost:3000"
email = "testuser@yahoo.com"
password = "password"
name = "Test User"
```

Mai apoi, am implementat o funcție pentru gestionarea alertelor. Această funcție așteaptă apariția unei alerte în browser și o gestionează automat apăsând pe butonul „OK” al alertei.

```
def handle_alert():
    try:
        WebDriverWait(driver, 5).until(EC.alert_is_present())
        alert = driver.switch_to.alert
        alert.accept()
    except:
        pass
```

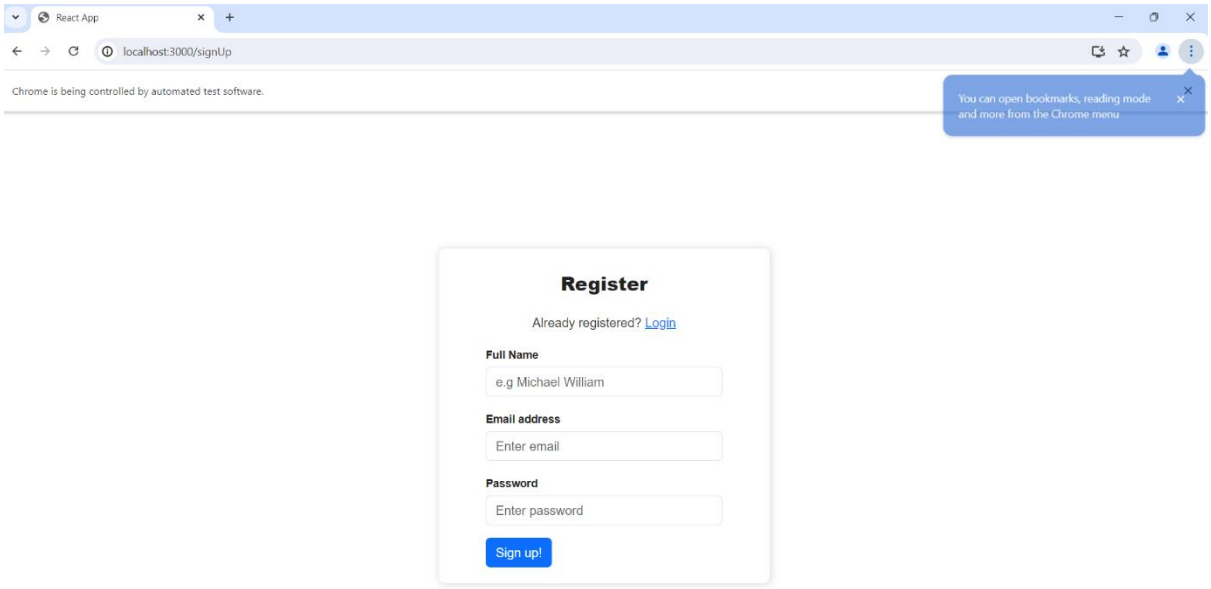
Următoarea funcție este funcția open_home_page. Aceasta deschide pagina de start a aplicației web în browser și așteaptă până când butonul „Register” devine vizibil. Se poate observa și în imaginea de mai jos, pagina de start a aplicației web a fost deschisă, iar butonul „Register” e prezent.

```
def open_home_page():
    driver.get(base_url)
    WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH,
"//button[text()='Register']")))
    time.sleep(2)
```



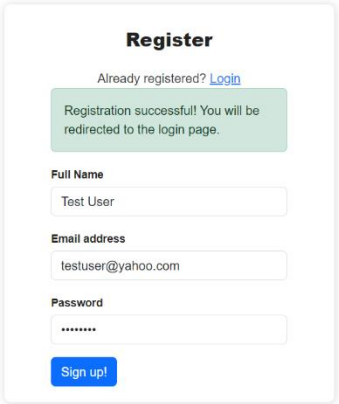
Mai apoi, funcția navigate_to_register face click pe butonul „Register” și așteaptă redirecționarea către pagina de înregistrare. În imagine se poate vedea cum scriptul a navigat la pagina de înregistrare, unde sunt afișate câmpurile necesare pentru înregistrare.

```
def navigate_to_register():
    register_button = driver.find_element(By.XPATH, "//button[text()='Register']")
    register_button.click()
    WebDriverWait(driver, 10).until(EC.url_to_be(f"{base_url}/signUp"))
    time.sleep(2)
```



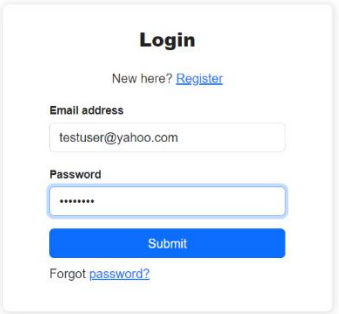
Funcția register completează câmpurile necesare pentru înregistrarea unui nou utilizator (nume, email și parolă) și face click pe butonul „Sign up!”. Informațiile despre utilizatorul de test au fost definite la începutul scriptului. După aceasta, apare un mesaj de succes, iar apoi utilizatorul este redirecționat către pagina de autentificare. Din poza de mai jos, se poate observa cum scriptul a completat câmpurile necesare pentru înregistrare și a făcut click pe butonul „Sign up!”.

```
def register():
    WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.NAME, "email")))
    email_field = driver.find_element(By.NAME, "email")
    name_field = driver.find_element(By.NAME, "name")
    password_field = driver.find_element(By.NAME, "password")
    register_button = driver.find_element(By.XPATH, "//button[text()='Sign up!']")

    name_field.send_keys(name)
    time.sleep(2)
    email_field.send_keys(email)
    time.sleep(2)
    password_field.send_keys(password)
    time.sleep(2)
    register_button.click()

    # Așteptarea redirecționării
    WebDriverWait(driver, 10).until(EC.url_to_be(f"{base_url}/login"))
    time.sleep(2)
```



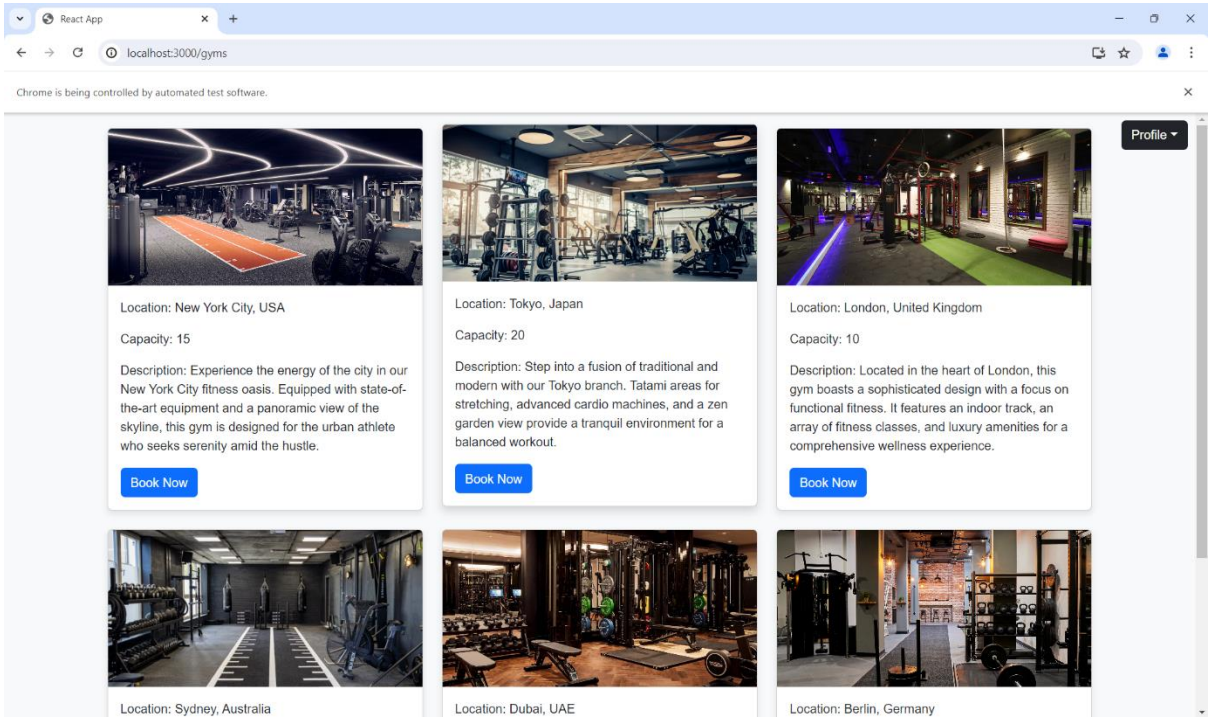
Dacă utilizatorul a creat un cont valid, din pagina de SignUp este redirecționat către pagina de Login. În această pagină, apare un alt formular care este completat cu ajutorul funcției de login. Funcția login introduce emailul și parola utilizatorului, apoi face click pe butonul „Submit”. După aceasta, așteaptă redirecționarea către pagina unde apar sălile de fitness. Din poza vedem cum scriptul a introdus datele pentru autentificare și a făcut click pe butonul „Submit”.

```
def login():
    email_field = driver.find_element(By.NAME, "email")
    password_field = driver.find_element(By.NAME, "password")
    login_button = driver.find_element(By.XPATH, "//button[text()='Submit']")

    email_field.send_keys(email)
    time.sleep(2)
    password_field.send_keys(password)
    time.sleep(2)
    login_button.click()

    # Așteptarea redirecționării
    WebDriverWait(driver, 10).until(EC.url_to_be(f"{base_url}/gyms"))
    time.sleep(2)
```



În cazul în care utilizatorul a completat corect datele pentru autentificare, acționarea butonul „Submit” din pagina de Login îl va redirecționa în pagina de prezentare a sălilor de fitness. Aici, am implementat funcția select_gym care selectează prima sală de fitness disponibilă și face clic pe butonul „Book Now”. Așteaptă apoi redirecționarea către pagina de intervale orare. În imaginea de mai jos se pot observa sălile de fitness, iar scriptul a selectat o sală de fitness și a navigat la pagina de intervale orare.

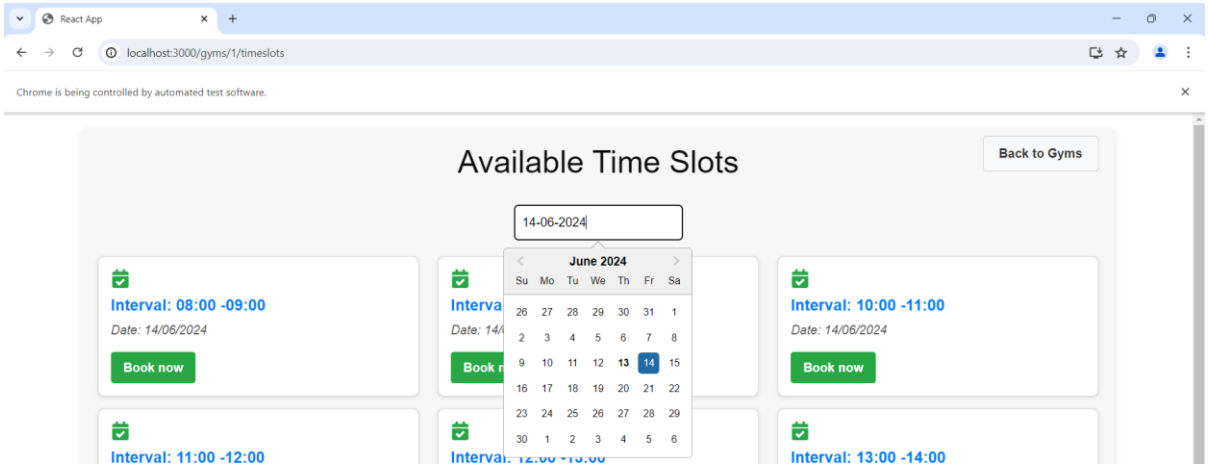
```
def select_gym():
    gym_cards = driver.find_elements(By.CLASS_NAME, "gym-card")
    if gym_cards:
        gym_card = gym_cards[0]  # Selectarea primei săli
        book_now_button = gym_card.find_element(By.XPATH, ".//button[text()='Book
Now']")
        book_now_button.click()

        # Așteptarea redirecționării la pagina de intervale orare
        WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.CLASS_NAME, "time-slot-card")))
        time.sleep(2)
```

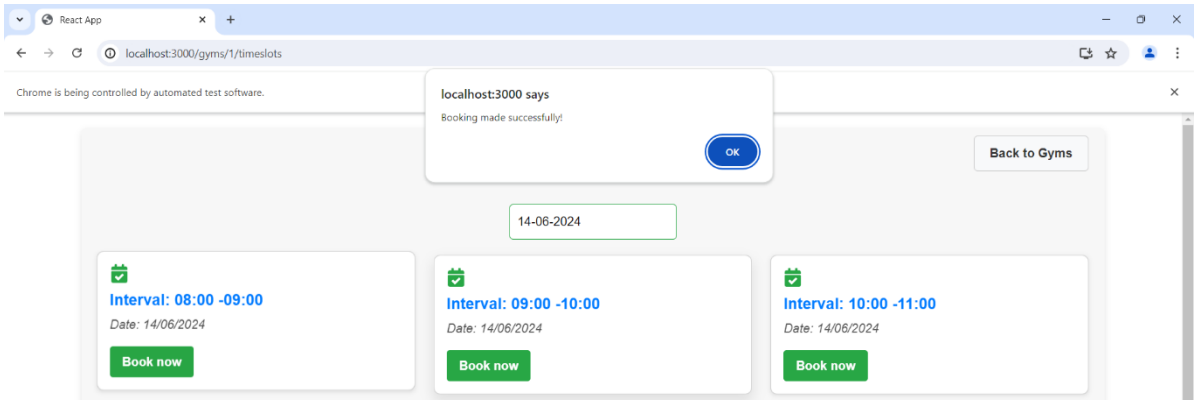


Următoarea funcție din script este funcția select_timeslot. Aceasta acționează în pagina cu intervalele orare specifice pentru sala de fitness selectată anterior. Funcția deschide datepicker-ul și selectează data de mâine. Apoi face click pe primul buton „Book now” și gestionează orice alertă apărută. Așteaptă redirecționarea către pagina de confirmare a

rezervării. Din pozele de mai jos, se poate vedea cum scriptul a selectat data de mâine din calendar, apoi a apăsat pe butonul „Book now” și a gestionat alerta apărută.

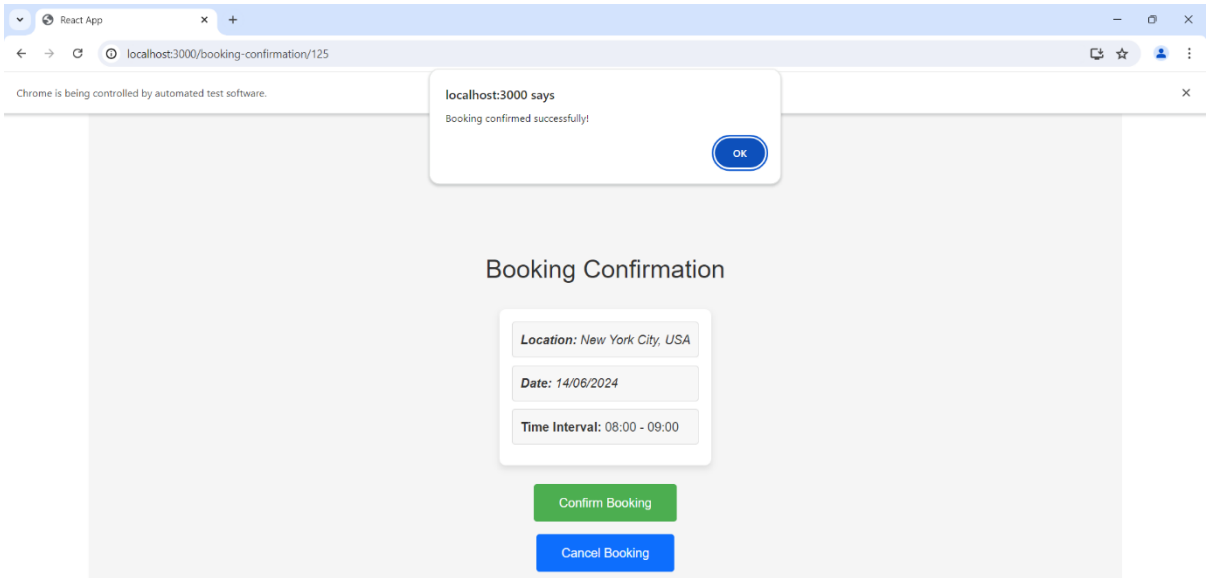
```
def select_timeslot():
    date_picker = driver.find_element(By.CLASS_NAME, "datepicker-custom")
    date_picker.click()
    time.sleep(1)

    # Selectarea datei de mâine
    tomorrow = datetime.now() + timedelta(days=1)
    tomorrow_str = tomorrow.strftime("%d")

    # Asigurarea vizibilității datepickerului și încercarea din nou dacă nu este găsit
    WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.CLASS_NAME,
"react-datepicker__day")))
    date_element = driver.find_element(By.XPATH, f"//div[contains(@class, 'react-
datepicker__day') and text()='{tomorrow_str}']")
    date_element.click()
    time.sleep(2)

    # Click pe primul buton "Book now"
    book_now_button = driver.find_element(By.XPATH, "(//button[text()='Book now'])[1]")
    book_now_button.click()

    # Gestionarea alertei
    handle_alert()

    # Așteptarea redirecționării la confirmarea rezervării
    WebDriverWait(driver, 10).until(EC.url_contains(f"{base_url}/booking-confirmation/"))
    time.sleep(2)
```





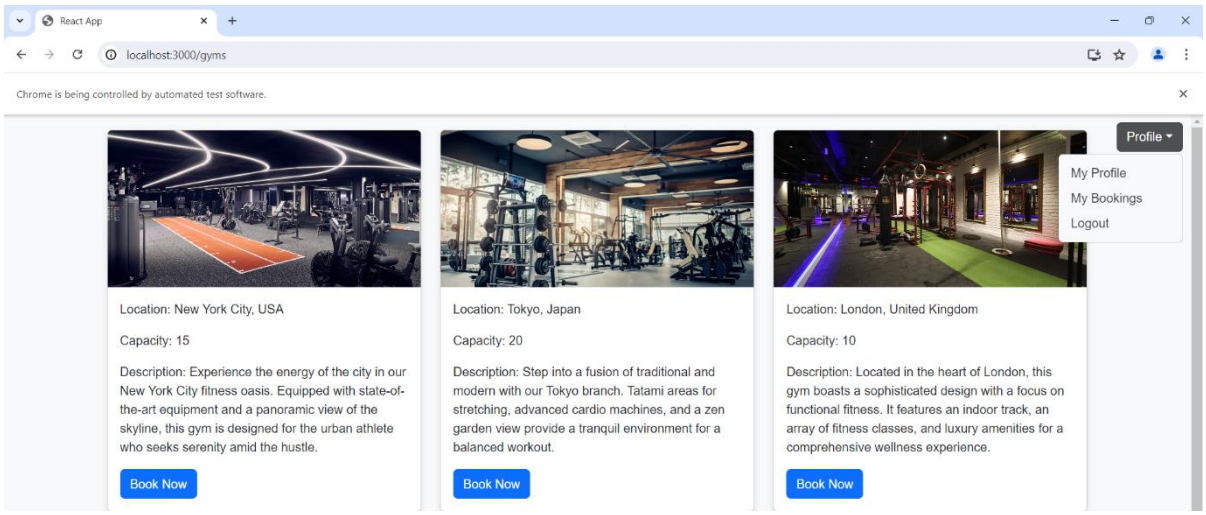
Pentru a asigura o gestionare cât mai corectă a rezervărilor, am implementat și o pagină de confirmare. Aici, utilizatorii pot să revizuiască datele despre rezervarea pe care doresc să o facă. Rezervarea va deveni activă doar în momentul în care se va apăsa de butonul „Confirm Booking”. Funcția confirm_booking, din scriptul de test, așteaptă apariția butonului „Confirm Booking”, face click pe acesta și gestionează alerta apărută. Apoi, așteaptă redirecționarea către pagina sălilor de fitness.

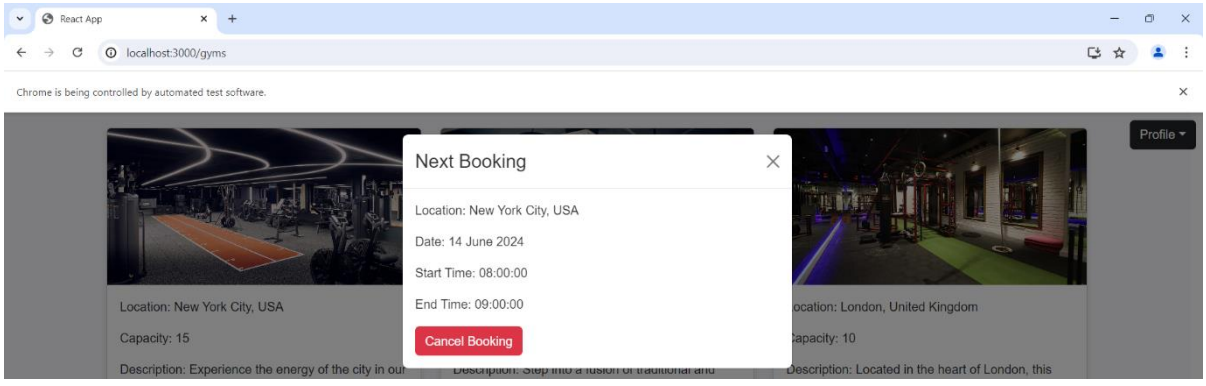
```
def confirm_booking():
    WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, 
"//button[text()='Confirm Booking']")))
    confirm_button = driver.find_element(By.XPATH, "//button[text()='Confirm Booking']")
    confirm_button.click()
    time.sleep(20)

    # Gestionarea alertei
    handle_alert()

    # Așteptarea redirecționării la pagina sălilor de fitness
    WebDriverWait(driver, 10).until(EC.url_to_be(f"{base_url}/gyms"))
    time.sleep(2)
```



După ce rezervarea a fost confirmată sau anulată, utilizatorul este redirecționat în pagina de prezentare a sălilor de fitness. În această pagină, există un meniu dropdown de unde utilizatorul poate să își verifice datele de profil, eventual să modifice ceva dacă a completat greșit. De asemenea, poate să vizualizeze datele despre următoarea sa rezervare activă sau să părăsească aplicația. În script, am introdus funcția view_my_bookings. Numele este unul destul de sugestiv, funcția accesează meniul „Profile” și selectează opțiunea „My Bookings”. După vizualizarea rezervării, închide meniul.

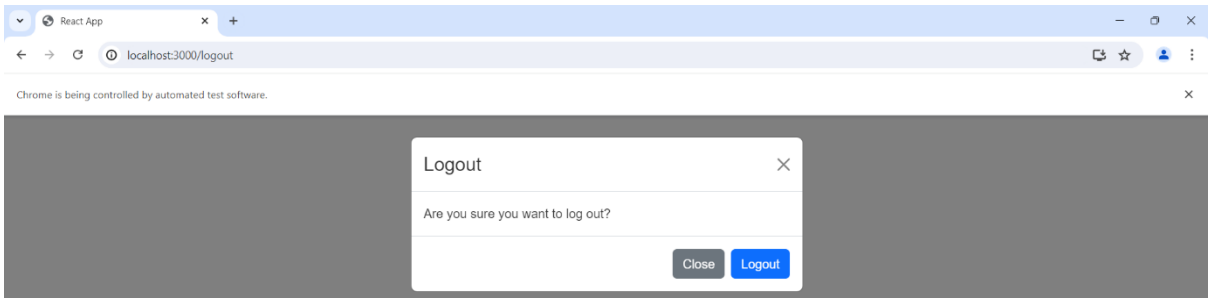
```
def view_my_bookings():
    profile_menu = driver.find_element(By.ID, "dropdown-basic")
    profile_menu.click()
    time.sleep(1)
    my_bookings = driver.find_element(By.XPATH, "//a[text()='My Bookings']")
    my_bookings.click()
    time.sleep(2)
    # Închiderea modalului "My Bookings" apăsând pe butonul 'X'
    close_button = driver.find_element(By.XPATH, "//button[@aria-label='Close']")
    close_button.click()
    time.sleep(2)
```





Fluxul de acțiuni se încheie cu părăsirea platformei. În acest pas, funcția logout accesează și ea meniul „Profile”, selectează opțiunea „Logout” și confirmă deconectarea. În final, se revine în pagina de „Home” a aplicației.

```
def logout():
    profile_menu = driver.find_element(By.ID, "dropdown-basic")
    profile_menu.click()
    time.sleep(1)
    logout_button = driver.find_element(By.XPATH, "//a[text()='Logout']")
    logout_button.click()
    time.sleep(1)
    confirm_logout = driver.find_element(By.XPATH, "//button[text()='Logout']")
    confirm_logout.click()
    time.sleep(2)
```



Testarea interfeței aplicației web cu Selenium mi-a permis validarea funcționalităților principale ale platformei. În acest fel, m-am asigurat că utilizatorii pot efectua cu succes operațiunile de înregistrare, autentificare, rezervare și confirmare a unei sesiuni de fitness. De asemenea, testele au confirmat că utilizatorii pot vizualiza rezervările existente și pot părăsi corect aplicația.

Cu ajutorul acestui script automatizat, am arătat care este numărul de interacțiuni pentru a îndeplini rezervarea unui interval orar. Astfel, un utilizator nou, are nevoie de 10 click-uri pentru a rezerva o ședință de antrenament. De asemenea, mai sunt necesare încă 3 click-uri pentru a vizualiza informațiile despre viitoarea rezervare și pentru a închide meniul. Iar pentru a părăsi platforma, este nevoie de 3 click-uri. Automatizarea acestui proces nu numai că validează funcționalitatea, dar îmbunătățește și experiența utilizatorului prin minimizarea pașilor necesari pentru realizarea unui task.

În concluzie, testarea automată a interfeței a demonstrat că aplicația oferă o experiență intuitivă și eficientă utilizatorilor.

8. Dezvoltări viitoare ale aplicației

În această secțiune, voi discuta despre posibilele dezvoltări viitoare pentru aplicația de gestionare a sălilor de fitness. Pentru acest pas am avut în vedere cerințele și feedback-ul utilizatorilor, precum și tendințele tehnologice actuale.

Unul dintre obiectivele principale ale dezvoltărilor viitoare este extinderea funcționalităților pentru a permite o analiză detaliată a comportamentului utilizatorilor. Acest lucru ar putea implica generarea de rapoarte personalizate pentru managerii de resurse umane, care să ofere informații valoroase despre modul în care angajații folosesc facilitățile de fitness. Aceste rapoarte ar putea include:

- Detalii despre numărul de rezervări, intervalele orare preferate și sălile de fitness cele mai populare.
- Identificarea angajaților care utilizează în mod regulat facilitățile și a celor care nu le utilizează deloc.
- Identificarea tendințelor sezoniere în utilizarea sălilor de fitness.

Pe baza analizei comportamentului utilizatorilor, aplicația ar putea oferi recomandări personalizate pentru angajați. Aceste recomandări ar putea include: programe de antrenament personalizate în funcție de obiectivele de fitness ale angajaților sau sugestii de antrenori personali pe baza preferințelor și istoricului de antrenament al angajaților.

Un alt pas important în dezvoltarea aplicației este integrarea notificărilor prin email. Această funcționalitate va asigura că angajații sunt informați cu privire la rezervările lor, confirmările sau anulările de rezervări. Astfel, utilizatorii ar putea beneficia de:

- Un email automat trimis imediat după efectuarea unei rezervări.
- Email-uri trimise cu o zi înainte de rezervarea programată.
- Notificări trimise în cazul în care o rezervare este anulată sau modificată.

În concluzie, testarea automată a interfeței a demonstrat că aplicația oferă o experiență intuitivă și eficientă utilizatorilor. Pe lângă asigurarea funcționalităților de bază, dezvoltările viitoare vor contribui la îmbunătățirea continuă a aplicației. Oferirea recomandărilor personalizate pentru utilizatori, notificări utile și o infrastructură scalabilă și sigură sunt doar câteva exemple. Implementarea acestor îmbunătățiri vor oferi un plus aplicației și se va asigura o experiență optimă pentru toți utilizatorii.

Concluzii

În această lucrare am abordat dezvoltarea și implementarea unei aplicații web dedicate gestionării rezervărilor pentru sălile de fitness dintr-o companie. Scopul principal al acestei aplicații a fost să îmbunătățească experiența angajaților și să optimizeze utilizarea resurselor companiei. Pe parcursul cercetării, am demonstrat cum un sistem bine proiectat poate eficientiza procesul de rezervare și poate oferi angajaților acces facil și organizat la facilitățile de fitness.

Utilitatea demersului meu constă în oferirea unei soluții tehnologice care simplifică și îmbunătățește gestionarea resurselor interne ale unei companii, având un impact pozitiv asupra productivității și satisfacției angajaților. Această cercetare confirmă importanța programelor de wellness pentru angajați și demonstrează eficiența utilizării tehnologiilor moderne în acest context.

Cu toate acestea, cercetarea prezintă și anumite limite, cum ar fi lipsa unei analize detaliate pe termen lung asupra impactului aplicației asupra sănătății și productivității angajaților. Într-o abordare viitoare, aceste aspecte ar putea fi aprofundate, iar aplicația ar putea fi extinsă cu noi funcționalități și optimizări ale infrastructurii back-end pentru a face față unui număr mai mare de utilizatori. În plus, am evidențiat posibilitatea extinderii funcționalităților aplicației, cum ar fi integrarea notificărilor prin email, generarea de rapoarte pentru manageri și utilizarea algoritmilor de machine learning pentru recomandări personalizate.

În final, doresc să adresez mulțumiri speciale profesorului coordonator, lect. univ. dr. Cristian Bologa, pentru îndrumarea și suportul oferit pe parcursul acestei lucrări. De asemenea, mulțumesc familiei mele pentru inspirația și sprijinul constant în dezvoltarea acestei aplicații.

Bibliografie

Golești̇eanu, G.C. (2018), Tehnica brainstorming-ului. Preluat de pe: https://neuropsihiatru-golesteanu.ro/tehnica-brainstorming-ului/

Marín-Farrona, M., Wipfli, B., Thosar, S. S., Colino, E., Gallardo, L., Felipe, J. L., și López-Fernández, J. (2023), Effectiveness of worksite wellness programs based on physical activity to improve workers' health and productivity: a systematic review. *Systematic Reviews*, 12 (ediția 87). Preluat de pe: https://doi.org/10.1186/s13643-023-02258-6

Takeuchi, H., & Nonaka, I. (1986). The new product development game. *Harvard Business Review*, 64 (ediția 1), p. 137-146. Preluat de pe: https://hbr.org/1986/01/the-new-new-product-development-game

***Archi, Site oficial: https://www.archimatetool.com/

***Bee-Up, Site oficial: https://bee-up.omilab.org/activities/bee-up/download-details/

***ChromeDriver, Site oficial: https://sites.google.com/chromium.org/driver/downloads

***Corporate Wellness Magazine, The benefits of workplace fitness programs and facilities Preluat de pe: https://www.corporatewellnessmagazine.com/article/the-benefits-of-workplace-fitness-programs-and-facilities

***Draw.io, Site oficial: https://app.diagrams.net/

***Express, Site oficial: https://expressjs.com/

***Java Script, Site oficial: https://developer.mozilla.org/en-US/docs/Web/JavaScript

***Luchidchart, Site oficial: https://www.lucidchart.com/

***Microsoft Excel, Site oficial: https://www.microsoft.com/en-us/microsoft-365/excel

***Microsoft Visio, Site: https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software

***Node.js, Site oficial: https://nodejs.org/en

***Postman, Site oficial: https://www.postman.com/

***Python, Site oficial: https://docs.python.org/3.10/

***React, Site oficial: https://react.dev/

***Sanopass, Impactul pozitiv al fitness-ului la locul de muncă asupra sănătății și productivității angajaților, Preluat de pe: https://companii.sanopass.ro/impactul-pozitiv-al-fitnessului-la-locul-de-muncă-asupra-sănătății-și-productivității-angajaților

***Selenium WebDriver, Site oficial: https://www.selenium.dev/documentation/webdriver/

***SendGrind, Site oficial: https://sendgrid.com/en-us

***Task Scheduler, Site oficial: https://learn.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page

***Visual Studio Code, Site oficial: https://code.visualstudio.com/

***Xampp, Site oficial: https://www.apachefriends.org/

Anexe

Anexa 1 – Interviu adresat principalelor grupe de actori din cadrul companiei.

„Întrebare” = „Î”

„Răspuns” = „R”

Interviu cu viitori utilizatori ai aplicației (angajații):

Î: Cât de des obișnuiți să folosiți sălile de fitness pe care compania vi le-a pus la dispoziție?

R: Minimul de ședințe de antrenament pe care le fac pe săptămână este 3, dar uneori ajung și de 4 ori.

Î: Care este intervalul orar în care mergeți de obicei la sală?

R: Depinde de schimbul în care lucrez într-o săptămână, dacă sunt schimbul unu prefer să merg seara, după program, pe la ora 18:00, în timp ce dacă sunt schimbul doi, merg mai devreme, pe la ora 11:00 sau 12:00.

Î: De obicei e aglomerat în sală, ați avut momente când a trebuit să așteptați pentru eliberarea unui aparat?

R: De cele mai multe ori da, în special seara, când majoritatea oamenilor vin la sală.

Î: Cam ce v-ați dori de la această platformă?

R: În principiu să pot rezerva ușor și rapid un interval orar, să pot vizualiza datele rezervării pe care am făcut-o și eventual să o anulez dacă intervine o schimbare în program. De asemenea, un sistem prin care să primesc o notificare legată de rezervarea următoare ar fi benefic.

Interviu cu cei care vor administra platforma

Î: Care credeți ca sunt principalele probleme pe care le întâmpinați?

R: Ne confruntăm cu gestionarea eficientă a rezervărilor și utilizatorilor, din acest motiv am dori acces cât mai ușor asupra bazei de date.

Î: Ce măsuri de securitate credeți că sunt necesare pentru o bună funcționare a platformei?

R: În mod obligatoriu parolele utilizatorilor ar trebui criptate. De asemenea, implementarea funcționalităților pentru recuperarea parolei sau modificarea datelor de profil ar fi o bună practică.

Î: Cât de importantă este implementarea unei interfețe ușor de înțeles pentru toți utilizatorii?
R: Interfața pentru utilizatori trebuie să fie una cât mai intuitivă. Compania este una foarte mare, astfel că avem angajați din toate clasele sociale. Prin urmare trebuie ca toți utilizatorii, chiar și cei care nu sunt foarte bine puși la punct cu tehnologia, să poată folosi platforma.

Î: Ce fel de funcționalități ar trebui să conțină panoul de control al aplicației, care va putea fi accesat doar de către administratori?
R: Panoul de control ar trebui sa fie unul simplu și ușor de gestionat, la fel ca interfața pentru utilizatori. Aici ar trebui să avem acces rapid asupra bazei de date și să putem efectua operații de vizualizare, adăugare, ștergere și editare.

Interviu cu managerii de resurse umane.

Î: Ce fel de impact considerați că au avut programele de fitness asupra productivității angajaților?
R: Impactul a fost unul foarte pozitiv. Din discuțiile pe care le-am avut cu majoritatea angajaților am înțeles faptul că aceștia vin cu mai mare entuziasm la locul de muncă. De asemenea, au scăpat și de stresul și anxietatea pe care venitul la muncă le provoca.

Î: Credeți că întocmirea unor rapoarte pentru a monitoriza activitatea din sălile de fitness ar fi utile?
R: Cu siguranță. Prin aceste rapoarte am avea o viziune mai clară asupra preferințelor angajaților și am putea adapta mai bine aceste programe la nevoile lor.

Î: Aveți anumite sugestii pentru a îmbunătăți interacțiunea angajaților cu aplicația?
R: Crearea unei secțiuni în care utilizatori să poată vedea cine sunt antrenorii personali și cum îi pot ajuta aceștia în cadrul antrenamentelor. Totodată, o zonă cu întrebările apărute frecvent în rândul utilizatorilor și eventual o sesiune de învățare a modului în care se folosește platforma, ar fi necesare.

Î: Din punctul dumneavoastră de vedere, care sunt problemele cu care vă confruntați?
R: Principala problemă se referă la imposibilitatea de a lua anumite decizii informate deoarece nu există un sistem de date centralizat, la care să avem acces rapid.