MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

APLICAȚIE WEB PENTRU SĂLI DE FITNESS

LUCRARE DE LICENŢĂ

Absolvent: **Cătălin Vescan**

Coordonator ştiinţific: **Asist. Prof. Ing. Cosmina Ivan**

2019

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Cătălin Vescan**

APLICAȚIE WEB PENTRU SĂLI DE FITNESS

1. **Enunţul temei:** *Proiectul îşi propune realizarea unui sistem pentru managementul clienților, al abonamentelor şi al rezervărilor pentru o sală de fitness.*

2. **Conţinutul lucrării:** *Cuprins, Introducere, Obiectivele proiectului, Studiu Bibliografic, Analiză şi Fundamentare Teoretică, Proiectare de Detaliu şi Implementare, Testare şi Validare, Manual de Instalare şi Utilizare, Concluzii, Bibliografie, Anexe.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**:

5. **Data emiterii temei:** 1 aprilie 2019

6. **Data predării:** 8 iulie 2019

Absolvent: _____

Coordonator ştiinţific: _____

Cuprins

Capitolul 1. Introducere – Contextul proiectului

1.1. Contextul proiectului

Sportul în viața unei persoane poate însemna foarte multe nu doar un hobby, recreere sau sănătate, ci poate fi chiar un mod de viața, unul in care să te regăsești atunci când simți nevoia de a lua o pauză de la activitățile de zi cu zi, când vrei să schimbi rutina, să iți limpezești gandurile sau pur și simplu vrei să iți acorzi putin timp, și spațiu, doar pentru tine, fără să stai închis în casă.

Având în vedere numărul atât de mare de sporturi care pot fi practicate în ziua de astăzi, foarte mulți oameni aleg să meargă la o sală de fitness din diverse motive fie că vor să meargă alături de un prieten sau o prietenă si el sau ea deja merg la o sală de fitness, fie au practicat un alt sport și vor să facă o schimbare. Indiferent care este alegerea lor, fiecare sală de fitness ar trebui să fie capabilă într-un mod sau altul să facă față unui număr foarte mare de clienți.

Proiectul propus este o aplicație care oferă managementul unei astfel de săli de fitness și al cărei scop vine atât în ajutorul clienților cât și în ajutorul sălii, aplicație care folosește un software rapid și performant, care să facă față concurenței din ziua de astăzi.

Conform studiilor efectuate în domeniu, și a articolului din [1], această aplicație se încadrează în categoria aplicațiilor MIS, aplicații care descriu un sistem informatic folosit în luarea unor decizii, coordonarea controlul, analiza și vizualizarea informațiilor într-o organizație.

Această aplicație are la bază un sistem de management care ajută la gestionarea utilizatorilor și a activităților desfășurate în sala de fitness, oferind funcționalități specifice pentru fiecare tip de utilizator care folosește aplicația, și anume administratorul, angajații și clienții.

1.2. Motivația

Ținând cont de numărul tot mai mare de clienți, și în același timp de diversitatea antrenamentelor, este necesară gestionarea acestora într-un mod cât mai simplu și mai rapid posibil, și totodată păstrarea tutuor informațiilor într-un mediu cât mai compact, evitând astfel utilizarea excesivă a hârtiilor, riscul pierderii acestora, și cel mai important, gestiunea acestora în cazul în care are loc un incident al cărui rezultat duce la deteriorarea sau distrugerea fără posibilitatea recuperării datelor (ex: incendiu).

În urma unei scurte analize a pieței, am constatat că sunt foarte puține săli de fitness care oferă posibilitatea comunicării in timp real cu personalul sălii pentru a afla informații adiționale, posibilitatea rezervării unui loc la antrenamentul dorit, și cel mai important, achiziționarea online a abonamentului dorit, fapt ce m-a determinat sa dezvolt o aplicație care să integreze aceste funcționalități în scopul gestiunii eficiente a datelor utilizatorilor, creșterii productivității și diminuarea drastică, daca nu chiar eliminarea completă a timpului petrecut la bar așteptând ca un angajat sa creeze mai multe abonamente.

1.3. Conținutul lucrării

În acest subcapitol este prezentat conținutul lucrării pe capitole insoțite de o scurtă descriere fiecare.

Capitolul 1 – Introducere

În acest capitol sunt prezentate contextul în care urmează sa ruleze aplicația și totodată o descriere succintă a scopului acestei aplicații, motivele care au contribuit la dezvoltarea acestei aplicații, și o scurtă descriere a conținutului lucrării.

Capitolul 2 – Obiectivele Proiectului

Capitolul 2 prezintă obiectivele principale și secundare propuse pentru implementarea aplicației prezentate.

Capitolul 3 – Studiu Bibliografic

Acest capitol cuprinde studiul concurenței care constă în comparația sistemului prezentat cu alte sisteme similare, concepte folosite în dezvoltarea aplicațiilor web și de asemenea arhitecturile folosite în dezvoltarea sistemului prezentat.

Capitolul 4 – Analiză și Fundamentare Teoretică

În acest capitol sunt prezentate cazurile de utilizare împreună cu actorii sistemului, inclusiv descrierea în detaliu a unor cazuri de utilizare în scopul aprofundării înțelegerii mecanismului aplicației, cerințele funcționale și cerințele non-funcționale ale aplicației, și descrierea succintă a tehnologiilor folosite pentru dezvoltarea aplicației și desigur argumentarea alegerii acestor tehnologii.

Capitolul 5 – Proiectare de Detaliu și Implementare

Capitolul 5 prezintă modul de implementare al aplicației, evidențiat prin arhitecturi și diagrame, este prezentată baza de date, si sunt explicate unele componente pentru înțelegerea funcționării aplicației.

Capitolul 6 – Testare, Validare și Evaluare

Acest capitol prezintă câteva teste care au fost efectuate asupra funcționalității aplicației și metode de validare a datelor.

Capitolul 7 – Manual de Instalare și Utilizare

În acest capitol sunt descrise în detaliu etapele care trebuie urmate pentru instalarea tuturor componentelor aplicației pe propriul dispozitiv, împreună cu resursele software și hardware minime necesare pentru funcționarea optimă a aplicației.

Capitolul 8 – Concluzii

Capitolul 8 conține concluziile rezultate asupra dezvoltării aplicației, precizându-se totodată obiectivele care au fost atinse și descrierea posibilităților de dezvoltare ulterioară.

Capitolul 2. Obiectivele Proiectului

2.1. Obiectivul principal

Această aplicație a fost dezvoltată cu scopul de a furniza o soluție software cât mai eficientă din punct de vedere al gestionării și totodată a stocării informațiilor utilizatorilor aplicației.

Obiectivul principal al acestei aplicații constă in diminuarea, respectiv eliminarea timpul de așteptare petrecut la bar până când angajatul creează abonamentul clientului, respectiv eliminarea tuturor hârtiilor cu informații despre clienți, abonamente, sau programările acestora și stocarea lor într-un mediu compact, și care oferă în același timp garanția recuperării datelor în cazul în care acestea au fost compromise.

Obiectivul este realizat prin implementarea unei componente care permite plata online a abonamentului, o componentă care permite efectuarea rezervărilor online la clasele dorite, un chat disponibil utilizatorilor neînregistrați care doresc să afle mai multe informații, și nu în ultimul rând, baza de date care stochează toate informațiile.

2.2. Obiective secundare

Acest capitol prezintă câteva obiective generale ale aplicației, printre care se numără interfața grafică, securitatea, si managementul.

2.2.1. Interfața grafică

Interfața grafică într-o aplicație software reprezintă unul dintre cele mai importante obiective, deoarece este prima interacțiune a aplicației cu utilizatorul, ceea ce daca nu este bine realizat poate îndepărta utilizatori..

Acest obiectiv este îndeplinit datorită faptului că interfața este „user-friendly”, ceea ce înseamnă defapt că această interfață este ușor de folosit de orice utilizator, nu este deloc incărcată cu detalii, fundalul închis la culoare este ales special pentru evitarea deranjului și a oboselii ochilor, și unul dintre cele mai importante aspecte care țin de interfață este faptul că utilizatorul primește mereu un feedback pentru operațiile pe care acesta le efectuează.

Spre exemplu operația de înregistrare, în urma căreia, dacă credețialele introduse de utilizator au trecut de etapa de validare, vor fi salvate în baza de date, după care acesta primește o fereastră de tip „pop-up” care îl anunță ca înregistrarea a fost efectuată cu succes.

Dacă datele introduse de utilizator în momentul înregistrării nu corespund cerințelor (ex: parolă care să conțină minim un caracter mic, minim un caracter mare, minim o cifră, minim un caracter special și să fie cuprinsă între 8 și 12 caractere), atunci utilizatorului îi va fi afișată o fereastră, și de această dată tot de tip „pop-up”, în care i se precizează faptul că datele introduse nu sunt corecte. La fel se procedează pentru toate operațiile considerate importante.

2.2.2. Securitatea

În ziua de astăzi, orice aplicație trebuie sa dețină un anumit nivel de securitate, pentru a evita orice neplăcere care poate fi cauzată de persoane rău intenționate și care vor doar sa pună bețe în roate altora.

Nivelul de securitate al acestei aplicații nu este unul foarte ridicat, însă el există totuși și este prezentat succint în acest capitol.

Pentru evitarea situațiilor în care un utilizator neautorizat / neînregistrat are acces la resurse sau informații importante care trebuie protejate, am decis ca la nivel de log-in să implementez sistemul de autentificare și autorizare.

Autentificarea constă în compararea și validarea / invalidarea credențialelor introduse de către utilizator / atacator, și conform răspunsului primit de la server se determină dacă persoana respectivă este cine pretinde că este sau nu.

Autorizarea este etapa imediat următoare autentificării, etapă în care utilizatorilor deja autentificați li se verifcă tipul, și în funție de acest câmp sunt redirecționați la paginile corespunzătoare acestora, unde fiecare are permisiuni în funcție de ce tip de utilizator este.

2.2.3. Managementul utilizatorilor

Aceată aplicație este concepută în scopul utilizării de către 4 tipuri de utilizatori: utilizatori neînregistrați, pe care în continuare îi voi numi vizitatori, utilizatori înregistrați, pe care în continuare îi voi numi clienți, angajații sălii, pe care îi voi identifica în continuare sub denumirea de antrenori / antrenori personali (eng. trainers), și nu în ultimul rând administratorul, care în acest caz este unul și același cu proprietarul sălii de fitness (eng. CEO).

Administratorul este singurul utilizator care are acces deplin la toate funcționalitățile aplicației. Acesta poate modifica atât informațiile legate de clienți sau antrenori, cât și informațiile interne precum orarul, abonamentele, ș.a.m.d.

Antrenorii sunt utilizatorii care au permisiuni parțiale, deoarece au dreptul să modifice conținutul orarului în scopul afișării claselor pe care aceștia le organizează, dar nu au dreptul să modifice informațiile clienților, abonamentelor sau prețurile.

Clienții sunt utilizatorii care au cele mai puține permisiuni, aceștia pot doar să vizualizeze informațiile puse la dispoziție de către administrator sau antrenori, îsi pot plăti abonamentul dorit direct pe site și își pot face rezervare la una din clasele dorite.

Vizitatorii sunt utilizatorii care nu au permisiuni deloc, aceștia putând doar sa vizualizeze informațiile puse la dispoziție de către administrator sau antrenori, însă fără un cont aceștia nu vor putea să își achiziționeze abonamentul dorit, și nici să își facă rezervare la vreuna din clasele organizate.

Capitolul 3. Studiu Bibliografic

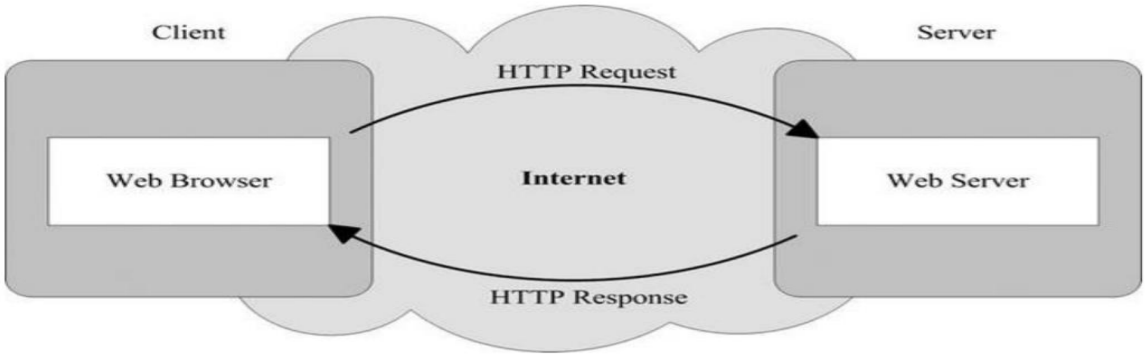
Acest capitol prezintă studiul necesar dezvoltării unei aplicații web, folosind o arhitectură clasică (MVC)[2], și comparația propriei aplicație cu alte aplicații similare, în scopul evidențierii funcționalităților care deosebesc aplicația prezentată de cele deja existente.

3.1. Dezvoltarea aplicațiilor web

Acest capitol prezintă câteva informații conform [3] și [4], care stau la baza dezvoltării aplicațiilor web, și voi începe cu o scurtă definiție, care spune că aplicațiile web sunt sisteme software complexe, în evoluție continuă.

Principiul de funcționare care stă la baza acestui concept este modul de comunicare, și anume relația client – server, în care rolul clientului este jucat de fiecare utilizator care accesează aplicația prin intermediul unui browser, iar rolul serverului este jucat de o mașină aflată la distanță, și care se ocupa de gestionarea cererilor din partea clienților, respectiv gestionarea răspunsurilor potrivite pentru fiecare solicitare în parte.

Imaginea de mai jos descrie foarte simplist modul de funcționare al aplicațiilor web:



[1]

Figura 3.1 Funcționarea aplicațiilor web la modul general

Modul prin care se face legătura dintre operațiile pe care le execută clientul in propriul browser și operațiile care trebuie executate de către server în scopul execuției operației respective și returnând răspunsul corect clientului care a efectuat solicitarea este determinat de URL care face redirecționările corespunzătoare.

Ceea ce se încarcă în browser atât la început cât și pe parcurs este un fișier JSP, și o altă resursă importantă care stă la baza constituirii acestui capitol este prezentată în [5].

[1] http://inf.ucv.ro/documents/mihais/DAW/DAW-1.pdf

3.2. Arhitectura Model View Controller

În prezent, conceptul de MVC este prezent în toate aplicațiile care oferă ca interacțiune cu utilizatorul o interfață grafică.

Motivul pentru care se alege acest model de implementare este datorat faptului că oferă modularitate în dezvoltarea aplicației, deoarece o împarte în 3 mari componente:

1. Model (business logic) – aici are loc partea de business logic, care presupune defapt maparea / transpunerea bazei de date în cod, și logica din spatele aplicației folosită la implementare.
2. View (prezentare) – această componentă este responsabilă pentru câmpurile și informațiile care se afișează fiecărui utilizator, fiind totodată prima interacțiune cu utilizatorul.
3. Controller – această componentă se regăsește între componenta de model și cea de view, și alternativ preia cererile de la componenta de prezentare și transmite datele către componenta de business logic, iar după procesarea datelor returnează un răspuns de la business logic la componenta de prezentare. Această componentă are în principal rolul de a transforma datele primite prin intermediul cererii de la componenta de vizualizare în date valide pentru componenta de business logic, și tranformarea inversă de la componenta de business logic către componenta de prezentare.

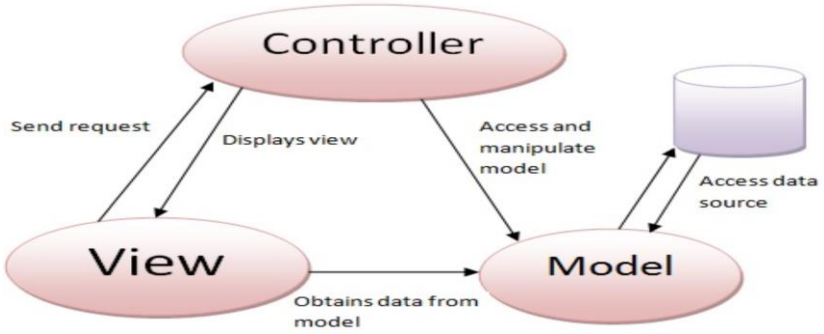


Figura 3.2 Arhitectura Model View Controller (MVC)[2]

3.3. Sisteme similare

În acest capitol sunt prezentate câteva aplicații similare pe care le-am ales în scopul comparării cu aplicația prezentată în actualul document, evidențiind funcționalitățiile cele mai importante.

3.3.1. Health&Style Gym

Aplicația celor de la Health&Style Gym[3] conține câteva secțiuni standard prin intermediul cărora pune la dispoziție informațiile de bază pentru vizitatori, neavând posibilitatea creării unei cont pentru gestiunea clienților, nu deține un sistem de chat prin

[2] Arhitectura MVC - http://www.asparticles.com/2017/01/aspnet-mvc-architecture-overview-model-view-controller.html

[3] Health&Style Gym - https://healthstylegym.wixsite.com/powergym/home

intermediul căruia vizitatorii să poată solicita informații suplimentare, nu prezintă un orar astfel încât vizitatorii să știe care este intervalul orar în care pot merge la sală, sau ce posibilități de alegere a claselor sau a antrenorilor au.

Totodată efectuarea rezervărilor și notificările prin mail nu sunt implementate, și de asemenea nu există nicio metodă de achiziționare a abonamentului online. Această aplicație nu dispune de o versiune mobile sau desktop ci doar web.

3.3.2. GGSport

Aplicația folosită de GGSport[4] structurează într-un mod compact informațiile generale însoțite de poze, dispune de o varietate largă de abonamente disponibile tuturor categoriilor de clienți.

Dintre funcționalitățile urmărite doar orarul și disponibilitatea claselor împreună cu antrenorii care le organizează sunt afișate vizitatorilor, neavând posibilitatea creării unui cont utilizatorii nu își pot face rezervări online, nu își pot achiziționa abonamentul dorit direct din aplicație, lipsind gestionarea clienților aceștia nu vor putea fi notificați în cazul în care sala este închisă.

Această aplicație nu dispune de un chat prin intermediul căruia clienții să poată solicita informații adiționale, și nu are dezvoltată nicio versiune mobile sau desktop ci doar web.

3.3.3. WorldClass

Aplicația utilizată de WorldClass[5] prezentată în acest subcapitol este cea mai complexă din toate punctele de vedere. Sunt prezentate informații legate de orarul sălii și clasele organizate, antrenori și ofertele disponibile utilizatorilor, dispune de managementul utilizatorilor prin posibilitatea creării unui cont, însă unul din marile inconveniente ale acestei aplicații constă în faptul că nu poate fi creat un cont numai dacă utilizatorul alege un abonament, ceea ce impune achiziționarea unui abonament online la înregistrare, lucru care poate să nu fie prea plăcut.

Odată înregistrat ca și client al acestei săli poți efectua rezervări la clasele / antrenamentele de grup dorite, vei primi notificări, poți achiziționa abonamentele dorite online, și cu toate acestea, aplicația nu dispune de un sistem de chat prin intermediul căruia vizitatorii să poată solicita informații suplimentare.

Această aplicație dispune atât de versiunea pentru web cât și de versiunea pentru mobile, însă nu funcționează și pe desktop.

[4] GGSport - http://www.ggsport.ro/index.php
[5] WorldClass - https://www.worldclass.ro/

Numele sălii / Funcționalități	Health&Style Gym	GG Sport	WorldClass	MyGym
Aplicație web	✓	✓	✓	✓
Orarul activităților	×	✓	✓	✓
Locație hartă	×	×	✓	✓
Creeare cont	×	×	✓	✓
Rezervări	×	×	×	✓
Notificări prin mail	×	×	×	✓
Achiziționare abonamente online	×	×	✓	✓
Chat în timp real	×	×	×	✓

Tabel 3.1 Tabel pentru comparația cu alte sisteme similare

Analizând tabelul de mai sus, se poate observa faptul că în timp, apar sisteme ale căror funcționalități sunt tot mai complexe și încearcă să acopere toate nevoile utilizatorilor aplicației.

În urma unei scurte analize putem constata faptul că sistemul implementat primează față de cel mai bun sistem actual prin 3 funcționalități:

Rezervări

Aceasta este una din funcționalitățile care diferențiază acest sistem de restul, deoarece până acum în sistemele similare analizate nu a fost implementată această opțiune.

Notificări prin mail

Trimiterea de notificări prin mail este o altă funcționalitate prin care se diferențiază acest sistem, deoarece am considerat că este important ca utilizatori, respectiv clienții să fie înștiințați din timp de evenimentele care vor urma.

Chat în timp real

Conform tabelului de comparație, nici cel mai bun sistem dintre cele căutate și analizate nu dispune de această funcționalitate, însă sistemul pe care l-am dezvoltat o implementează și este considerată una dintre cele mai importante funcționalități, deoarece întotdeauna vor exista informații care nu au fost precizate, și mai ales întrebări din partea utilizatorilor, iar aceștia trebuie să beneficieze de un „spațiu” expres pentru acest lucru.

Capitolul 4. Analiză şi Fundamentare Teoretică

4.1. Cazuri de utilizare

4.1.1. Actorii sistemului

Utilizatorii care pot accesa această aplicație sunt prezentați pe scurt in paragraful următor, evidențiindu-se totodată și cazurile de utilizare asociate fiecăruia :

Vizitatorii au pe pagina de home câteva detalii despre sala, pot vizualiza atat orarul, antrenorii personali, lista abonamentelor, imagini din interiorul sălii, cât și produsele pe care aceștia le pot achiziționa, pot vizualiza locația exactă pe hartă, pot lua legatura cu cineva în măsură (administratorul sau antrenorii) să le răspundă diferitelor întrebari, și se pot înregistra.

Clienții înregistrați au datele personale legate de cont afișate pe pagina de home, pot vizualiza orarul, antrenorii personali, și produsele pe care aceștia le pot achiziționa, își pot face rezervări la clasele preferate, își pot modifica datele asociate contului sau se pot

Antrenorii personali care își pot vizualiza contul în pagina de home, pot modifica orarul, pot vizualiza lista clienților pe care aceștia îi antrenează sau clienții care s-au înscris la clasa pe care antrenorii o organizeaza, pot cumpăra produse de la sala respectiva, pot răspunde pe chat vizitatorilor care au diferite nelămuriri, își pot modifica datele asociate contului, sau se pot deloga de pe pagina personală.

Administratorul cu acces deplin la toate funcționalitățile sistemului.

În următoarele figuri sunt prezentate diagramele use-case pentru fiecare din utilizatorii descriși mai sus, cu scopul de a evidenția modul de funcționare al aplicației, funcționalitățile oferite de sistem, și modul în care interacționează cu tilizatorii.

În figura următoare este prezentat cazul de utilizare în care actorul principal este un vizitator :

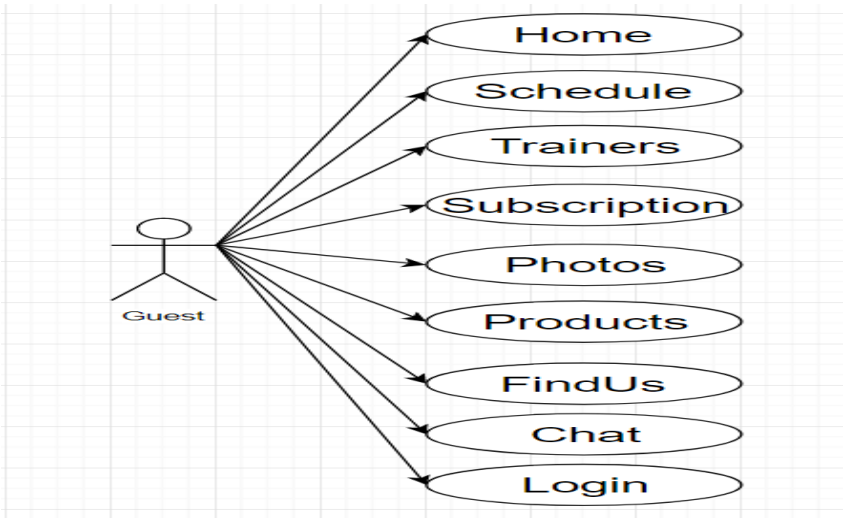


Figura 4.1 Cazuri de utilizare pentru vizitatori

În figura următoare este prezentat cazul de utilizare în care actorul principal este un utilizator înregistrat :

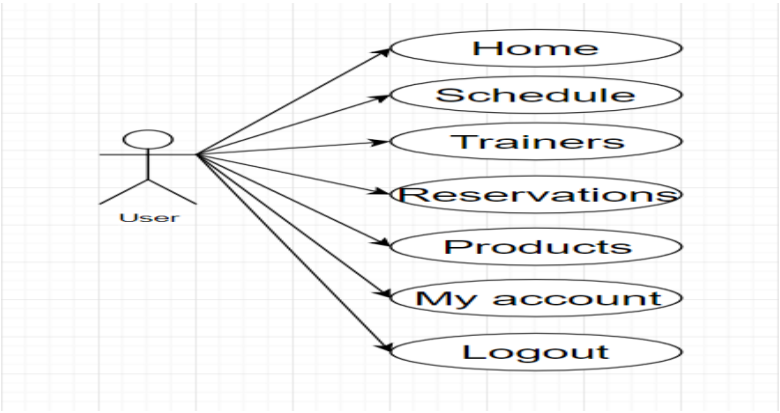


Figura 4.2 Cazuri de utilizare pentru utilizatorii înregistrați

În figura următoare este prezentat cazul de utilizare în care actorul principal este un antrenor personal :

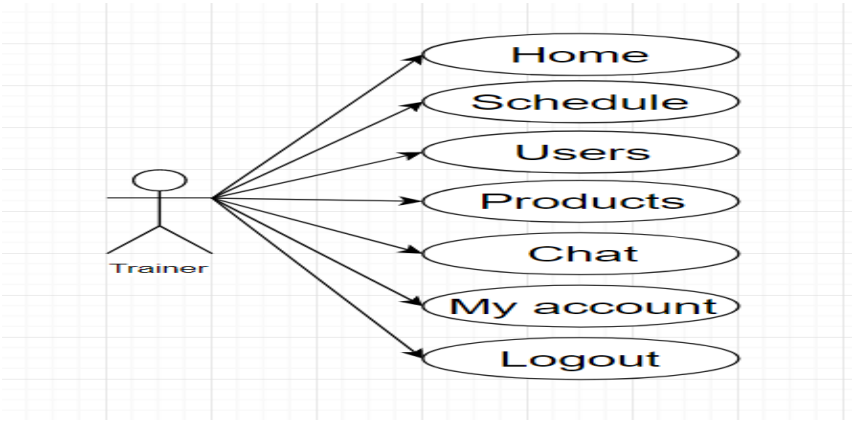


Figura 4.3 Cazuri de utilizare pentru antrenorii personali

În figura următoare este prezentat cazul de utilizare în care actorul principal este un administrator :

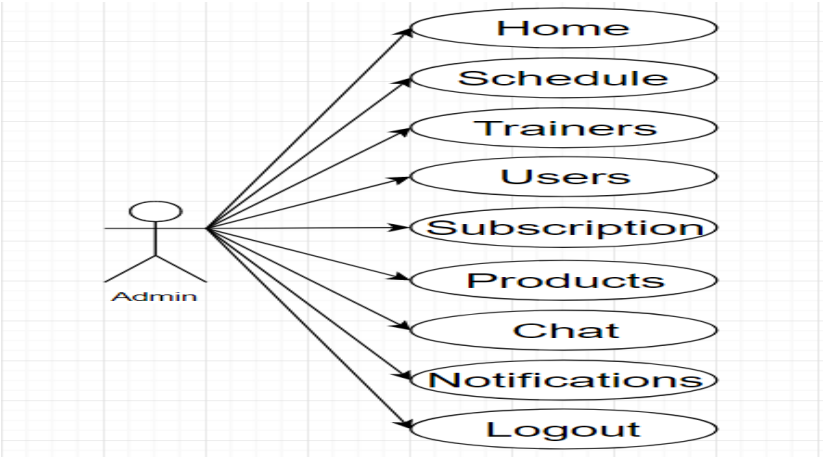


Figura 4.4 Cazuri de utilizare pentru administrator

4.1.2. Descrierea detaliata a cazurilor de utilizare

În această secțiune sunt prezentate diagramele „flowchart” care descriu funcționalitatea componentelor sistemului, acestea fiind reprezentate în funcție de cazurile de utilizare:

Caz de utilizare 1
Numele cazului de utilizare: înregistrarea unui utilizator
Actorul: client / antrenor personal
Părțile interesate:

Utilizatorul care se înregistrează: dorește să folosească aplicația în scopul în care aceasta a fost dezvoltată, bucurându-se de funcționalitățile care îi sunt permise. Inițial înregistrarea se face cu tipul unui utilizator generic pentru ambele categorii de utilizatori, urmând ca după o discuție reală cu administratorul, acesta să modifice tipul specific antrenorului personal, dacă este cazul.

Administratorul: dorește să gestioneze și să aibă evidența tuturor utilizatorilor care intră în sistem.

Precondiții:

Utilizatorul trebuie sa respecte condițiile impuse creării unui cont, iar acestea sunt legate de parolă, care trebuie să conțină cel puțin o literă mică, cel puțin o literă mare, cel puțin o cifră, cel puțin un caracter special, și să conțină între 8 și 12 caractere, și în același timp, să nu mai existe un alt cont cu același nume de utilizator.

Postcondiții:

După înregistrare, utilizatorul trebuie să introducă aceleași credențiale pentru autentificare ca cele introduse în momentul înregistrării, și să utilizeze sistemul și funcționalitățile în scopul în care acestea au fost realizate, iar în cazul depistării unor erori în timpul funcționării aplicației, să le reclame administratorului și să nu profite de ele în interese proprii.

Scenariul de succes:

1. Clientul accesează pagina de autentificare / înregistrare.
2. Introduce credențialele conform condițiilor specificate și în secțiunea de precondiții.
3. Credențialele sunt validate și introduse în baza de date.
4. Clientul primește un pop-up de confirmare cum că înregistrarea a fost efectuată cu succes.

Scenariul de eșec are loc atunci când se întâmplă una din următoarele evenimente:

1. Utilizatorul introduce un nume de utilizator care este deja înregistrat în baza de date.
2. Utilizatorul introduce o parolă care nu corespunde cerințelor specificate și în precondiții.
3. Spațiul de stocare a ajuns la capacitate maximă și nu se mai pot efectua înregistrări.
4. Indisponibilitatea serverului.

Diagrama următoare arată în detaliu fluxul acestei funcționalități:

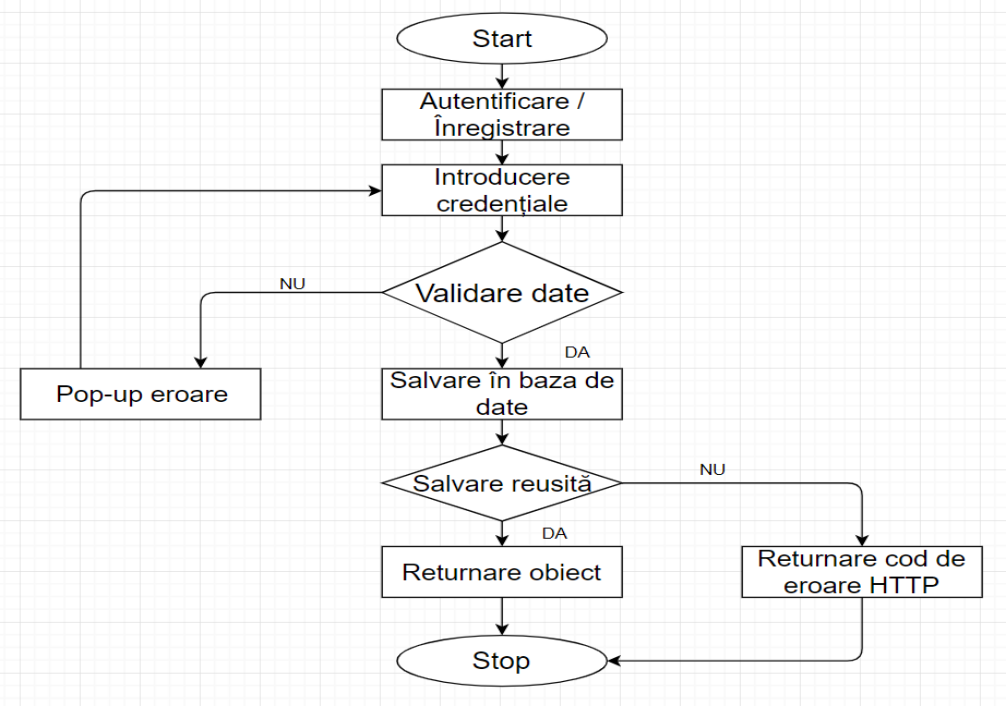


Figura 4.5 Diagrama flow-chart pentru înregistrarea unui utilizator

Următoarea etapă constă în prezentarea unei diagrame de secvență care are rolul de a evidenția etapele fluxului acestei funcționalități:

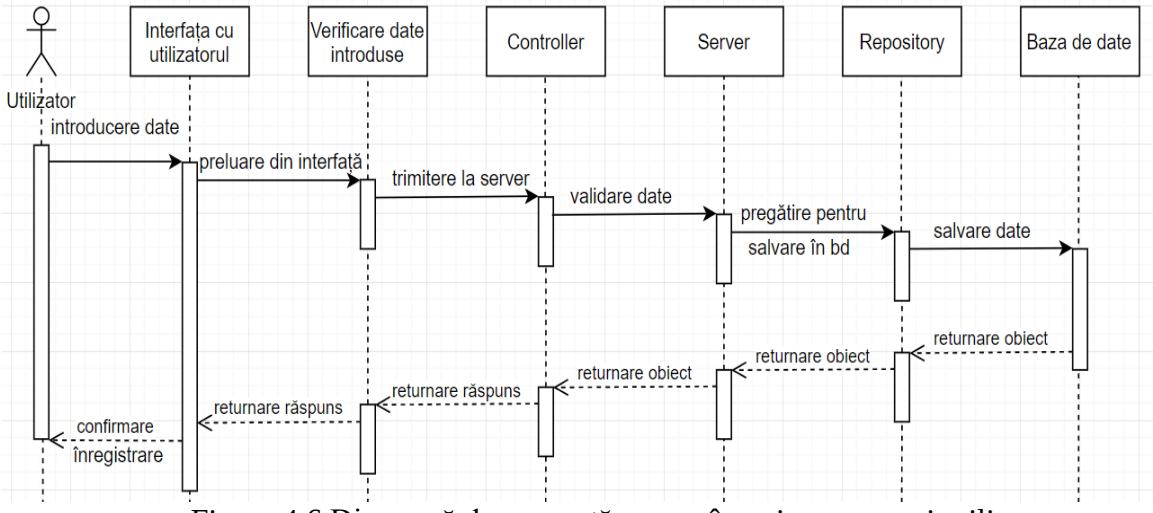


Figura 4.6 Diagramă de secvență pentru înregistrarea unui utilizator

Caz de utilizare 2
Numele cazului de utilizare: Achiziționarea unui abonament online
Actorul: Clientul înregistrat
Părțile interesate:

Utilizatorul înregistrat este principala parte interesată pentru această funcționalitate, deoarece aceasta este implementată special pentru client cu scopul de a evita să mai fie supus situațiilor în care trebuie să aștepe până cand persoana care îi creează abonamentul îi introduce datele în sistem, și mai ales dacă se întâmplă să mai fie și alte persoane care așteaptă la rând.

Administratorul este și el parte interesată, deoarece dorește să aibă o evidență constantă asupra clienților și a abonamentelor acestora. Analizând informațiile stocate, acesta poate avea control asupra persoanelor care încearcă să acceseze sala fără ca abonamentul să fie valid, sau chiar în scopul efectuării unor statistici care să contribuie la modificarea abonamentelor astfel încât acestea să fie cat mai mult în avantajul ambelor părți.

Precondiții:

Condițiile pe care clientul trebuie să le îndeplinească înaine de a putea efectua achiziționarea abonamentului sunt următoarele:

1. Clientul trebuie să parcurgă manualul de utilizare pentru a vedea cum se folosește funcționalitatea respectivă.
2. Clientul trebuie să fie autentificat în propriul cont.
3. Clientul trebuie să se informeze în legătură cu abonamentele și detaliile acestora.
4. Clientul trebuie să dețină un card valid, și de asemenea o sumă care să acopere valoarea abonamentului.
5. Orice încercare de fraudare a sistemului de plată poate fi sancționată.

Postcondiții:

În urma achiziției abonamentului, clientului îi este recomandat:

1. Să verifice suma retrasă de pe card.
2. Să contacteze administratorul în cazul în care a sesizat o eroare.

Scenariul de succes:

1. Clientul accesează pagina de autentificare și se loghează în aplicație în contul său.
2. Clientul accesează secțiunea pentru achiziționarea abonamentului.
3. Clientul introduce date valide în câmpurile solicitate.
4. Clientul așteaptă procesarea și confirmarea plății.

Scenariu de eșec are loc atunci când se întâmplă unul din următoarele evnimente:

1. Clientul nu se poate autentifica în cont din diverse motive.
2. Clientul a introdus date invalide.
3. Clientul nu are suficient sold.
4. Insiponibilitate sau probleme pe server.

Fluxul acestei funcționalități este prezentat în următoarea diagramă:

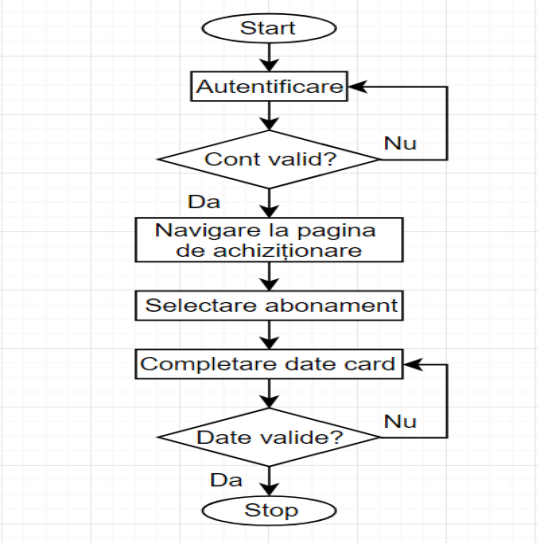


Figura 4.7 Diagrama flow-chart pentru achiziționarea unui abonament

Următoarea diagramă de secvențe ne va ajuta să înțelegem mai bine ce se întâmplă defapt în timp ce rulează această funcționalitate, și cum se „mișcă” datele între componente:

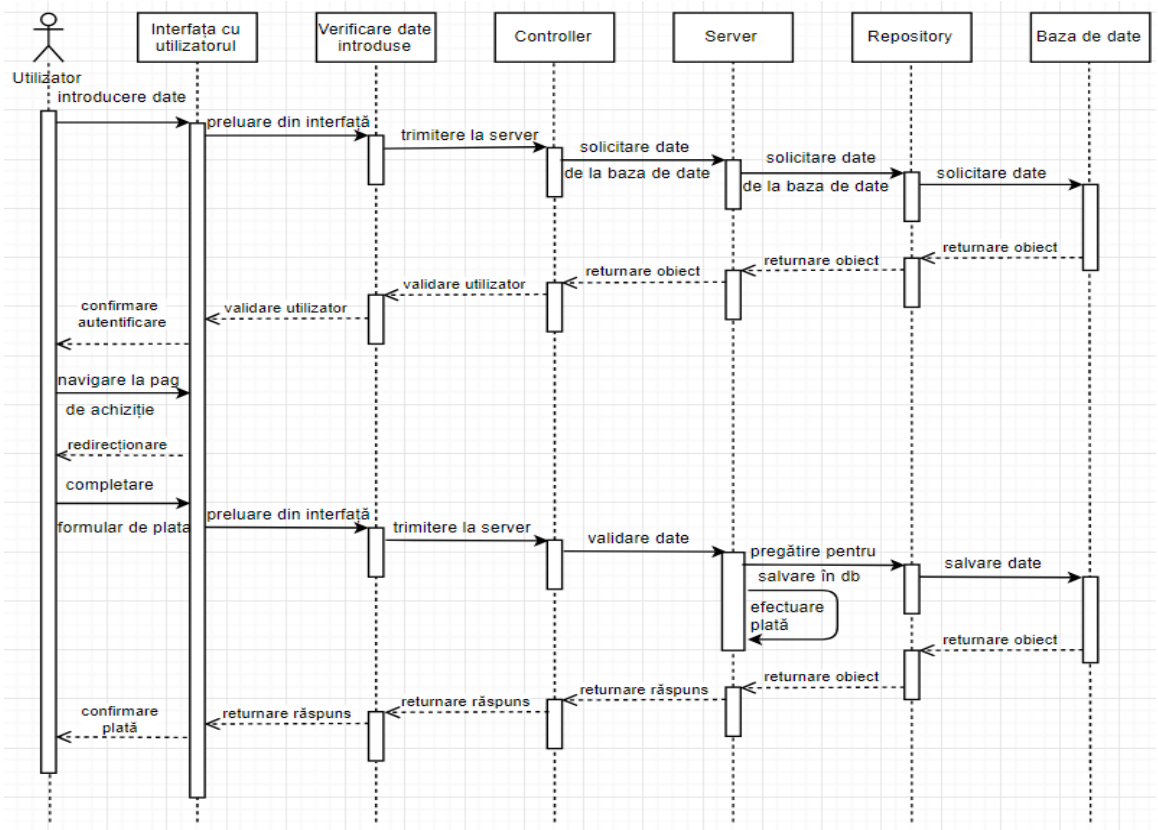


Figura 4.8 Diagramă de secvență pentru achiziționarea unui abonament

4.2. Tehnologii utilizate în dezvoltarea aplicației web

4.2.1. Java EE

Java EE[6], după cum se specifică și în [6], este o colecție de tehnologii și API-uri pentru platforma Java, care vine în sprijinul dezvoltării și implementării aplicațiilor enterprise (pentru întreprinderi), care pot fi în general clasificate ca fiind aplicații distribuite și la scară largă.

Furnizorul de produse Java EE este de obicei un furnizor de aplicații server, web-server, folosind servlet-uri, sau de baze de date, folosind JPA, care oferă clase ce implementează interfețele care au fost definite în interfață.

Am ales să folosesc această tehnologie, deoarece are la bază limbajul de programare Java, și cu acesta dintre toate am cea mai vastă experiență, este o tehnologie utilziată la scară largă, și din acest motiv este și mult mai ușor de găsit soluții și rezolvări în cazul în care apar probleme.

4.2.2. Spring boot framework

Spring Boot, după cum se specifică și în [7], folosește un model de dezvoltare complet nou pentru a ușura dezvoltarea programelor folosind Java, evitând niște pași de dezvoltare și configurație.

Principalul obiectiv al acestui framework este de a reduce timpul de dezvoltare, de testare precum și de a ușura dezvoltarea aplicațiilor web spre deosebire de Spring framework care necesită mai mult timp.

Am decis să folosesc acest framework în dezvoltarea aplicației propuse datorită faptului că evită complet configurările XML, evită definirea mai multor configurații de adnotări, combinarea diferitelor adnotări existente in Spring cu o adnotare simplă și unică, evitarea scrierii unui numar mare de importuri și furnizează unele setări implicite pentru a crea proiecte noi in cel mai scurt timp.

4.2.3. Maven

Așa cum este prezentat și în [7], Apache Maven este un instrument pentru managementul proiectelor și pentru înțelegere, bazat pe fișierul pom.xml, care vine de la POM.

Maven poate gestiona construirea, raportarea și documentarea unui proiect dintr-o informație centrală, abordează două aspecte legate de construirea software-ului, și anume descrie modul în care software-ul este construit și dependințele acestuia.

Am ales Maven pentru dezvoltarea sistemul propus datorită faptului că acesta urmează un set de standarde, utilizat peste tot în domeniu, acolo unde se folosește acest instrument.

Oferă o configurație simplă a proiectului, care utilizează cele mai bune practici, proiectele urmând o structură consistentă. Având o structură consistentă, modificările ulterioare sunt mult mai ușor de făcut.

[6] Toby Hede - https://stackoverflow.com/questions/106820/what-is-java-ee

Maven oferă o modalitate convenabilă de a declara aceste dependințe de proiect, într-un fișier separat numit pom.xml, care descarcă și instalează automat aceste dependințe și permite utilizarea lor în proiect.

4.2.4. Java Persistence

Conform informațiilor din [7] și [8], JPA este doar o specificație care descrie interfețele cu care clientul operează și metadatele standard de mapare.

Dincolo de definiția API, JPA explică modul în care aceste specificații ar trebui să fie implementate Hibernate era deja o implementare Java ORM până în momentul în care specificația JPA a fost lansată pentru prima dată.

JPA se ocupă de modul în care obiectele Java sunt stocate în baza de date relațională, cum pot fi accesate și modul în care starea obiectului poate fi stocată astfel încât să poată fi restaurată la reluarea aplicației.

Figura de mai jos arată unde se află mai exact interfața JPA în cadrul unei aplicații:

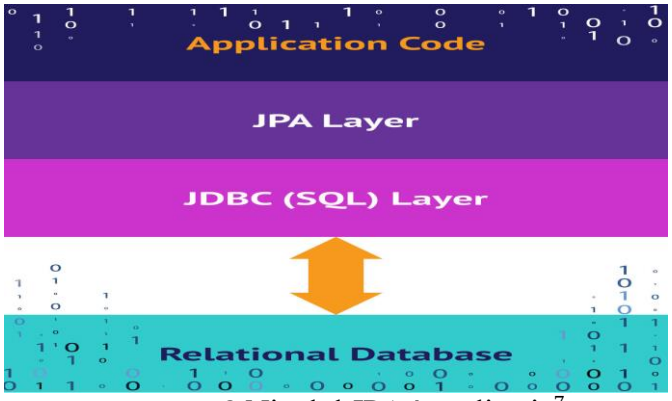


Figura 4.9 Nivelul JPA în aplicație[7]

4.2.5. RESTFul Web Services

După cum scrie și în [7], un serviciu web (WS) este o aplicaţie web de tip client-server, în care un server furnizor de servicii este accesibil unor aplicaţii client pe baza adresei URL a serviciului.

Principalul merit al serviciilor web este acela că asigură interoperabilitatea unor aplicaţii software implementare pe platforme diferite şi cu instrumente diferite. Diferenţa dintre o aplicaţie web clasică şi un serviciu web constă în principal în formatul documentelor primite de client şi modul în care acestea sunt folosite: într-o aplicaţie web, clientul primeşte documente HTML transformate de un browser în pagini afişate, iar clientul unui serviciu web primeşte un document XML / JSON folosit de aplicaţia client, dar care nu se afişeaza direct pe ecran.

[7] JPA - https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

Servicii REST (Representational State Transfer)

REST, conform informațiilor din [9], reprezintă un model arhitectural pentru crearea serviciilor web. Acesta descrie o arhitectură orientată pe resurse.

Motivul realizării unei astfel de arhitecturi a pornit de la faptul că serviciile de tip RPC şi cele care folosesc SOAP aduc o oarecare complexitate. Astfel, simplitatea web a reprezentat sursa de inspiraţie a serviciilor web de tip REST.

Deşi nu este un standard, el se foloseşte de următoarele standarde:

- HTTP
- URI
- XML / HTML / JPEG

REST este un set de reguli la care o arhitectură ar trebui să se conformeze, astfel putem spune că este un stil arhitectural.

HTTP furnizează patru metode de bază ce descriu operaţiile cele mai comune, și totodată suficiente pentru a realiza o arhitectură de tip REST:

- Crearea unei noi resurse: HTTP POST
- Extragerea unei resurse: HTTP GET
- Actualizarea unei resurse existente: HTTP PUT
- Ştergerea unei resurse: HTTP DELETE

Cu toate că metodele GET și POST sunt cele mai folosite, trebuie ținut cont de câteva detalii, precum diferențele dintre acestea, după cum se specifică și în [10], și anume faptul că prin intermediul metodei GET, informația transmisă de utilizator este încărcată în url, și este metoda care se folosește atunci când se solicită date de la server, respectiv baza de date.

În tot acest timp, metoda de tip POST este folosită atunci când vrem să transmitem informații spre server, respectiv baza de date, însă aceste informații nu sunt încărcate în url, ci se folosește JSON.

Totodată trebuie precizat și faptul că între aceste 2 metode există și o diferență de siguranță asupra informațiilor, conform [11], metoda GET oferă siguranță asupra informațiilor, în timp ce metoda POST nu oferă acest beneficiu.

Avantajele folosirii serviciilor REST sunt următoarele:

- Este mai simplu şi are o documentaţie mai uşor de înţeles decât alte servicii
- Permite mai multe formate de date
- În general, foloseşte JSON, acesta fiind procesat mult mai rapid, lucru care duce la o navigare mai rapidă pentru client
- Are o performanţă şi o scalabilitate mai bună
- Serviciile REST sunt folosite de către mari companii, cum ar fi: Yahoo, Flickr, Amazon, Ebay, etc.

Imaginea următoare arată modul în care informațiile circulă folosind serviciile REST:

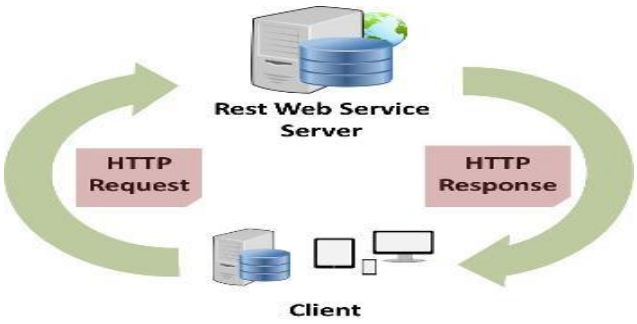


Figura 4.10 Servicii REST[8]

Am decis că pentru sistemul implementat cea mai potrivită arhitectură este arhitectura REST datorită numărului mare de avantaje ale acesteia.

4.2.6. Baza de date

O bază de date, așa cum specifică și în [7], reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

De obicei o bază de date este memorată într-unul sau mai multe fișiere. Bazele de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date. Cel mai răspândit tip de baze de date este cel relațional, în care datele sunt memorate în tabele.

Pe lânga tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, mecanisme de securitate și de gestiune a tranzacțiilor.

MySQL, după cum scrie și în [12], este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeza MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală, fiind o componentă cheie a stivei LAMP.

Deși este folosit foarte des împreună cu limbajul de programare PHP, cu MySQL se pot construi aplicații în orice limbaj major. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum ar fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc.

Am decis să folosesc MySQL, deoarece conform celor de la Oracle [13], MySQL este cea mai populară bază de date open source din lume. Prin performanţa dovedită, fiabilitate şi uşurinţă de utilizare, MySQL a devenit cea mai importantă bază de date pentru aplicaţiile bazate pe web, utilizată de companii de renume, cum ar fi Facebook, Twitter, YouTube şi toate cele cinci site-uri web de top. De asemenea, este o alegere foarte populară ca bază de date integrată, distribuită de mii de furnizori şi producători de software.

Așa cum prea bine știm, pentru o bună funcționare a aplicațiilor care folosesc baze de date, acestea trebuie să fie bine implementate, prin urmare în proiectarea bazei de date s-a folosit normalizarea[14]. Normalizarea este o tehnică de generare a unor relații atent concepute astfel încât informațiile stocat în baza de date să fie memorate corect, iar nivelele sunt FN1, FN2, FN3, FN Boyce-Codd, FN4, FN5.

[8] REST - https://www.stsoftware.com.au/site/ST/blog/article/rest-define/

4.3. Cerințele sistemului

Precum este specificat și în [15], cerințele sistemului reprezintă descrierile la nivel înalt referitoare la anumite servicii de sistem, constrângeri sau la o specificație detaliată care este generată în timpul procesului de colectare a cerințelor.

Acestea sunt împărțite in doua categorii:

- Cerințe funcționale: descriu serviciile sistemului, modul în care sistemul ar trebui să raspundă la anumite date de intrare și modul in care sistemul ar trebui să se comporte în situații concrete.
- Cerințe non-funcționale: descriu atributele sistemului, specifică criteriile care pot fi folosite pentru a analiza funcționare unui sistem în anumite condiții.

4.3.1. Cerințe funcționale

Identificator	Descrierea cerinței funcționale	Beneficiar
CF 1	Vizualizare orar	Toți utilizatorii
CF 1.1	Editare orar	Administratorul, Antrenorii
CF 2	Vizualizare abonamente	Toți utilizatorii
CF 2.1	Editare abonamente	Administratorul
CF 3	Vizualizare locație pe hartă	Utilizatorii neînregistrați
CF 4	Comunicare în timp real cu personalul sălii (Chat)	Utilizatorii neînregistrați, Administratorul, Antrenorii
CF 5	Autentificare	Utilizatorii înregistrați
CF 6	Înregistrare	Utilizatorii neînregistrați
CF 7	Trimitere de notificări	Administratorul
CF 8	Modificarea datelor contului	Utilizatorii înregistrați, Antrenorii
CF 9	Rezervări	Utilizatorii înregistrați
CF 10	Achiziționarea abonamentului dorit	Utilizatorii înregistrați

Tabel 4.1 Cerințele funcționale ale aplicației

4.3.2. Cerințe non-funcționale

Cerințele non-funcționale descriu modul în care trebuie să funcționeze sistemul, definesc proprietăți și constrângeri ale sistemului și pot fi de diferite tipuri precum cerințe de portabilitate, cerințe de fiabilitate, cerințe legate de viteză, securitate, etc.

Utilizabilitatea – măsura în care un produs poate fi utilizat de utilizatori specifici pentru a atinge scopuri specifice cu eficacitate, eficienţă şi satisfacţie într-un context de utilizare specificat. [16]

Utilizabilitatea se referă la ușurința modului de utilizare a aplicației de către utilizatori indiferent de nivelul cunoștințelor tehnice.

Unul din motivele care evidențiază utilizabilitatea aplicației este faptul că interfața este simplă și utilizatorii găsesc foarte repede informațiile căutate, iar timpul de familiarizare cu aplicația este unul foarte scurt.

Securitatea – respectă un standard care are la baza urmatoarele principii care definesc securitatea informatiei: confidentialitatea, integritatea si disponibilitatea informatiei, conform [17] și [18].

Confidentialitatea: asigurarea faptului că informația este accesibilă doar persoanelor autorizate.

Integritatea: păstrarea acurateței și completitudinii informației precum și a metodelor de procesare;

Disponibilitatea: asigurarea faptului că utilizatorii autorizați au acces la informație și la resursele asociate atunci când este necesar.

Utilizatorilor le este garantată protecția atât a datelor personale stocate în baza de date, cât și la datele bancare, prin intermediul framework-ului folosit, în cazul în care aceștia doresc să achiziționeze abonamente sau produse online.

Scalabilitatea – un sistem ce rămâne neschimbat în prezența unei creșteri semnificative a numărului de resurse, componente și utilizatori, după cum scrie în [19].

Sistemul se bazează pe faptul că modul de implementare și dezvoltare este în așa fel conceput încât ulterior să poată fi modificat, respectiv extins cu ușurință atât din punct de vedere al resurselor disponibile cât și din punct de vedere al utilizatorilor care folosesc acest sistem.

Aplicația este construită modular atât din punct de vedere al fișierelor / claselor, cât și din punct de verdere al metodelor / funcțiilor, ceea ce face codul mult mai ușor de înteles, corectat sau modificat în cazul extinderii sau adăugării unor noi funcționalități.

Performanța – măsurarea calității serviciului furnizat de un sistem informatic conform [20].

Cu alte cuvinte, performanța reprezintă eficiența aplicației din punct de vedere al numărului de utilizatori pe care îi poate gestiona în același timp, și al timpului de răspuns.

Suportă un minim de 10 000 de utilizatori înregistrați, însă numai aproximtiv 200 de cereri simultane / concurente fără pierderi de performanță, și răspunde cererilor efectuate de utilizatori în mai puțin de 3 secunde. Aplicația folosește conexiune la o bază de date suficient de puternică pentru a suporta un numar foarte mare de clienți.

Din punct de vedere hardware, este indicat ca utilizatorii să dispună de un procesor de minim 1.6 GHz, o placă video de minim 128 Mb, și o viteză cât mai mare la internet, pentru o funcționare optimă.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol sunt prezentate în detaliu proiectarea și implementarea, sau cu alte cuvinte sunt descrise diagramele utilizate în dezvoltarea aplicației web și secvențe de cod importante.

5.1. Arhitectura sistemului

5.1.1. Arhitectura aplicației web

Pentru acest proiect am ales sa folosesc o arhitectură pe trei nivele, specifică unei aplicați client – server, această arhitectură, descrisă în detaliu în, încapsulează partea de interacțiune cu utilizatorul, logica funcționării și accesul la baza de date în module independente.

Acest tip arhitectural, fiind bine modularizat, permite modificarea sau chiar înlocuirea ușoara a întregului modul, fără să afecteze în vreun fel modul de funcționare a celorlalte module.

SGBD (sistemul de gestiune al bazei de date) conține logica de acces și stocare a datelor. Imaginea următoare descrie foarte general această arhitectură, urmând ca fiecare nivel din această arhitectură să fie descris in detaliu.

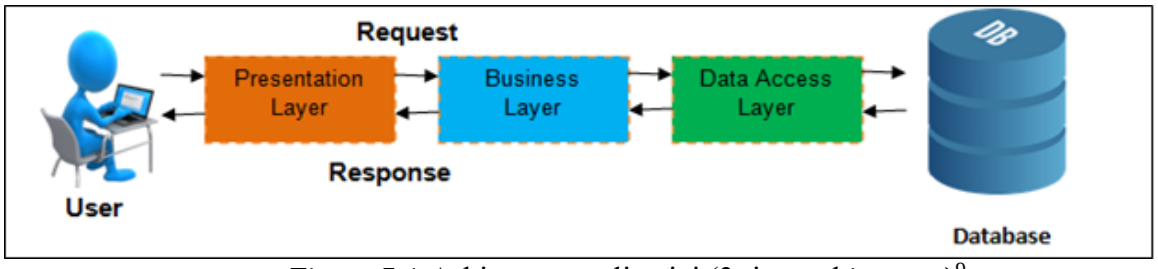


Figura 5.1 Arhitectura aplicației (3 tier architecture)[9]

5.1.1.1. Nivelul de prezentare (Presentation Layer)

Acest nivel este direct responsabil pentru preluarea datelor de intrare și a cererilor efectuate de clienți și totodată afișarea răspunsurilor primite de la server, respectiv atenționările în cazul în care datele introduse nu sunt valide, într-un format cât mai prietenos și plăcut pentru utilizator.

Deși acest nivel este specific pentru prezentare, după cum îi este și numele, nimic nu mă împiedică să am și o mică parte de logică tot la acest nivel, din contră acest lucru este chiar indicat, deoarece sunt anumite validări ale datelor de intrare ale utilizatorilor care pot fi efectuate foarte repede în acest nivel și nu doar că nu are rost, dar este și mai ineficient să se trimită datele la nivelul de logică pentru a fi validate.

Spre exemplu, în cazul în care utilizatorul introduce o parolă care nu îndeplinește criteriile solicitate, validarea făcută la nivelul de prezentare nu mai implică serverul, care

[9] 3 tier architecture - https://www.techopedia.com/2/32100/software/a-detailed-look-at-3-tier-software-architecture

în această situație ar fi nevoit să trimită un răspuns conform căruia utilizatorul să fie înștiințat că datele introduse nu sunt corecte.

Imaginea de mai jos reprezintă modul în care este partiționat acest nivel de prezentare:

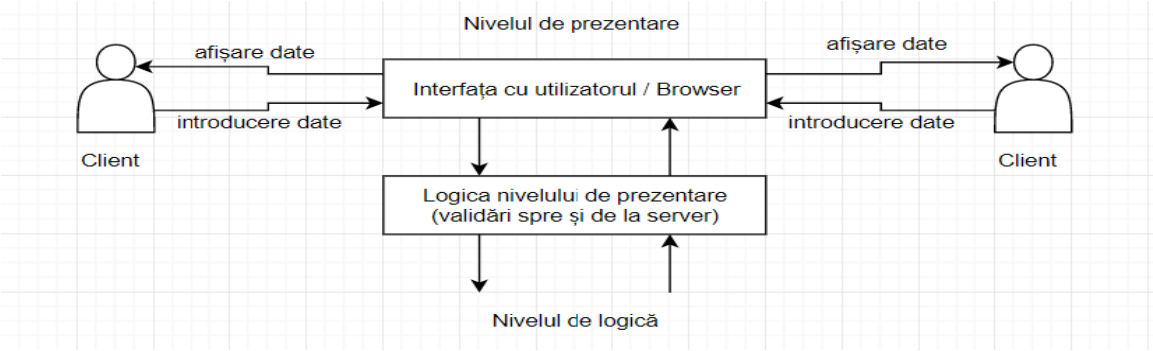


Figura 5.2 Nivelul de prezentare. Primul nivel din arhitectura folosită

Am ales să implementez în acest fel nivelul de prezentare ca să ofer atât o viteză de răspuns mai mare cât și o transformare validă a datelor din ceea ce introduce utilizatorul în ceea ce trebuie să primească serverul, și desigur transformarea inversă, din ceea ce răspunde serverul în afișarea prietenoasă care trebuie afișata clienților.

5.1.1.2. Nivelul de logică (Business Layer)

Acest nivel definește soluții complexe pentru rezolvarea problemelor de business, și în același timp conține module care fac legătura între frontend și backend, prin utilizarea adnotărilor care conțin url-urile specifice pe care clienții le accesează din browser, și totodată fac legătura dintre server și baza de date.

În acest nivel intermediar are loc atât tranziția datelor de la clienți la server și de la server la clienți, cât și funcționalitatea de trimitere de mail-uri / notificări, și diferite validări ale datelor.

În imaginea de mai jos se poate observa modul în care sunt împărțite modulele acestui nivel, și care sunt legăturile dintre ele:

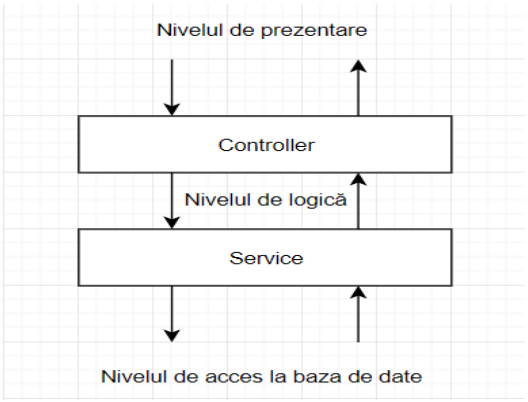


Figura 5.3 Nivelul de logică. Al 2-lea nivel din arhitectura folosită

Controller

Aceasta este componenta care mapează fiecare url pe care clientul îl accesează din browser la metoda responsabilă pentru transmiterea datelor către server, și de asemenea răspunsul primit de la componenta de service pe care îl trimite înapoi spre client. Metodele implementate în această componentă returnează un cod HTTP specific operației efectuate (OK – reprezintă faptul că cererea a fost efectuată cu succes, CREATED), sau un cod HTTP de eroare în caz că nu s-au îndeplinit toate criteriile (BAD_REQUEST, UNAUTHORIZED, NOT_FOUND).

Service

Această componentă realizează în primul rând tranzitul datelor de la componenta de control (Controller) la componenta care realizează accesul la baza de date și trimiterea răspunsurilor primite de la această componentă înapoi la Controller.

Tot în această componentă are loc și trimiterea de mail-uri și notificări către utilizatori pe baza adresei de mail specificată de aceștia în propriul cont.

5.1.1.3. Nivelul de acces la baza de date (Data Access Layer)

În acest nivel este implementată o componentă al cărei rol principal constă în accesarea bazei de date și totodată tranziția datelor spre și de la aceasta.

Această componentă primește de la componenta Service de la nivelul de logică un obiect cu datele care trebuie prelucrate, iar în funcție de ce operație trebuie să execute asupra bazei de date va returna spre Service un obiect care conține datele modificate din baza de date.

Datorită JPA (Java Persistence API), o interfață simplă și ușor de utilizat, toate operațiile de bază, precum adăugarea, citirea, actualizarea sau ștergerea, care trebuie efectuate asupra unei baze de date sunt deja implementate și tot ce trebuie făcut este să fie apelate metodele specifice.

În următoarea imagine se poate observa componenta de acces la baza de date:

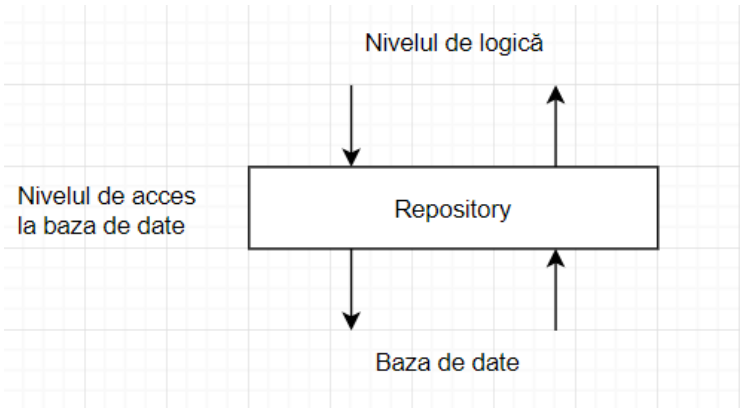


Figura 5.4 Nivelul de acces la baza de date. Al 3-lea nivel

5.1.1.4. Baza de date (Database)

Baza de date este o componentă independentă, folosită pentru stocarea și manipularea datelor. La nivelul de acces al bazei de date se execută o metodă care reprezintă defapt operația care trebuie efectuată la nivelul bazei de date, desigur transmițând obiectul / datele de prelucrat ca parametru al metodei.

În imaginea de mai jos se poate observa fluxul datelor spre și de la baza de date:

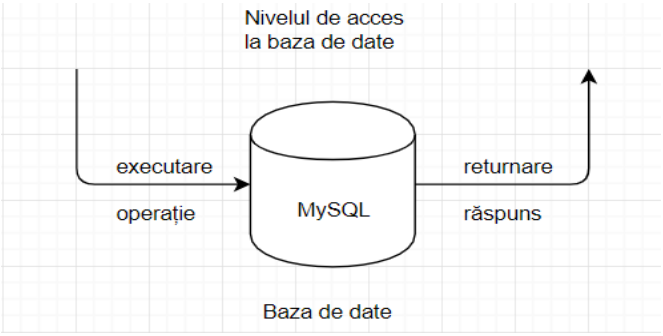


Figura 5.5 Nivelul bazei de date

5.2. Diagramele sistemului

5.2.1. Diagrame de navigare

Acest capitol prezintă operațiile posibile pentru fiecare tip de utilizator în parte, sub formă de diagramă. O diagramă de navigare prezintă toate paginile / componentele pe care un utilizator le poate accesa.

Diagrama de navigare corespunzătoare oricărui utilizator neînregistrat este următoarea:

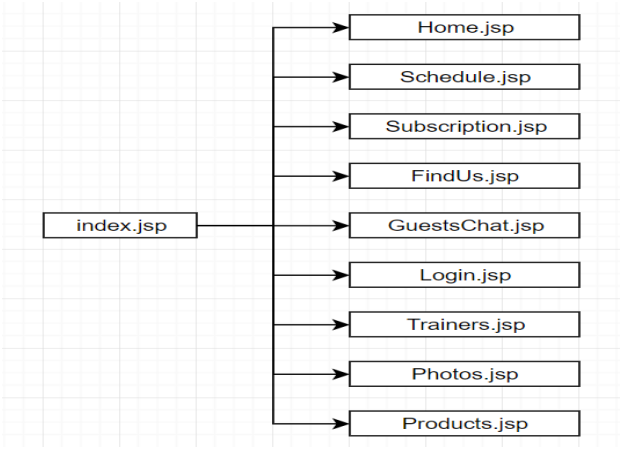


Figura 5.6 Diagrama de navigare pentru utilizatorii neînregistrați

Diagrama de navigare corespunzătoare administratorului:

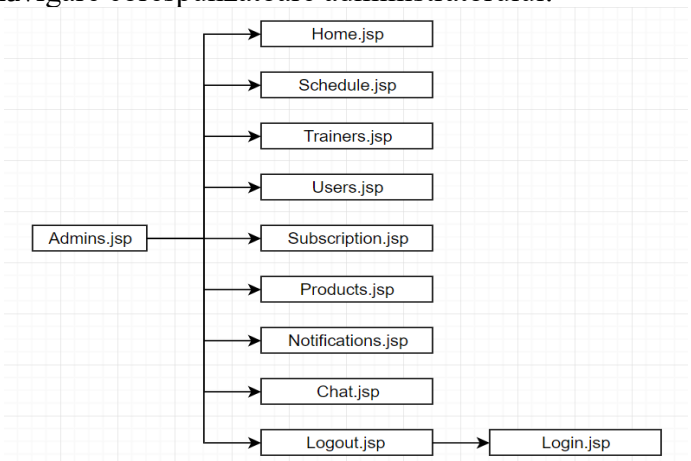


Figura 5.7 Diagrama de navigare pentru administrator

Diagrama de navigare corespunzătoare antrenorilor:

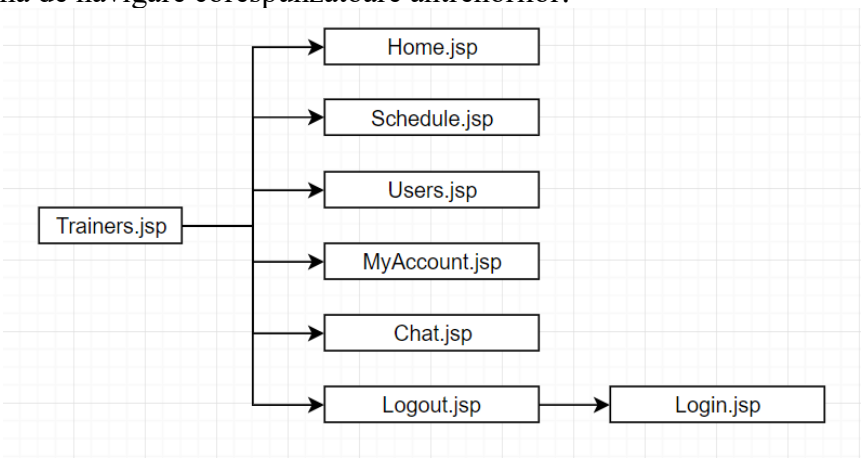


Figura 5.8 Diagrama de navigare pentru antrenorii personali

Diagrama de navigare corespunzătoare utilizatorilor înregistrați:

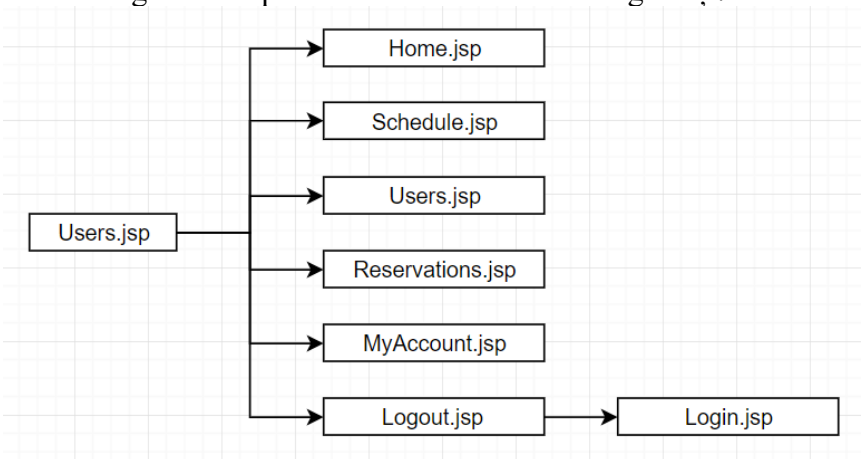


Figura 5.9 Diagrama de navigare pentru utilizatorii înregistrați

5.2.2. Diagrama de pachete

Această aplicație este concepută din 4 pachete (entities, controller, service, repository) în partea de back-end, iar partea de front-end este constituită din 6 pachete (actions, constants, images, pages, reusable_components, server), care la rândul lor pot conține și ele alte pachete / sub-pachete.

În prima parte a acestui capitol sunt prezentate diagramele de pachete atât pentru back-end cât și pentru front-end, ulterior fiind detaliate pachetele și prezentate totodată și diagramele de clase corespunzătoare acestora.

Imaginea de mai jos prezintă diagrama de pachete pentru backend:

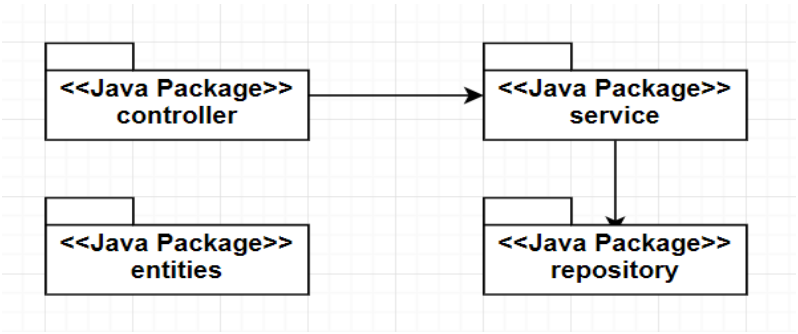


Figura 5.10 Diagrama de pachete pentru backend

Următoarea imagine prezintă diagrama de pachete pentru frontend:

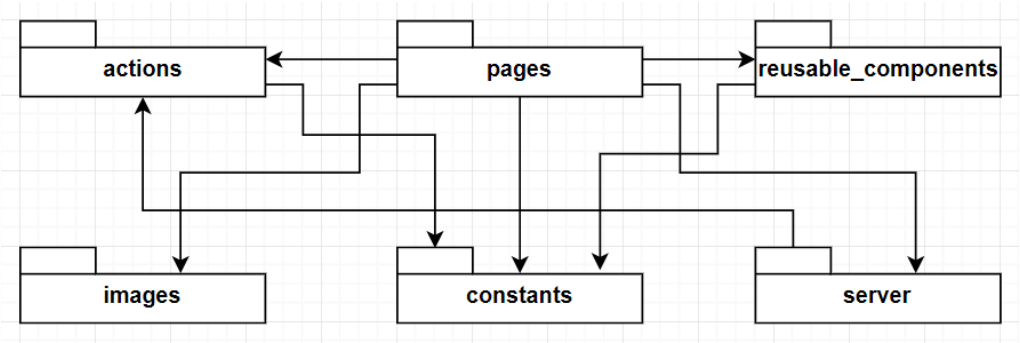


Figura 5.11 Diagrama de pachete pentru frontend

5.2.2.1. Controller

În acest pachet se regăsesc toate clasele specifice pentru partea de control a datelor care se îndreaptă de la client la baza de date, respectiv de la baza de date la client, iar următoarea imagine reprezintă diagrama de clase specifică acestui pachet:

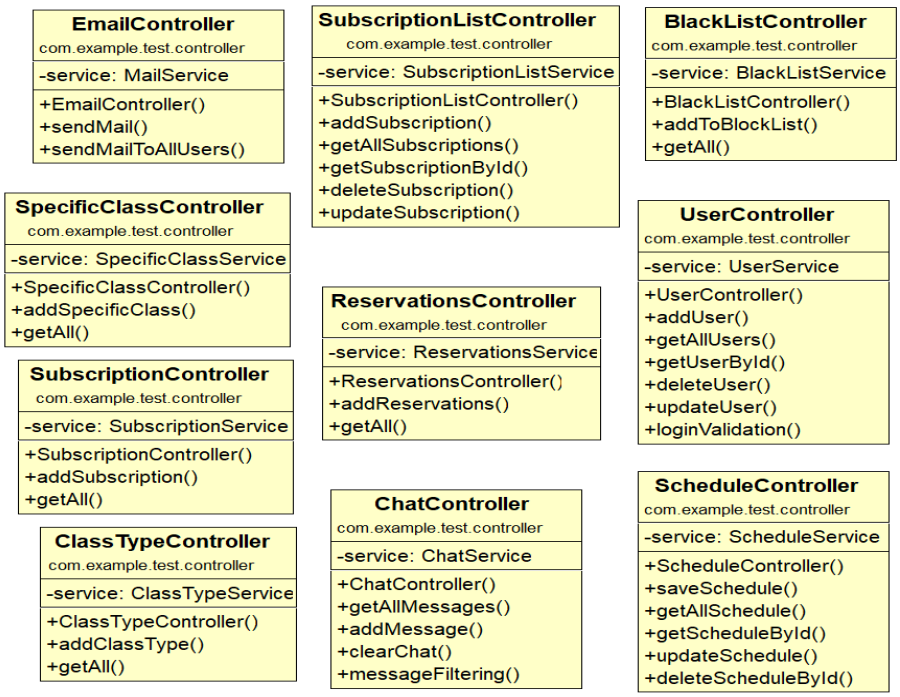


Figura 5.12 Diagrama de clase pentru pachetul controller

5.2.2.2. Service

Acest pachet conține clasele de legătură dintre controller și repository, fiind în același timp cel mai apropiat nivel de accesul la baza de date. Aici este concepută o parte din nivelul de logică, iar mai jos se află diagrama de clase asociată acestui pachet:

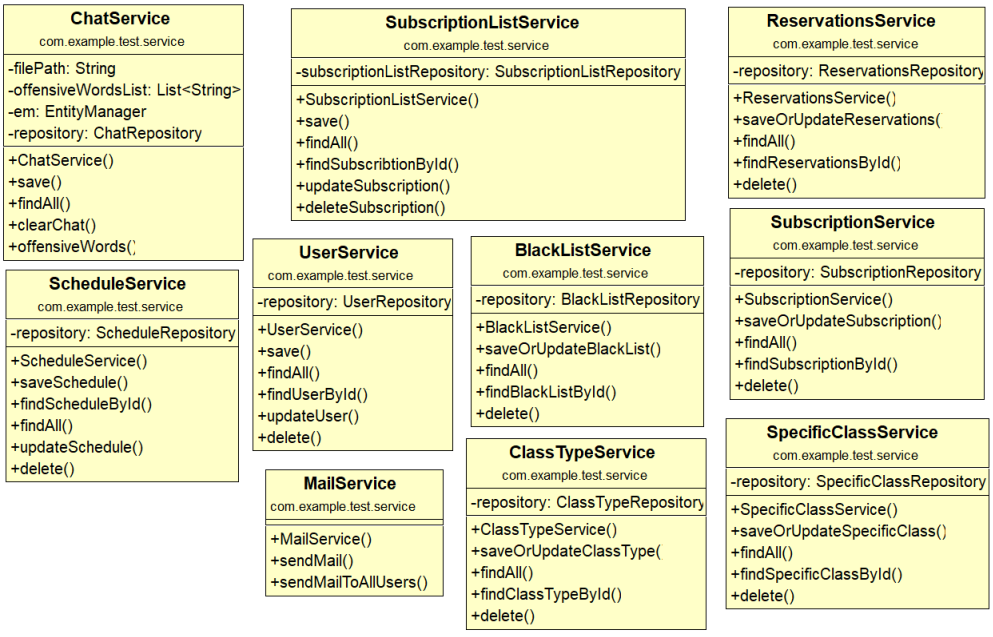


Figura 5.13 Diagrama de clase pentru pachetul service

5.2.2.3. Entities

Pachetul denumit „entities” conține clasele de model ale aplicației, cu alte cuvinte, fiecare clasa este maparea 1 la 1 a tabelului corespunzător din baza de date, și reprezintă obiectele care se transmit ca parametrii metodelor din nivelul de acces al bazei de date, și în care se rețin valorile care trebuie trimise la baza de date sau care sunt primite de la baza de date și trebuie returnate utilizatorilor.

Imaginea următoare reprezintă diagrama de clase a pachetului specific modelelor:

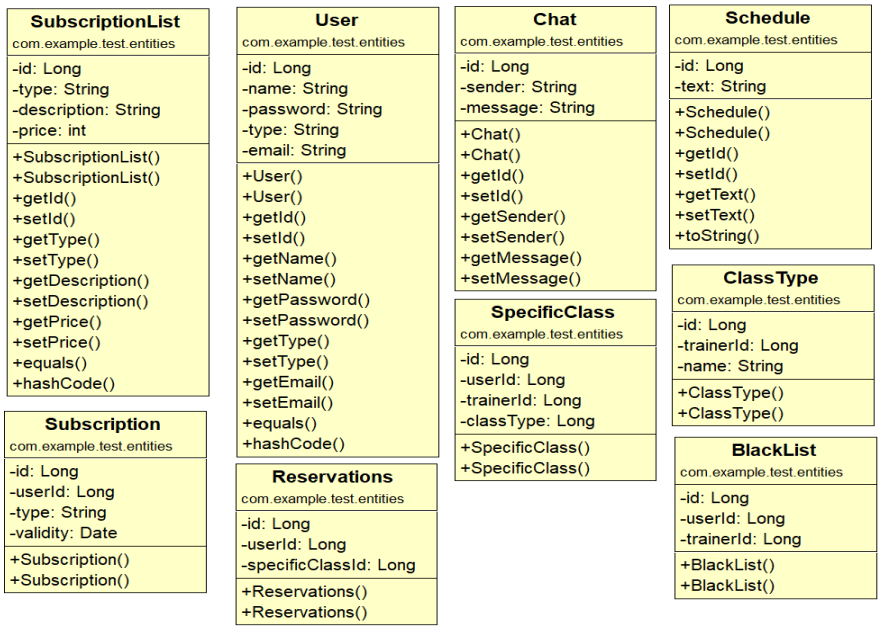


Figura 5.14 Diagrama de clase pentru pachetul entities

5.2.2.4. Repository

În acest pachet sunt regăsite clasele, care constituie accesul la baza de date. Aceste clase sunt defapt interfețe care la rândul lor extind interfața JpaRepository, în care sunt implementate metodele esențiale pentru realizarea operațiilor CRUD, iar pentru utilizarea altor metode precum găsirea unui utilizator în funcție de id-ul acestuia trebuie definite metodele explicit în propria interfață.

Mai jos este prezentată diagrama de clase a pachetului responsabil de accesul la baza de date:

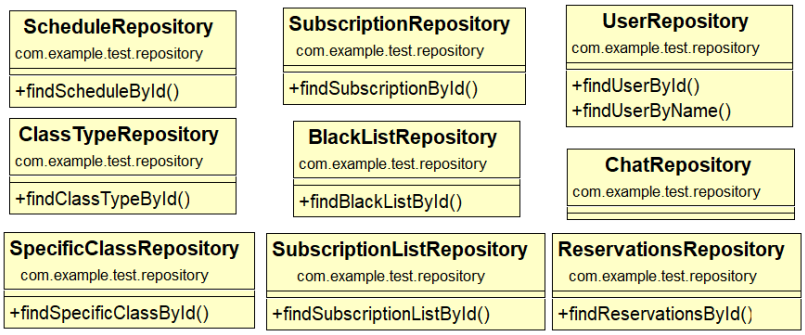


Figura 5.15 Diagrama de clase pentru pachetul repository

5.2.3. Diagrama de implementare

Diagrama de implementare este o diagramă care arată configurația sistemului, și componentele din care acesta este conceput. Scopul unei astfel de diagrame este de a prezenta structura hardware necesară funcționării sistemului.

Aplicația web poate fi accesată de pe orice calculator care are conexiune la internet și îndeplinește cerințele specificate în manualul de instalare și utilizare. Componentele care compun întregul sistem sunt următoarele:

Browserul

Indiferent care este motorul principal de căutare folosit de utilizator (Chrome, Mozzila, Safari, Opera, etc.), aplicația va funcționa exact la fel din punctul de vedere al interacțiunii cu utilizatorul, mici excepții fiind datorate performanțelor fiecărui motor de căutare în parte.

Serverul

Serverul este componenta fizică a sistemului, și cea mai importantă, deoarece este componenta de legătura între client și baza de date. Aici este concepută logica aplicației, și deciziile pe care trebuie să le ia sistemul în funcție de datele de intrare primite de la utilizator.

Baza de date

Baza de date este componenta folosită în sistem pentru stocarea informațiilor. Aceasta primește datele care trebuie prelucrate de la server prin intermediul interfeței JpaRepository, și returnează datele modificate serverului, pentru a le trimite la rândul lui utilizatorului.

Mai jos este prezentată diagrama de implementare a sistemului:

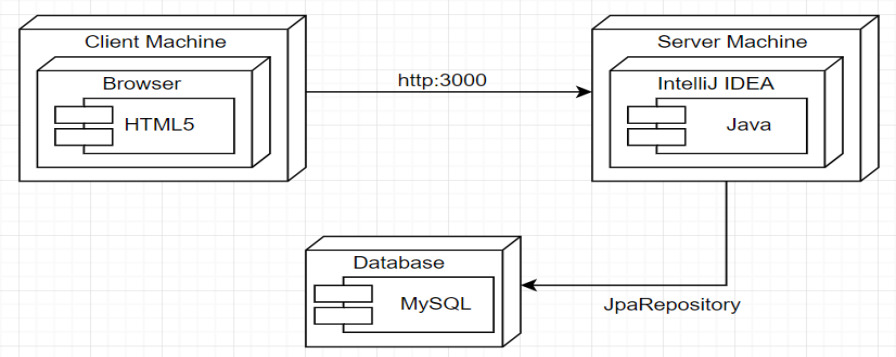


Figura 5.16 Diagrama de implementare (deployment)

5.3. Proiectarea bazei de date

Baza de date folosită în dezvoltarea acestui sistem este MySQL. Aceasta este împărțită în 9 tabele, iar următoarea imagine reprezintă diagrama bazei de date, care cuprinde tabelele și legăturile dintre tabele:

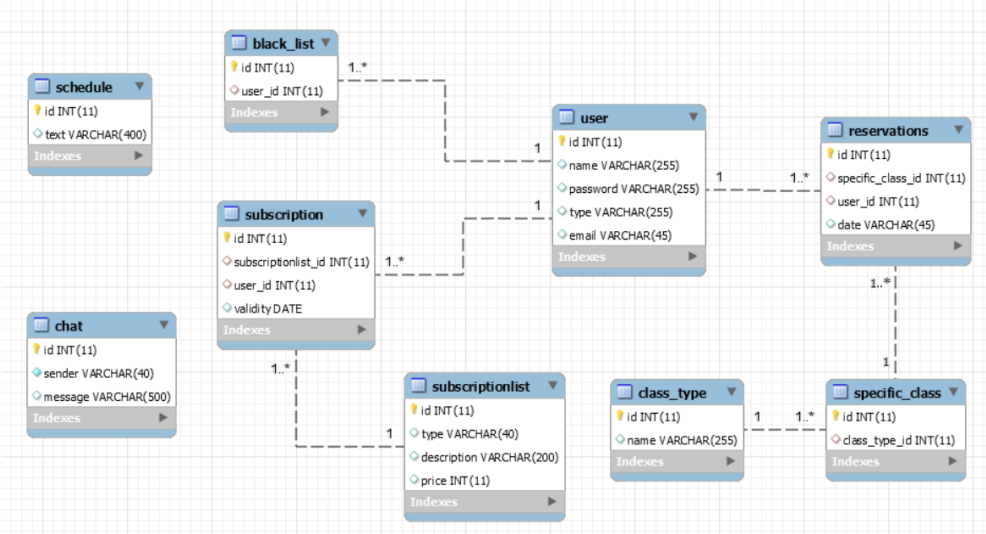


Figura 5.17 Diagrama bazei de date

Baza de date utilizată în dezvoltarea aplicației conține următoarele tabele:

Schedule

În acest tabel sunt stocate informațiile care au legătură strict cu orarul. O parte din informații sunt prestabilite (informațiile din capul tabelului), iar aceastea nu pot fi modificate.

Subscriptionlist

Acest tabel conține informațiile referitoare la abonamente, și anume tipul acestora, o scurtă descriere despre oportunitățile fiecărui abonament, și prețul asociat acestuia.

Chat

Tabelul acesta este construit special pentru funcționalitatea de chat din aplicație, și cuprinde emițătorul mesajului, împreună cu mesajul propriuzis.

User

Tabelul destinat utilizatorilor este conceput astfel încât să stocheze atât informațiile necesare pentru autentificarea fiecărui utilizator, cât și tipul și email-ul acestuia. Acesta este tabelul principal din aplicație, și are cele mai multe conexiuni cu alte tabele.

Blacklist

În acest tabel sunt păstrați clienții care au încălcat regulile și au fost sancționați prin interzicerea accesului în sală.

Reservations

Acesta este un tabel de legătură între utilizator și clasa specifică la care acesta s-a înscris, având totodată rolul de a păstra evidența rezervărilor, respectiv a locurilor disponibile la clasele organizate.

Specific_class

Tabelul acesta conține o referință la tipul explicit al clasei la care utilizatorul s-a înscris.

Class_type

În acest tabel sunt stocate informațiile minime necesare legate de clasele organizate, și anume cum se numesc clasele respective.

Subscription

Tabelul descris în acest paragraf are rolul de a păstra evidența abonamentelor utilizatorilor. Acesta conține toate informațiile necesare pentru acest lucru, și anume id-ul utilizatorului care a achiziționat abonamentul respectiv, id-ul abonamentului achiziționat de către utilizator, și un detaliu foarte important, termenul de valabilitate al abonamentului.

5.3.1. Normalizarea

Normalizarea este o tehnică de generare a unor relații atent concepute astfel încât informațiile stocat în baza de date să fie memorate corect.

În implementarea bazei de date, pentru a evita existanța câmpurilor repetitive, și pentru a ușura modul de acces la informații s-a efectuat normalizarea asupra bazei de date. Mai jos sunt prezentate tipurile de normalizare:

Forma normală 1, cere ca fiecare înregistrare din baza de date să poată fi identificată dupa o cheie primară, și totodată contribuie la eliminarea redundanțelor datorită faptului că fiecare câmp din baza de date cuprinde valori atomice.

Forma normală 2, susține faptul că toate elementele unui tabel trebuie să fie dependente de cheile primare, însă daca elementele sunt dependente numai de o parte a cheilor primare, atunci trebuie separate în tabele diferite. În această situație nu se pune problema, deoarece fiecare tabel conține o singură cheie primară, ceea ce înseamnă că tabelul respectiv este automat în forma normală 2.

Forma normală 3, în această formă normală se definește dependența tranzitivă ca fiind o relație între 1 atribut care este dependent de cheia primară, prin intermediul altui atribut. Pentru a elimina acest tip de dependențe și pentru a îndeplini forma normală 3, se șterg coloanele care sunt tranzitiv dependente de cheia primară, și se creează o noua tabelă cu acestea și atributele prin care erau depenente tranzitiv, care acum reprezintă cheia primară în noul tabel.

Forma normală Boyce-Codd, susține faptul că o relație este în forma normală Boyce-Codd dacă orice determinant din relație este cheie candidat. Tot odată, această formă spune că o relație cu o singură cheie candidat care se află în forma normală 3, se află automat și în forma normală Boyce-Codd. Această formă normală se asigură că în baza de date nu există relații de dependență parțială și tranzitivă.

Baza de date respectă forma normală Boyce-Codd, care este și cea mai restrictivă dintre toate.

5.4. Implementare

Implementarea este cea mai complexă și mai longevivă etapă din dezvoltarea unei aplicații. În acest capitol sunt descrise metodele care contribuie la funcționalitățile cele mai complexe atât din partea de frontend cât și din partea de backend.

5.4.1. Frontend

Server

În această clasă regăsim metoda care realizează conexiunea dintre client și server pentru funcționarea corectă și în timp real a chat-ului.

Metoda respectivă se numește „on” și primește 2 parametrii, primul fiind un șir de caractere care are rolul unei etichete pentru a realiza legarea corectă a socket-ului de pe client cu cel de pe server și invers, iar cel de-al doilea parametru este o expresie lambda, sau cu alte cuvinte o funcție, care la rândul ei poate avea sau nu parametrii.
Un amănunt important de precizat este faptul că prin intermediul acestor socket-uri pot fi transmise atât de la client la server cât și de la server la client orice tip de date.

Am decis să folosesc această metodă pentru implementarea chat-ului în timp real, deoarece este una dintre cele mai simple metode atât de implementat cât și de înțeles, și totodată una dintre cele mai eficiente, deoarece folosește conexiune UDP în loc de TCP, ceea ce favorizează eficiența comunicării.

Imaginea de mai jos evidențiază toate aspectele descrise mai sus:

```
io.on('connection', socket => {

    socket.on('message', data => {
        actions.sendMessage(data).then(response => io.sockets.emit('event', response))
    })

    socket.on('delete', () => {
        actions.clearChat().then(result => io.sockets.emit('acknowledge', result))
    })

    socket.on('disconnect', () => {
    });
});
```

Figura 5.18 Metodă pentru comunicarea în timp real

Table

Această componentă este una dintre cele mai importante la nivel de frontend, deoarece este folosită peste tot unde informațiile trebuie afișate sub formă de tabel, și este refolosită aceeași componentă, și nu este creată una nouă pentru fiecare tabel.

Tabelul este împărțit în 2, partea de header / capul tabelului reprezentând prima linie din tabel, și partea de body / corpul tabelului reprezentând toate celelalte linii din tabel. Un aspect important care face ca această componentă să fie atât de specială este faptul că fiecare celulă din tabel este independentă de celelalte, și poate fi accesată / modificată fără ca celelalte celule din tabel să fie influențate.

Imaginea următoare arată modul în care celulele sunt „desenate” atât în capul tabelului cât și în corpul acestuia:

```
renderHeadingRow = (_cell, cellIndex) => {
    const {headings} = this.props;

    return(
        <Cell
            key = {`heading-${cellIndex}`}
            content = {headings[cellIndex].split("_")[0]}
            header = {true}
            adminRole = {this.props.adminRole}
            id = {headings[cellIndex].split("_")[1]}
            buttonName = {this.props.buttonName}
            buttonKey = {this.props.buttonKey}
            editable = {this.props.editable}
            value = {this.props.value}
            onClick = {this.props.onClick}
            onChange = {this.props.onChange}
        />
    )
};
```

```
renderRow = (_row, rowIndex) => {
    const {rows} = this.props;

    return(
        <tr key={`row-${rowIndex}`}>
            {rows[rowIndex].map((_cell, cellIndex) => {
                return(
                    <Cell
                        key = {`${rowIndex}-${cellIndex}`}
                        content = {rows[rowIndex][cellIndex].split("_")[0]}
                        adminRole = {this.props.adminRole}
                        id = {rows[rowIndex][cellIndex].split("_")[1]}
                        buttonName = {this.props.buttonName}
                        buttonKey = {this.props.buttonKey}
                        editable = {this.props.editable}
                        value = {this.props.value}
                        onClick = {this.props.onClick}
                        onChange = {this.props.onChange}
                    />
                );
            })}
        </tr>
    );
};
```

```
render() {
    const {headings, rows} = this.props;

    this.renderHeadingRow = this.renderHeadingRow.bind(this);
    this.renderRow = this.renderRow.bind(this);

    const theadMarkup = (
        <tr key = "heading">
            {headings.map(this.renderHeadingRow)}
        </tr>
    );

    const tbodyMarkup = rows.map(this.renderRow);

    return(
        <table className = "Table">
            <thead> {theadMarkup}</thead>
            <tbody> {tbodyMarkup} </tbody>
        </table>
    );
}
```

Figura 5.19 Metode pentru „desenarea” celulelor tabelului

Funcțiile din imaginea de mai sus, „renderHeadingRow” și „renderRow” returnează fiecare o celulă, care primește diferiți parametrii, aceștia din urmă fiind opționali. În urma unei scurte analize asupra funcției de desenare (render) din partea dreaptă, se observă faptul că „headings” și „rows” sunt 2 liste care conțin informații. Aceste informații sunt informațiile care ajung în celule, deoarece apelează o funcție numită „map”, al cărei rol este să parcurgă lista cu aceste informații, și pentru fiecare element din listă apelează funțiile din stânga, care returnează celulele corespunzătoare.

Cât despre celulele tabelului, acestea sunt componente independent definite care primesc diferiți parametrii, pe care îi procesează în interiorul lor și care își modifică starea în funcție de evenimentele care se produc asupra celulelor.

O celulă ca și componentă, împreună cu o parte din logică arată în felul următor:

```
export default class Cell extends React.Component {
    render() {
        const {id, value, editable} = this.props
        const cellMarkup = this.props.header ? (
            <th className = "Cell"
                id = {this.props.id}
            >
                {this.props.adminRole && editable && this.props.id === this.props.buttonKey ?
                    <div>
                        <CellInput
                            editable = {this.props.editable}
                            value = {this.props.value === '' ?
                                this.props.content : this.props.value
                            }
                            onChange = {this.props.onChange}
                        />
                    </div>
                    :
                    <div>
                        {this.props.content}
                    </div>
                }
            </th>
```

Figura 5.20 Logica dintr-o celulă a unei tabele

Chat

Una din cele mai importante metode implementată în această componentă este metoda „onKeyDown”, care se execută atunci când este apasată tasta enter pentru a acționa trimiterea mesajului în chat.

Un aspect care face ca această metodă să fie specială este în primul rând faptul că se execută asincron ceea ce aduce un plus de performanță aplicației. Din altă perspectivă, tot aici are loc și o parte de logică, precum faptul că utilizatorul nu poate trimite un mesaj gol, cu alte cuvinte, această metodă nu va face absolut nimic daca utilizatorul nu a introdus niciun mesaj.

O altă caracteristică foarte importantă a acestei componente este faptul că filtrează mesajele astfel încât utilizatorul să nu poată trimite un mesaj care conține cuvinte obscene sau de natură jignitoare.

Imaginea de mai jos prezintă această metodă împreună cu logica implementată:

```
async onKeyDown(e) {
    if(e.key === 'Enter' && this.state.message !== '') {
        await this.setState({
            conversation: {
                senderId: this.props.senderId,
                senderMessage: this.state.message
            }
        })

        let offensiveWords = await Actions.getOffensiveWords()
        let message = this.state.conversation.senderMessage

        if(messageIsClear(message, offensiveWords)) {
            let data = {
                sender: this.state.conversation.senderId,
                message: this.state.conversation.senderMessage
            }

            socket.emit('message', data)
        } else {
            alert('Your message contains inapropriate content')
        }

        document.getElementById('input').value = ''
        this.setState({
            message: ''
        })
    }

}
```

Figura 5.21 Metoda pentru filtrarea și transmiterea mesajului în chat

5.4.2. Backend

Această parte a aplicației a fost generată automat utilizând utilitarul „spring initializr”, care pe lângă faptul că generează fișierele esențiale, utilizatorul poate selecta și framework-uri pe care acest utilitar să le integreze automat, fără ca utilizatorul să mai fie nevoit să importe alte librării.

Mail Service

Una dintre funcționalitățile principale ale acestei aplicații este reprezentată de trimiterea de mail-uri / notificări utilizatorilor atunci când au loc diferite evenimente precum concursuri organizate, închiderea sălii, etc.

Acest serviciu utilizează protocolul SMTP, care folosește metoda „send” din clasa Transport, pentru a transmite mesajul. Mesajul transmis este un obiect de tipul Message, care are 5 parametrii:

1. emițătorul mesajului – setat prin intermediul metodei setFrom
2. receptorul mesajului – setat prin intermediul metodei setRecipients
3. subiectul mesajului – setat folosind metoda setSubject
4. conținutul mesajului – setat prin intermediul metodei setContent
5. data expedierii mesajului – setată prin intermediul metodei setSentDate

Imaginea următoare prezintă metoda responsabilă pentru trimiterea mail-urilor:

```
public synchronized String sendMail(User user, String message) {
    Properties properties = new Properties();
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.port", "587");
    properties.put("mail.smtp.auth", "true");

    Session session = Session.getInstance(properties,
            new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication( userName: "vescan.catalin96@gmail.com",  password:  );
                }
            });
    session.setDebug(true);

    Message mimeMessage = new MimeMessage(session);
    try {
        mimeMessage.setFrom(new InternetAddress( address: "vescan.catalin96@gmail.com",  strict: false));
        mimeMessage.setRecipients(Message.RecipientType.TO, InternetAddress.parse(user.getEmail()));
        mimeMessage.setSubject("My Gym");
        mimeMessage.setContent(message,  s: "text/html");
        mimeMessage.setSentDate(new Date());

        Transport.send(mimeMessage);
    } catch (MessagingException e) {
        e.printStackTrace();
    }

    return "Email successfully sent!";
}
```

Figura 5.22 Metodă pentru transmiterea unui mail

În imaginea de mai sus se poate observa faptul că la începutul acestei metode sunt setate câteva proprietăți. Aceste proprietăți sunt utilizate pentru a preciza protocolului că host-ul folosit este „gmail”, portul utilizat pentru transport este „587” și faptul că protocolul trebuie să folosească metoda de autentificare, vizibilă în momentul în care se creează o nouă sesiune.

Tot din această categorie face parte și următoarea metodă care are un impact foarte mare asupra performanței aplicației, deoarece utilizează stream-uri paralele (multi-threading) cu ajutorul cărora mail-urile sunt trimise mult mai repede, atunci când spre exemplu un mail trebuie trimis la multe persoane:

```
@Async
public String sendMailToAllUsers(List<User> users, String message) {

    users.parallelStream().forEach(x -> sendMail(x, message));

    return "Emails successfully sent!";
}
```

Figura 5.23 Metodă pentru timiterea mai multor mailuri concomitent

Am decis să implementez aceste metode în mod asincron, pentru a câștiga un plus în materie de performanță, codul necesar implementării acestora fiind unul relativ scurt, și ușor de înțeles.

Controlul și Serviciul

În ceea ce privește nivelul de logică al aplicației, acesta este împărțit în control (pachetul controller) și servicii (pachetul service).

Partea de control cuprinde metodele care sunt folosite pentru maparea metodelor POST, GET, și DELETE ale serviciului REST. Acestea folosesc adnotări pentru specificarea tipului fiecărei metode.

O metodă de tip POST este folosită atunci când vrem să trimitem anumite date de la client spre server, cum ar fi de exemplu completarea unui formular. În imaginea de mai jos este reprezentată o metodă de acest fel:

```
@PostMapping("/update")
/Duplicates/
public ResponseEntity<?> updateUser(@Valid @RequestBody User user, BindingResult result) {
    if(result.hasErrors()){
        Map<String, String> errorMap = new HashMap<>();

        for(FieldError err : result.getFieldErrors()) {
            errorMap.put(err.getField(), err.getDefaultMessage());
        }

        return new ResponseEntity<Map<String, String>>(errorMap, HttpStatus.BAD_REQUEST);
    }

    User updatedUser = service.updateUser(user);

    return new ResponseEntity<User>(updatedUser, HttpStatus.OK);
}
```

Figura 5.24 Metodă de tip POST

DELETE este metoda specifică folosită pentru pentru a comunica serverului faptul că utilizatorul dorește ștergerea informațiilor din baza de date. Mai jos este prezentată o imagine cu metoda specifică operației de ștergere:

```
@DeleteMapping("/{user_id}")
public ResponseEntity<?> deleteUser(@PathVariable Long user_id) {
    service.delete(user_id);

    return new ResponseEntity<String>( body: "successfully deleted", HttpStatus.OK);
}
```

Figura 5.25 Metodă de tip DELETE

Metodele de tip GET sunt folosite atunci când solicităm informații de la server, precum sunt tabelele din cadrul aplicației, care în momentul încărcării paginii acestea sunt deja completate cu informații. Imaginea următoare prezintă metoda descrisă:

```
@GetMapping("/all")
public Iterable<User> getAllUsers() {
    return service.findAll();
}

@GetMapping("/{user_id}")
public ResponseEntity<?> getUserById(@PathVariable Long user_id) {
    User user = service.findUserById(user_id);

    return new ResponseEntity<User>(user, HttpStatus.OK);
}
```

Figura 5.26 Metodă de tip GET

După cum se poate observa în imaginea de mai sus, există 2 metode de tip GET, deoarece am vrut să evidențiez faptul că folosind acest tip de metodă, imformația transmisă de la utilizator este încărcată direct în url, în timp ce la operațiile de tip POST acest lucru nu este posibil.

Un alt detaliu important de precizat la acest capitol constă în faptul că metoda de tip GET oferă și siguranță, în timp ce metoda de tip POST nu dispune de această proprietate.

Cât despre servicii, acestea reprezintă un nivel intermediar între control și accesul la baza de date realizat prin intermediul interfețelor JpaRepository. Aici sunt implementate metode menite să transmită informațiile primite de la controller-e spre nivelul de acces la baza de date, și să solicite date de la acesta pentru a le returna spre utilizatori. Acest transfer al datelor se realizează prin intermediul variabilei „repository”, iar metodele apelate sunt cele predefinite în interfața JpaRepository, sau cele declarate în propria interfață care extinde la rândul ei interfața JpaRepository.

Următoarea imagine prezintă nivelul de servicii descris:

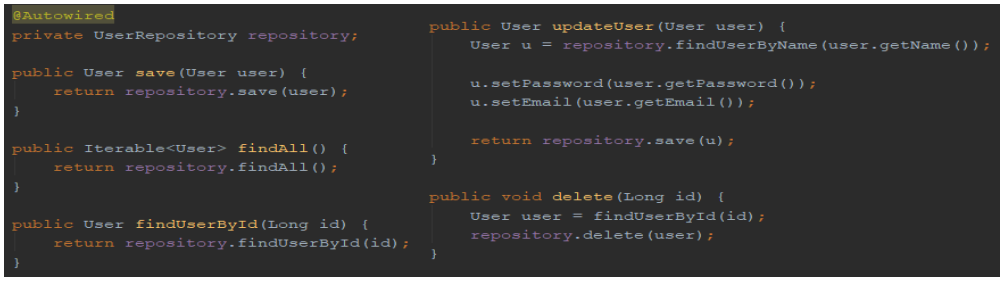


Figura 5.27 Metodele responsabile pentru tranzitul datelor

Accesul la baza de date

Accesul la baza de date se face prin intermediul pachetului de interfețe numit „repository”, în care sunt definite interfețele proprii în care se pot declara noi metode pentru accesul la baza de date, precum căutarea unui utilizator dupa id-ul, sau dupa numele acestuia. Aceste interfețe extind interfața JpaRepository, care oferă accesul la câteva metode precum save, findAll, delete și altele.

În următoarea imagine este prezentată o astfel de interfață:

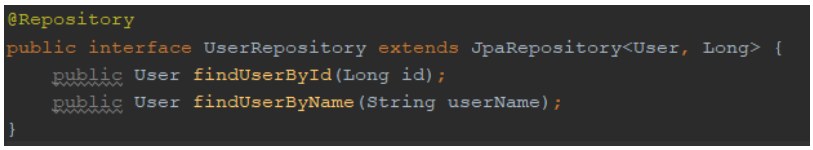


Figura 5.28 Interfața repository

5.5. Unelte folosite pentru implementare

5.5.1. Bitbucket

Bitbucket este o platformă utilizată de dezvoltatorii de aplicații pentru a gestiona mai eficient task-urile pe care fiecare membru din echipă le are de îndeplinit. Nu este cazul acum, însă eu am folosit platforma pentru a salva constant etapele la care am ajuns cu dezvoltarea, pentru a avea o garanție că nu voi pierde tot ce am lucrat în cazul unei greșeli sau a unei defecțiuni.

O alternativă a acestei platforme este GitHub, însă eu am ales să folosesc această platformă deoarece am mai folosit-o și în trecut, ma descurc mai bine cu ea și consider că are o interfață mai prietenoasă.

5.5.2. Metodologia de lucru: Agile[10]

Agile este o metodă de lucru folosită fie în forma simplă, fie combinată cu alte metodologii, în cadrul dezvoltării unei aplicații, pentru a avea control constant asupra echipei de dezvoltare și a sarcinilor membrilor.

Unul dintre beneficiile acestei metodologii constă în faptul că pot fi definite sprint-uri. Sprint-ul este o perioadă definită în care trebuie terminate de implementat toate task-urile propuse la începutul sprint-ului. Acesta este planificat de o persoană numită „scrum master", care face legătura dintre echipa de dezvoltare și persoana responsabilă de business-ul aplicației.

5.5.3. Trello

Trello este un serviciu folosit pentru a gestiona și pentru a ușura manipularea task-urile pe care dezvoltatorii le au de realizat în cadrul unui sprint.

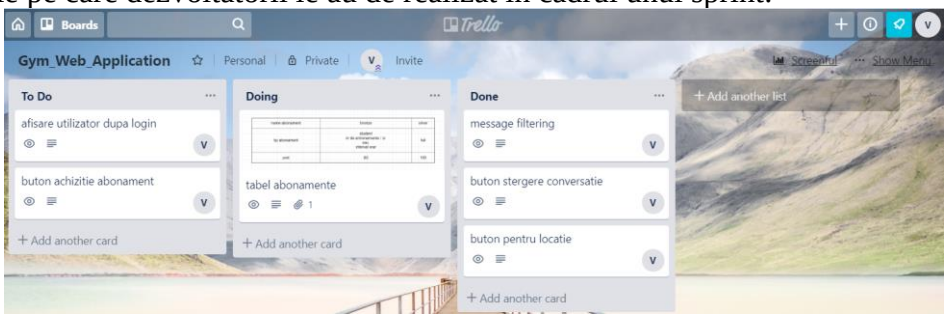


Figura 5.29 Tabla utilizată pentru gestionarea sarcinilor

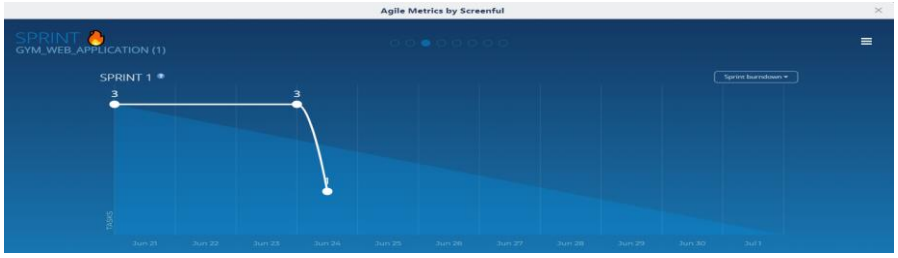


Figura 5.30 Grafic realizat pentru evidența sprint-ului

Trello permite adăugarea unui dashboard pentru a simplifica tot managementul de pe parcursul unui sprint cu tot ce ține de gestionarea task-urilor și realizarea graficelor și statisticilor.

Graficul din imaginea de mai sus a fost creat cu un astfel de utilitar, și anume Agile Metrics by Screenful, și reprezintă parcursul sprint-ului. Mai exact, în partea stângă se poate observa faptul că sunt definite 3 task-uri la începutul sprint-ului, în partea de jos a triunghiului format cu o culoare puțin mai deschisă este reprezentată perioada sprint-ului, de la prima zi (în partea stângă), până la ultima zi (în partea dreaptă), și în funcție de numărul task-urilor îndeplinite graficul coboară sau stagnează.

[10] Ionut Anghel, Cursuri Dezvoltarea și integrarea sistemelor informatice - http://users.utcluj.ro/~ianghel/DISI/2_Curs/

Capitolul 6. Testare şi Validare

În acest capitol este prezentată testarea, de la ce înseamnă ea, până la modul în care se utilizează.

Voi începe capitolul cu o scurtă definiție a ceea ce reprezintă testarea:
Testarea, conform informațiilor din [21], reprezintă alegerea unui set de date specifice pentru o funcționalitate, și evaluarea modului în care funcționalitatea respectivă își îndeplinește atribuțiile, comparând în final rezultatele așteptate, cu rezultatele obținute.

Testările sunt realizate cu scopul de a determina și elimina toate erorile din cadrul programului, înainte ca acesta să fie lansat.

Testarea se poate realiza în diferite moduri, după cum este precizat și în [22], iar câteva dintre acestea sunt prezentate mai jos:

1. Testare funcțională:
   1.1. Testare unitară
   1.2. Testare de integrare
   1.3. Testarea sistemului
   1.4. Testarea utilizabilitității
   1.5. Testarea compatibilității

2. Testare non-funcțională:
   2.1. Testare de performanță
   2.2. Testare de încărcare
   2.3. Testare de recuperare
   2.4. Testare de securitate

În cazul testării unitare se execută teste asupra unui modul / asupra unei funcționalități, timp în care sunt luate în considerare următoarele caracteristici:

- Funcționalitatea care se testează
- Condițiile care trebuie să fie îndeplinite înainte pentru a putea testa funcționalitatea respectivă
- Acțiunile / operațiile care se execută în timpul testării pentru a duce procesul la bun sfârșit
- Rezultatul dorit încă dinaintea începerii testării
- Rezultatul obținut la finalizarea testării

Caz de testare 1: Modificarea informațiilor din orar de către administrator
Precondiții: Informația introdusă în tabel să nu fie un șir de caractere gol
Rezultatul final așteptat: Modificarea informației din celula accesată

Acțiunea efectuată	**Rezultatul așteptat**	**Rezultatul obținut**
Administratorul accesează pagina principală a aplicației	Pagina principală este afișată cu toate butoanele și mesajele	Pagina principală este afișată cu toate butoanele și mesajele
Administratorul accesează butonul „Login”	Redirecționarea utilizatorului la pagina de autentificare	Redirecționarea utilizatorului la pagina de autentificare
Administratorul își introduce credențialele și apasă butonul de logare	Administratorul este redirecționat către pagina corespunzătoare	Administratorul este redirecționat către pagina corespunzătoare
Administratorul accesează pagina cu orarul folosind „Schedule”	Administratorul este redirecționat, și orarul este afișat fără erori	Administratorul este redirecționat, și orarul este afișat fără erori
Administratorul accesează o celula pe care dorește să o modifice, iar după introducere informațiilor apasă butonul „save”	Administratorul așteaptă răspunsul din partea sistemului. Pagina se reîncarcă, și se afișează orarul modificat	Pagina este reîncărcată, și orarul afișat este modificat

Tabel 6.1 Caz de testare pentru modificarea informațiilor din orar

Conform tabelului, rezultatele așteptate în urma execuției fiecărui pas al testării coincid cu rezultatele obținute efectiv în urma testării propriuzise, ceea ce înseamnă ca testarea a fost realizată cu succes, și funcționalitatea își îndeplinește în totalitate atribuțiile. Rezultatul este vizibil și în imaginea de mai jos, în partea sângă fiind orarul înaintea testării, iar în partea dreaptă este orarul după ce testarea s-a terminat:

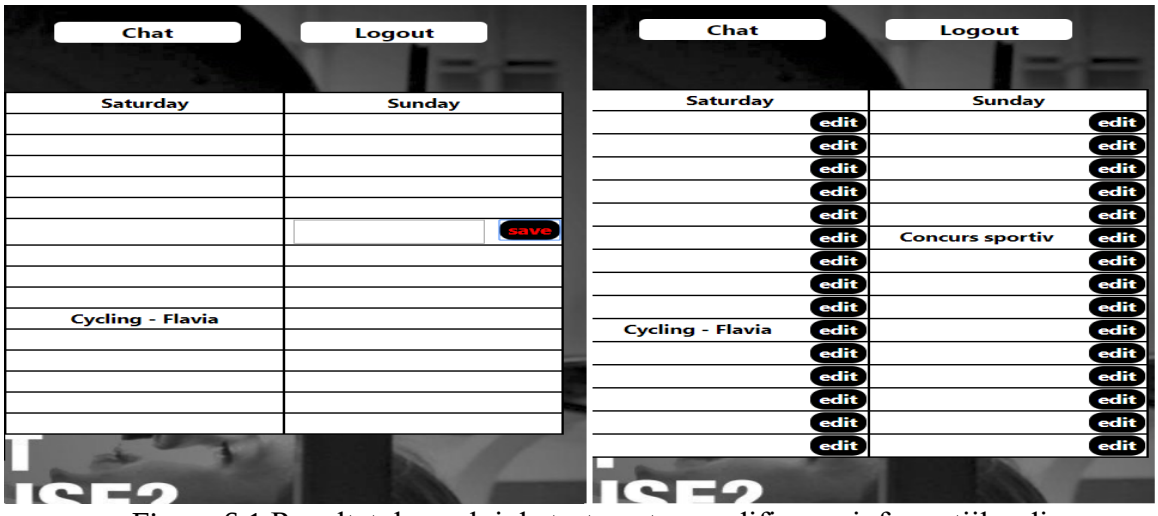


Figura 6.1 Rezultatul cazului de test pentru modificarea informațiilor din orar

Caz de testare 2: Achiziționarea unui abonament
Precondiții: Clientul trebuie să fie logat în aplicație
Rezultatul final așteptat: Asocierea abonamentului corespunzător clientului

Acțiunea efectuată	**Rezultatul așteptat**	**Rezultatul obținut**
Clientul accesează pagina de autentificare	Clientul este redirecționat la pagina de autentificare	Clientul este redirecționat la pagina de autentificare
Clientul se autentifică în aplicație	Clientul este autentificat cu succes	Clientul este autentificat cu succes
Clientul accesează pagina pentru achiziționarea abonamentului folosind butonul „Purchase”	Clientul este redirecționat cu succes la pagina pentru achiziționarea abonamentului	Clientul este redirecționat cu succes la pagina pentru achiziționarea abonamentului
Clientul introduce date valide în câmpurile necesare care trebuie completate	Efectuarea plății rezultată cu asocierea abonamentului corespunzător clientului	Efectuarea plății rezultată cu asocierea abonamentului corespunzător clientului

Tabel 6.2 Caz de testare pentru achiziționarea unui abonament

Conform tabelului, rezultatele așteptate în urma execuției fiecărui pas al testării coincid cu rezultatele obținute efectiv în urma testării propriuzise, ceea ce înseamnă ca testarea a fost realizată cu succes, și funcționalitatea își îndeplinește în totalitate atribuțiile. Rezultatul este vizibil și în imaginea de mai jos, care reprezintă înregistrarea aferentă din baza de date:

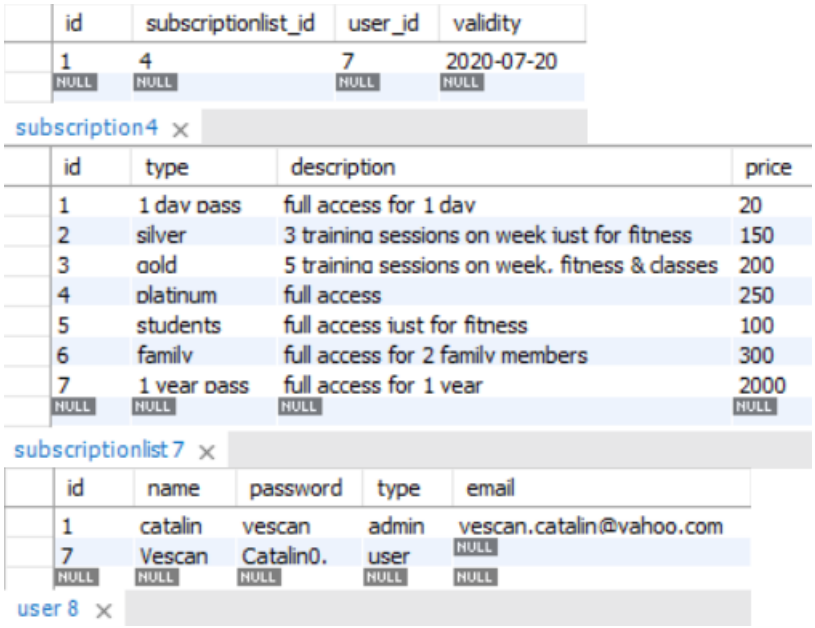


Figura 6.2 Rezultatul cazului de test pentru achiziționarea unui abonament

Primul tabel reprezintă asocierea id-ului corespunzător abonamentului achiziționat, a id-ului corespunzător utilizatorului care l-a achiziționat și a datei în care expiră, într-un singur tabel independent.

În cel de-al 2-lea tabel sunt stocate toate informațiile despre abonamente, iar în cel de-al 3-lea tabel sunt stocate toate informațiile despre utilizatori.

Capitolul 7. Manual de Instalare si Utilizare

Acest capitol descrie atât pașii care trebuie urmați de către utilizatori pentru a realiza cu succes instalarea componentelor pe propriul dispozitiv, cât și resursele de care aceștia au nevoie pentru o funcționare optimă a aplicației.

7.1. Instalare

În mod normal aplicația trebuie să funcționeze pe orice sistem de operare care dispune de un browser și o conexiune la internet, însă este de preferat utilizarea sistemului de operare window, deoarece este cel mai răspândit, și totodată dezvoltarea aplicației a fost realizată tot pe acest sistem de operare.

Instalarea constă din instalarea a 3 componente după cum urmează:

1. **Instalarea utilitarului IntelliJ IDEA și a componentelor necesare**
   1. Descărcarea și instalarea mediului de dezvoltare Java IntelliJ IDEA poate fi efectuată de aici[11], prin selectarea sistemului de operare specific utilizatorului.
   2. Descărcarea și instalarea chitului de dezvoltare (JDK) poate fi efectuată de aici[12]. Pentru compatibilitate maximă și evitarea oricăror inconveniente, versiunea potrivită este JDK 8.
   3. Descărcarea și instalarea software-ului Java care poate fi efectuată de aici[13], de asemenea ca și la pasul anteriori, recomand versiunea Java 8.
   4. Dupa ce am terminat de descărcat și instalat componentele, rulăm executabilul Intellij pentru a porni mediul de dezvoltare.
   5. Pentru importarea proiectului trebuie respectată următoarea secvență:

   File -> open, iar din căsuța care se deschide, navigați până la directorul în care se află sursa pentru a importa proiectul.

2. **Instalarea MySQL workbench**
   1. Descărcarea și instalarea MySQL workbench poate fi efectuată de aici[14], iar recomandarea mea este să se folosească versiunea 6.3 pentru windows.
   2. După ce utilitarul a fost instalat, este necesară configurarea setărilor astfel încât să corespundă cu setările efectuate pe server, și anume parola și numele de utilizator. Utilizatorul prestabilit este „root” și parola „root”.
   3. Odată terminate configurările, trebuie descărcat fișierul „myGym.sql”.
   4. Pentru a importa fișierul trebuie efectuat următorul flux:

   se creează o schemă cu numele „gymwebapp”, apoi din bara de meniuri Server -> Data import -> selectați „Import from Self-Contained File” și navigați la directorul fișierului -> în câmpul „Default Target Schema” introduceți numele schemei specificate la început -> Start Import.

[11] IntelliJ IDEA - https://www.jetbrains.com/idea/download/#section=windows

[12] JDK - https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[13] Software Java - https://www.java.com/en/download/

[14] MySQL Workbench - https://dev.mysql.com/downloads/workbench/

3. **Instalarea modulului pentru front-end**
   1. Înainte de toate, trebuie instalat utilitarul Node JS, iar un mod de a instala acest utilitar poate fi efectuat urmărind tutorialul de aici[15].
   2. Partea de front-end poate fi dezvoltată în orice editor de text, însă eu recomand să se folosească același editor, și anume Visual Studio Code, care poate fi găsit aici[16].
   3. După instalarea editorului de text, este necesară efectuarea următoarelor operații pentru a importa proiectul:

   File -> Open folder, iar în fereastra deschisă navigați către directorul specific proiectului.

   4. Pentru a evita orice conflicte, recomand ștergerea fișierului „node_modules” din directorul proiectului.
   5. Din bara de meniuri deschideți 2 terminale diferite: Terminal -> New Terminal.
   6. Într-un terminal rulați comanda „npm install”, această comandă are rolul de a instala toate pachetele necesare în directorul „node_modules”.
   7. După ce instalarea s-a terminat, într-un terminal rulați comanda „npm start” pentru a rula front-end-ul.
   8. În celălalt terminal deschis, navigați spre directorul „src/server”, iar apoi rulați comanda „nodemon Server.js”. Această comandă pornește serverul care este responsabil pentru comunicarea în timp real din cadrul chat-ului.

4. **Resursele hardware necesare**
   1. Arhitectura sistemului să fie pe 64 de biți.
   2. Memoria RAM să dispună de o capacitate minimă de 4 Gb.
   3. Procesorul trebuie să aibă o frecvență minimă de 2.0 GHz, și minim 2 nuclee.
   4. Placa video trebuie să dispună de o memorie de cel putin 128 Mb.
   5. Placa de rețea cu interfața PCI-E cu o viteza de transfer de 10/100/1000 Mbps.

Rezultatul aplicației poate fi acum vizualizat prin accesarea adresei „localhost:3000”, unde 3000 reprezintă portul asociat.

Un aspect important de precizat este faptul că resursele hardware specificate sunt resursele minime necesare pentru funcționarea la nivel optim al aplicației, cu un număr mic de utilizatori. În cazul în care numărul utilizatorilor care folosesc aplicația concomitent este foarte mare, atunci este posibil ca resursele minime să nu fie în același timp și suficient pentru a respecta cerințele funcționale legate de timpul de răspuns.

[15] Instalare Node JS - https://blog.teamtreehouse.com/install-node-js-npm-windows
[16] Descărcare Visual Studio Code - https://code.visualstudio.com/

7.2. Utilizare

Manualul de utilizare al aplicației are rolul de a îndruma utilizatorul pentru a folosi în mod corect aplicația.

Pentru a acoperi toate cazurile, în acest capitol vor fi prezentate cele mai importante operații pe care le poate efectua fiecare utilizator, modul în care aceștia trebuie să procedeze, și rezultatele la care trebuie să ajungă dacă au efectuat corect pașii necesari.

Important de menționat este faptul că până în momentul înregistrării, respectiv autentificării, toate operațiile prezentate în secțiunea „Vizitator” sunt valabile pentru toți utilizatorii aplicației.

7.2.1. Vizitator

Pagina principală care se afișează în urma accesării url-ului specificat în etapa de instalare, și anume „localhost:3000”, conține o bara de navigare, prin intermediul căreia utilizatorul se poate „plimba” prin aplicație, și un scurt mesaj de întâmpinare, pagină care se afișează de asemenea și la apăsarea butonului „Home”:

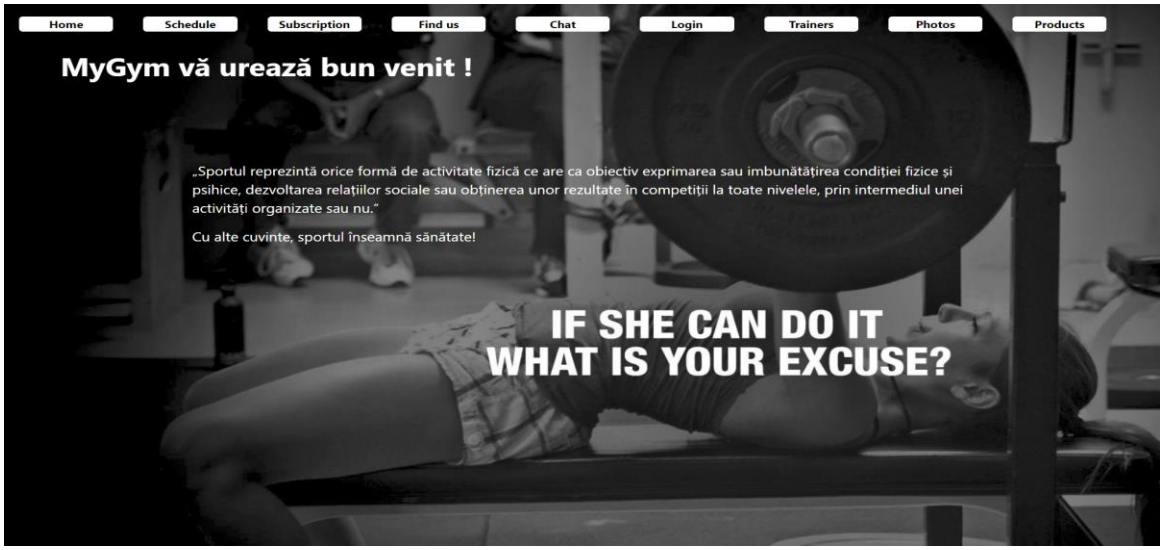
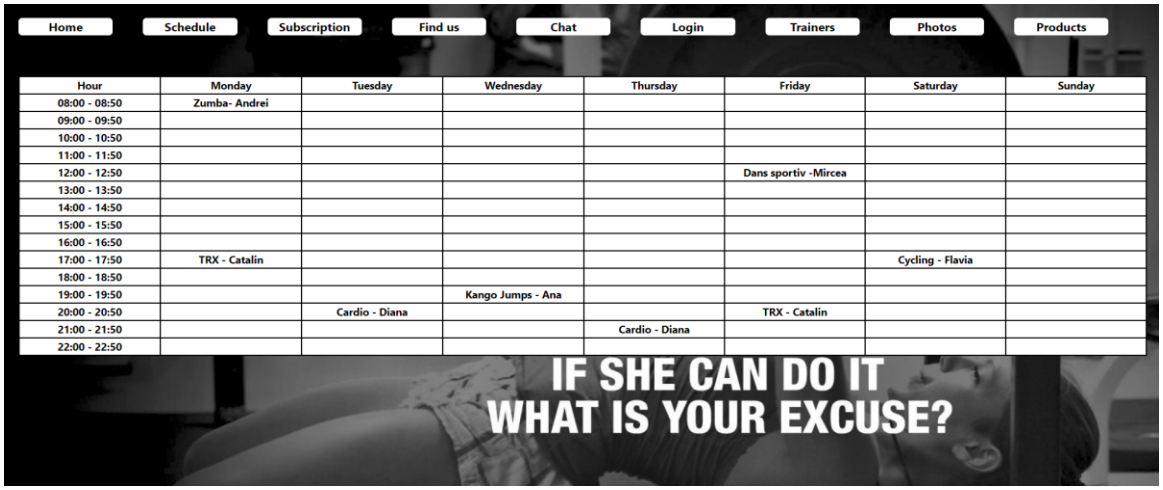


Figura 7.1 Pagina principală a aplicației

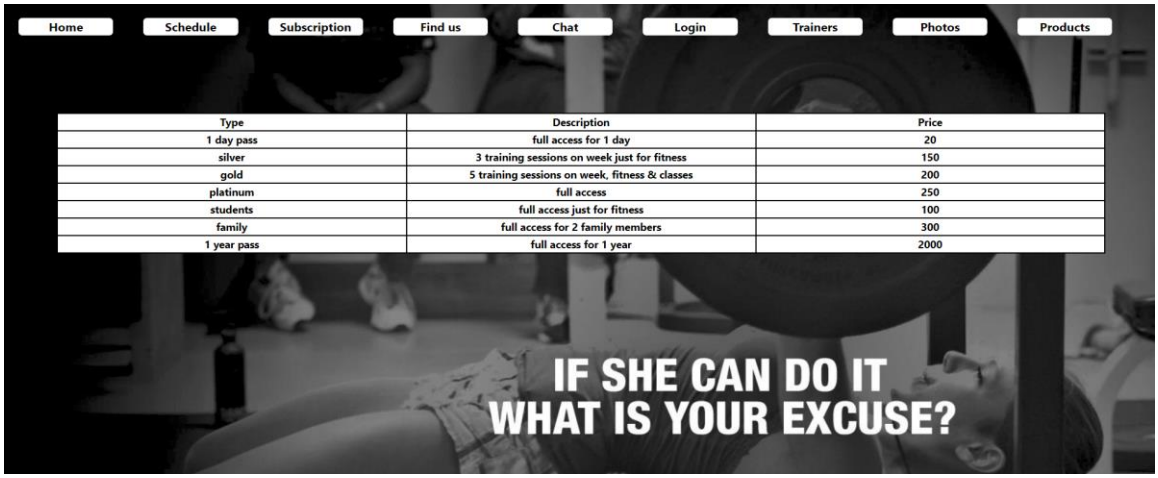
Următoarea funcționalitate disponibilă utilizatorilor constă în vizualizarea orarului și a programelor claselor susținute de antrenori și instructori, care apare în momentul în care utilizatori apasă pe butonul „Schedule”:



Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
08:00 - 08:50	Zumba- Andrei						
09:00 - 09:50							
10:00 - 10:50							
11:00 - 11:50							
12:00 - 12:50					Dans sportiv -Mircea		
13:00 - 13:50							
14:00 - 14:50							
15:00 - 15:50							
16:00 - 16:50							
17:00 - 17:50	TRX - Catalin					Cycling - Flavia	
18:00 - 18:50							
19:00 - 19:50			Kango Jumps - Ana				
20:00 - 20:50		Cardio - Diana			TRX - Catalin		
21:00 - 21:50				Cardio - Diana			
22:00 - 22:50							

Figura 7.2 Vizualizarea orarului

Conform meniului de navigare, următoarea funcționalitate de care beneficiază utilizatorii neînregistrați / care încă nu s-au autentificat în aplicație este vizualizarea abonamentelor. Această funcționalitate este prezentată utilizatorilor atunci când apasă butonul „Subscription”, și le prezintă tipul abonamentului, o scurtă descriere a acestuia, și prețul asociat:



Type	Description	Price
1 day pass	full access for 1 day	20
silver	3 training sessions on week just for fitness	150
gold	5 training sessions on week, fitness & classes	200
platinum	full access	250
students	full access just for fitness	100
family	full access for 2 family members	300
1 year pass	full access for 1 year	2000

Figura 7.3 Vizualizarea abonamentelor

O altă funcționalitate importantă pentru utilizatori este faptul că aceștia pot vizualiza pe hartă locația exactă a sălii, apăsând butonul „Find us”, în această situație însă dispare bara de navigare, iar pentru a reveni utilizatorii sunt nevoiți să folosească butonul de întoarcere al browser-ului. Pentru a marca locația exactă pe hartă a fost folosit un „marker”:

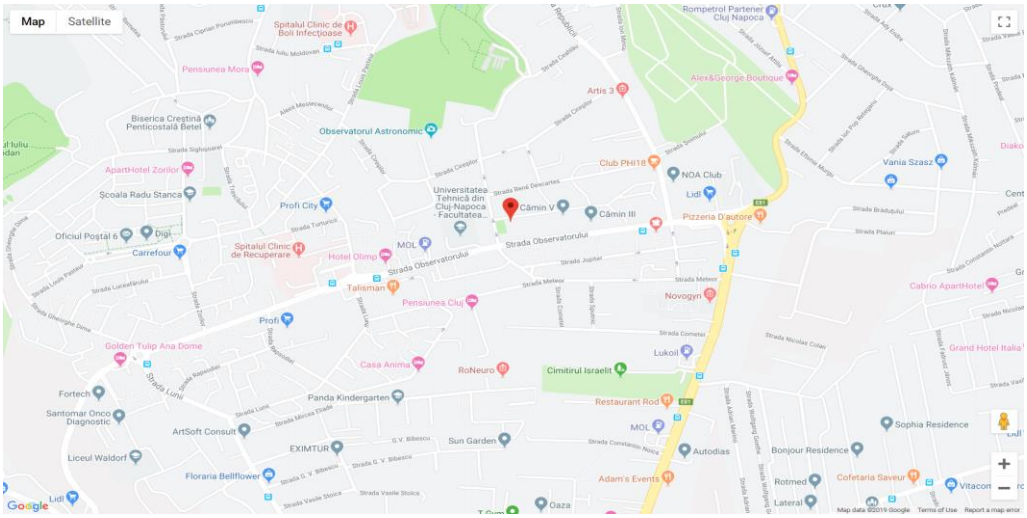


Figura 7.4 Vizualizare locație

Foarte important pentru utilizatorii este să poată lua legătura cu personalul sălii pentru a cere informații suplimentare, iar acest lucru poate fi realizat prin apăsarea butonului „Chat”:

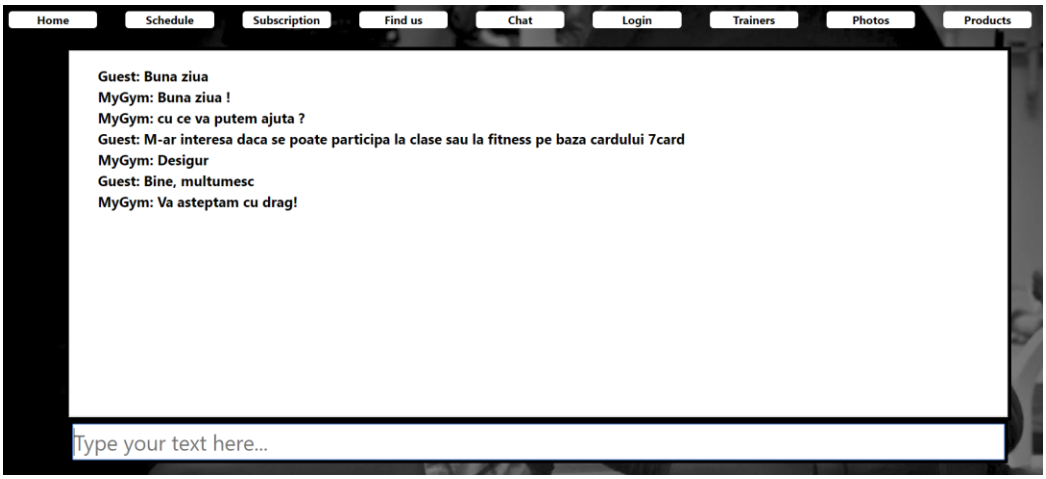


Figura 7.5 Comunicare în timp real de pe pagina vizitatorilor

Utilizând butonul „Login”, utilizatorului i se vor afișa 2 câmpuri, în care acesta trebuie să introducă propriile credențiale în caz de autentificare (Login), sau credențiale noi în caz de înregistrare (Register), care să respecte următoarele condiții:

1. Numele de utilizator trebuie să fie unic, în caz contrar sistemul va înștiința utilizatorul, iar acesta va fi nevoit să introducă un alt nume de utilizator.
2. O parolă puternică, care să fie formată dintr-un număr cuprins între 8 și 12 caractere, să conțină cel puțin o cifră, cel puțin o literă mare, cel puțin o literă mică, cel puțin 1 caracter special, și de preferat, să nu fie cuvinte din dicționar, sau informații despre propria persoană.

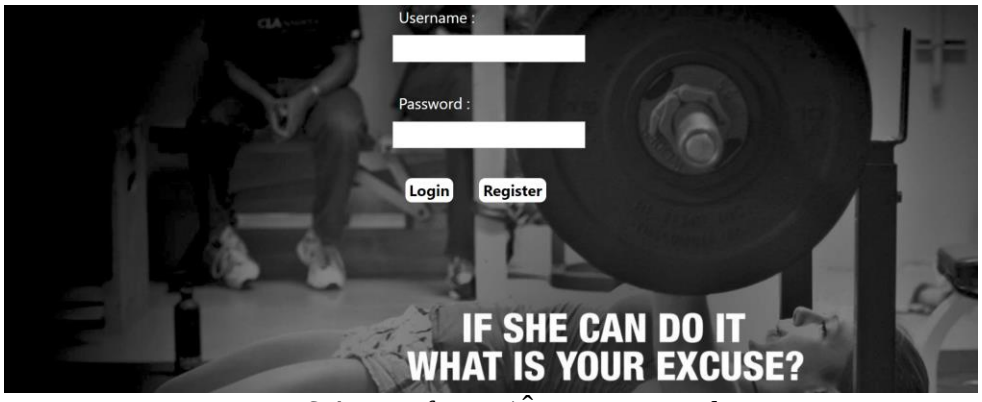


Figura 7.6 Autentificare / Înregistrare utilizator

În urma procesului de autentificare, utilizatorul va fi redirecționat la pagina corespunzătoare în funcție de rolul pe care acesta îl are în aplicație.

7.2.2. Utilizator înregistrat

După ce utilizatorul s-a autentificat cu succes în aplicație, sau când apasă butonul „Home”, acesta este redirecționat către secțiunea corespunzătoare clienților, unde în pagina principală îi este afișat un mesaj de bun venit, alături de numele de utilizator pe care acesta l-a ales în momentul înregistrării și un scurt mesaj motivațional:

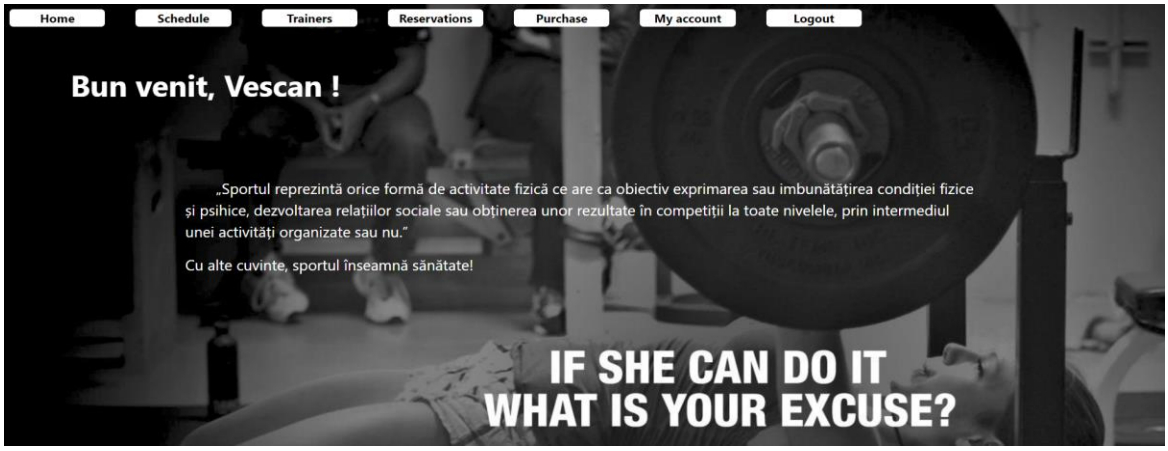
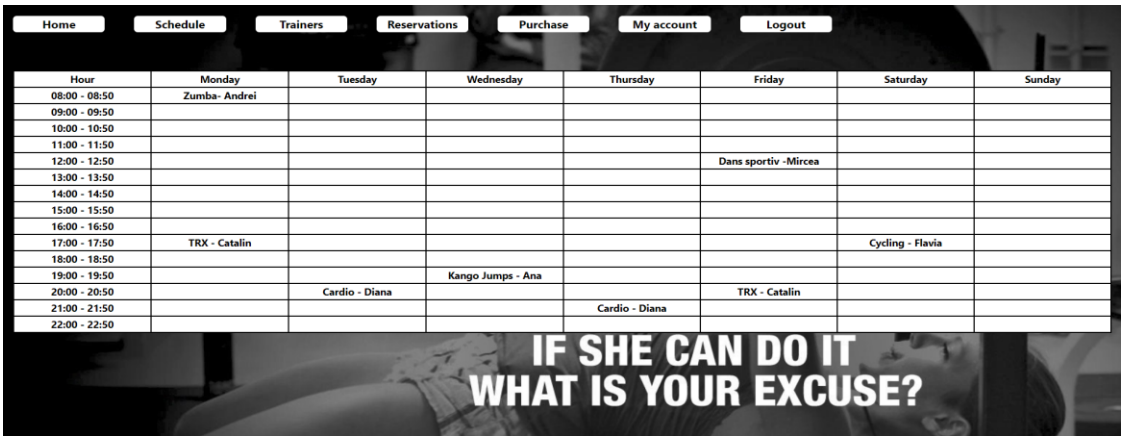


Figura 7.7 Pagina principală a unui client autentificat

De asemenea, la fel ca și în cazul utilizatorilor neînregistrați, și clienții au acces la vizualizarea orarului, prin accesarea butonului „Schedule”:



Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
08:00 - 08:50	Zumba- Andrei						
09:00 - 09:50							
10:00 - 10:50							
11:00 - 11:50							
12:00 - 12:50					Dans sportiv -Mircea		
13:00 - 13:50							
14:00 - 14:50							
15:00 - 15:50							
16:00 - 16:50							
17:00 - 17:50	TRX - Catalin					Cycling - Flavia	
18:00 - 18:50							
19:00 - 19:50			Kango Jumps - Ana				
20:00 - 20:50		Cardio - Diana			TRX - Catalin		
21:00 - 21:50				Cardio - Diana			
22:00 - 22:50							

Figura 7.8 Vizualizarea orarului din pagina clientului

Accesând butonul „My account”, clientul își poate modifica parola contului, și daca dorește să primească notificări pe mail, va trebui să își introducă adresa de mail în câmpul asociat:

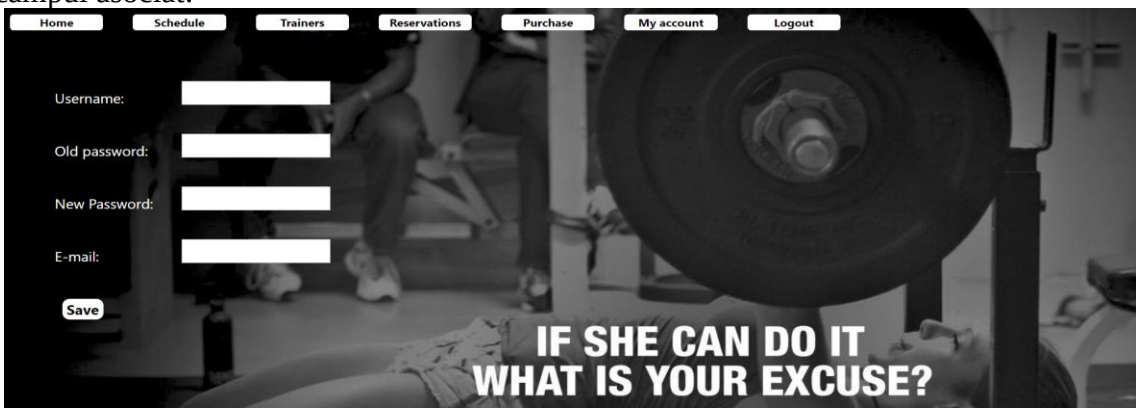


Figura 7.9 Modificarea datelor contului

Utilizatorul odată autentificat în aplicație are acces la o secțiune specifică pentru achiziționarea abonamentelor online, din aplicație. Acesta trebuie să completeze câmpurile din stânga cu datele corecte și valide, iar din partea dreaptă trebuie doar să selecteze din listă tipul abonamentului pe care acesta dorește să-l achiziționeze:

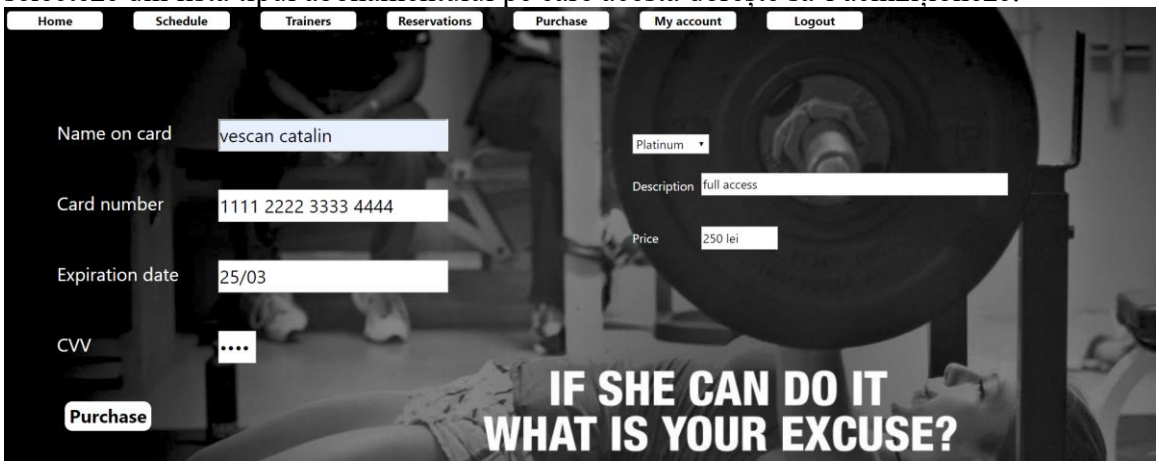


Figura 7.10 Achiziționarea online a abonamentului

Următoarea funcționalitate vine atât în sprijinul clienților, deoarece prin intermediul rezervării le este asigurat un loc la clasa dorită, de la ora dorită de aceștia, dar în același timp vine și în ajutorul personalului, deoarece astfel antrenorii pot ține evidența numărului de locuri disponibile, al persoanelor înscrise, și în urma unei scurte analize, în timp se poate observa care sunt cele mai frecventate clase. Tot ce trebuie să facă utilizatorul este să introducă în câmpurile din dreapta ora de la care vrea să facă rezervarea, respectând formatul din poză, și numele clasei la care vrea să participe, iar în partea stângă este suficient să dea click pe ziua în care dorește să facă rezervarea:

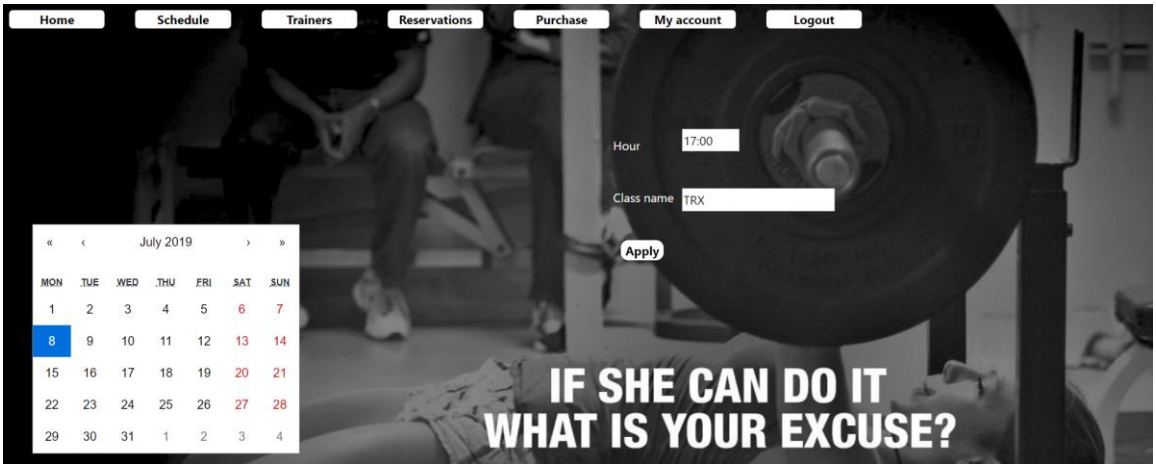


Figura 7.11 Rezervare pentru participarea la clasa dorită

7.2.3. Antrenori

Ținând cont de faptul că suntem la o altă categorie de utilizatori, și anume antrenorii, pe lângă faptul că pot vizualiza orarul, aceștia au și dreaptul de a-l modifica, in scopul de a-și adăuga clasele pe care le susțin, fără să altereze informațiile inițiale:

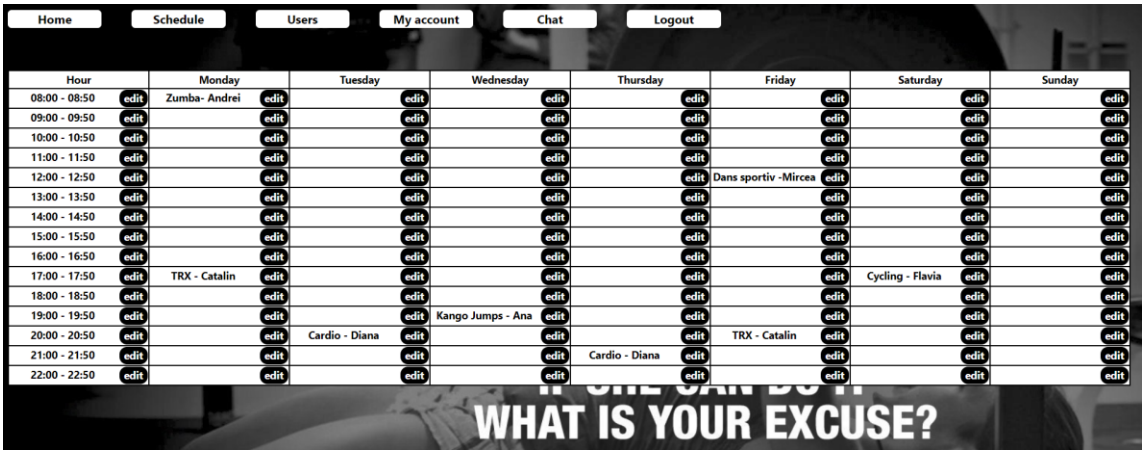


Figura 7.12 Orarul de pe pagina antrenorilor personali

Așa cum utilizatorii neînregistrați dispun de un sistem de comunicare în timp real, la celălalt capăt al „firului”, trebuie să existe și o persoană care să le răspundă acestora. Aici intervin antrenorii, care în limita timpului disponibil vor răspunde mesajelor de pe chat, accesând butonul specific numit „Chat”:

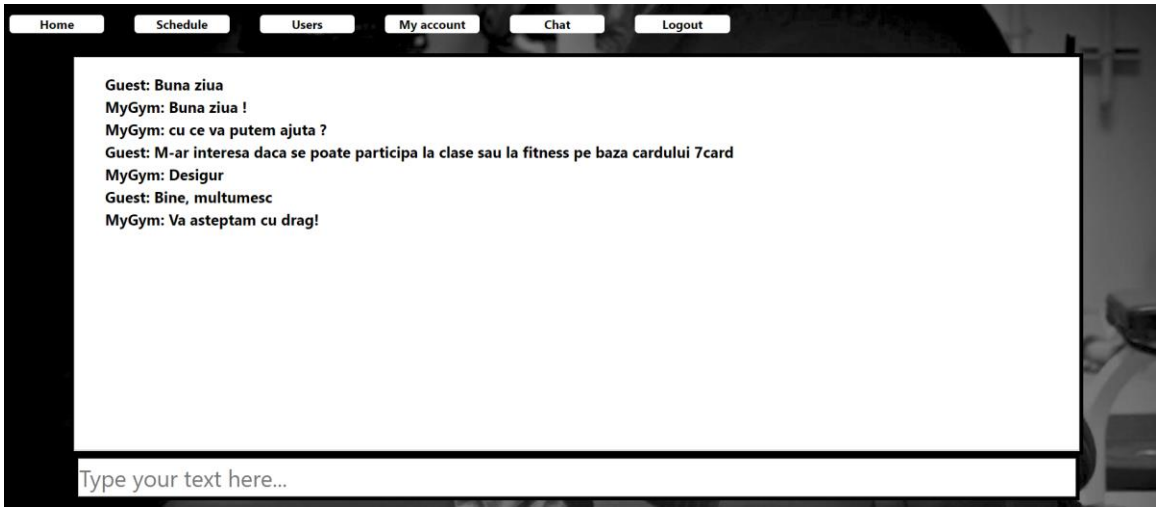


Figura 7.13 Comunicare în timp real de pe pagina antrenorilor

7.2.4. Administrator

Administratorul este acel utilizator care deține controlul complet asupra tuturor funcționalităților aplicației, prin urmare și administratorul are dreptul de a modifica orarul, însă acesta poate modifica și intervalul orar sau zilele în care sala este deschisă:

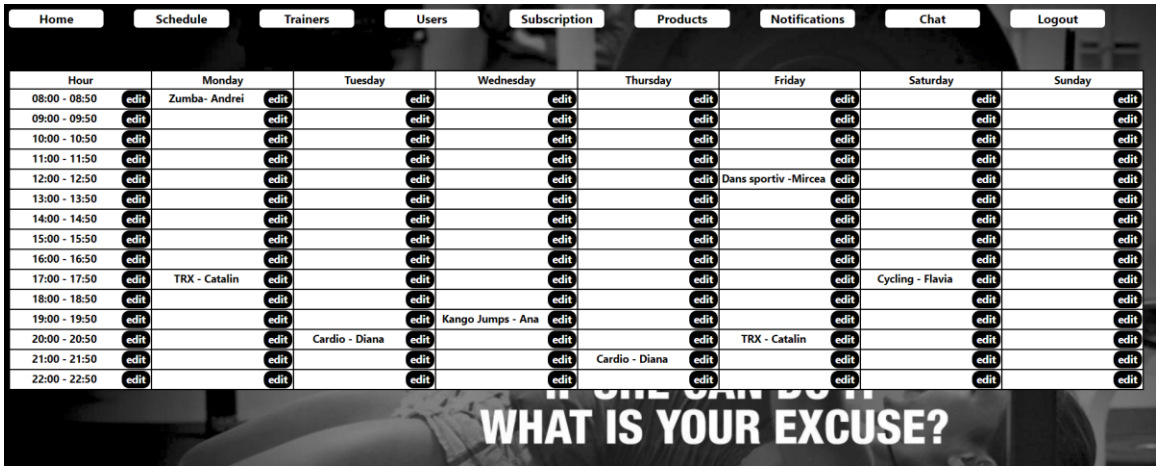


Figura 7.14 Orarul de pe pagina administratorului

Abonamentele fac parte dintr-o funcționalitate destinată clienților, și gestionată de administrator, antrenorii neavând nicio tangență cu abonementele. Astfel, singurii care pot modifica informațiile legate de abonamente sunt administratorii:

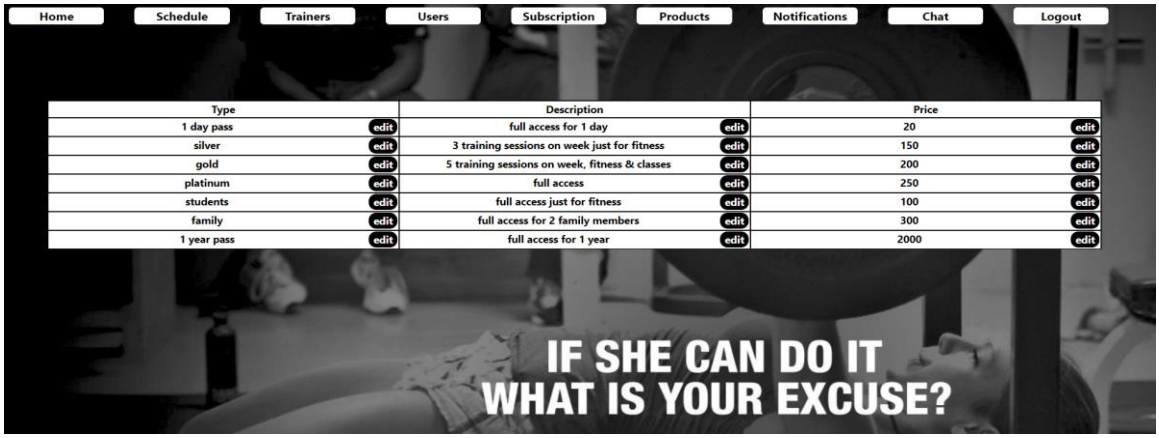


Figura 7.15 Tabelul cu abonamente de pe pagina administratorului

Spre deosebire de alte aplicații, aici administratorii pot trimite notificări pe email, utilizatorilor care și-au adăugat adresa de mail contului înregistrat. Tot ce trebuie să facă este să introducă textul în câmpul destinat mesajelor, vizibil de altfel și în imaginea de mai jos, după care să apese butonul „Send”, pentru a trimite mesajul utilizatorilor:

Figura 7.16 Trimitere notificări prin mail

Datorită faptului că nu întotdeauna antrenorii sunt disponibili, o parte din atribuțiile acestora pot fi îndeplinite și de administratori, iar una dintre acestea este comunicarea cu clienții, de asemenea prin intermediul chat-ului. Ceea ce diferențiază administratorul de antrenori este faptul că administratorul poate să șteargă în orice moment toată conversația:

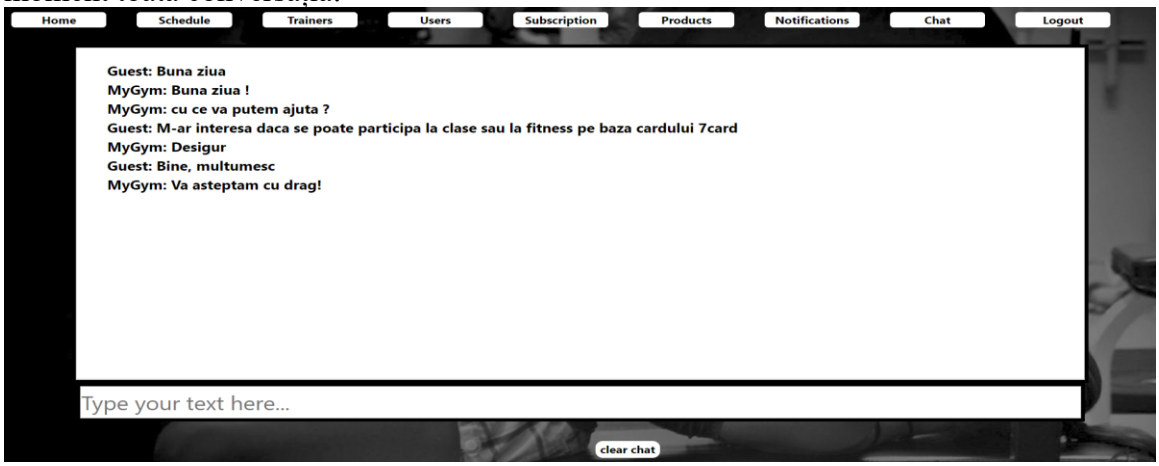


Figura 7.17 Comunicare în timp real de pe pagina administratorului

Capitolul 8. Concluzii

În acest capitol sunt prezentate funcționalitățile realizate prin implementarea aplicației, și totodată funcționalități noi care pot fi implementate ulterior pentru a face aplicația mult mai performantă și mai utilă.

8.1. Realizarea obiectivelor propuse

Sistemul realizat reușește să își îndeplinească cu succes atât obiectivul principal, cât și obiectivele secundare care au fost propuse și specificate și în capitolul 2, obținând astfel cu succes o aplicație web care să ajute prin funcționalitățile ei la un management mai eficient al clienților și al resurselor precum abonamentele, sau rezervările la clase organizate.

Conform propunerilor, aplicația dispune de 4 tipuri de utilizatori, fiecare dintre aceștia având privilegii și restricții în funcție de tipul fiecăruia. Aplicația a fost astfel concepută încât interfața grafică să fie una simplă, prietenoasă – oferind feedback constant – și ușor de utilizat.

În concluzie, aplicația oferă atât clienților, cât și personalului sălii și inclusiv administratorului un mod mult mai plăcut de a-și gestiona timpul atunci când vine vorba fie de plata sau evidența abonamentelor, fie de asigurarea unui loc sau gestionarea persoanelor care participă la clasele organizate.

8.2. Devoltări ulterioare

Ținând cont de faptul că tehnologia avansează foarte repede, sistemul a fost proiectat astfel încât să poată fi ușor de dezvoltat atunci când vine vorba de a modifica o funcționalitate, sau pur și simplu adăugarea unei noi funcționalități, fără să afecteze funcționalitățile deja existente.

O primă îmbunătățire care poate fi adusă sitemului este reprezentată de asocierea unui card (unic) pentru fiecare client înregistrat, având ca scop principal verificarea validității abonamentului clientului în momentul în care acesta intră în sală, prin scanarea cardului.

Altă îmbunătățire de care se poate bucura această aplicație este reprezentată de dezvoltarea unei aplicații mobile, disponibilă pe toate sistemele de operare, și care ar putea fi mult mai rapidă decât aplicația web.

O a 3-a dezvoltare a sistemului constă în posibilitatea conectării aplicației, fie în varianta web, fie în varianta mobile, cu un ceas inteligent, iar informațiile preluate de acesta să fie transmise aplicației pentru diferite procesări și analize precum pulsul din timpul antrenamentului, intensitatea antrenamentului, tipul antrenamentului, sau altele, în scopul efectuării unor sugestii legate de alimentația corespunzătoare efortului depus, posibilitatea setării unor praguri pe care clientul își dorește să le atingă, sau pur și simplu efectuarea unor statistici / grafice, pe care clientul le poate analiza ulterior.

Bibliografie

[1] Afërdıta Berisha – Shaqiri, Management Information System and Decision-Making, Vol 3, No 2 2014,
Disponibil la: https://www.mcser.org/journal/index.php/ajis/article/view/2943

[2] Micheu Katalin, Curs MVC (Model View Controller), 2019,
Disponibil la:
http://www.science.upm.ro/~traian/web_curs/Web_tech/lucr_stud/Micheu_Katalin.pdf

[3] Sabin Buraga, Curs Dezvoltarea Aplicațiilor Web, 2014,
Disponibil la: https://www.slideshare.net/busaco/web03-dezvoltareaaplicatiilorweb-serviciiapimashuparhitectura

[4] Mihai Stancu, Curs Dezvoltarea Aplicațiilor Web, 2019,
Disponibil la: http://inf.ucv.ro/documents/mihais/DAW/DAW-1.pdf

[5] Ionut Anghel, Cursuri Dezvoltarea Aplicațiilor Web, 2019,
Disponibil la: http://users.utcluj.ro/~ianghel/DAW/2_Curs/

[6] Oracle, Documentația oficială, 2019,
Disponibil la:
https://www.oracle.com/technetwork/java/javaee/documentation/index.html

[7] Roman Andra, Lucrare de licență Sistem pentru managementul unui complex sportiv, 2017,
Disponibil la:
http://users.utcluj.ro/~civan/thesis_files/2017_RomanA_Managcomplexsp.pdf

[8] Matthew Tyson, JavaWorld, 2019,
Disponibil la: https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

[9] Tavi Bolog, RESTful Web Services folosind Jersey, 2019,
Disponibil la: https://www.todaysoftmag.ro/article/81/restful-web-services-folosind-jersey

[10] Crystal Mind Academy, Procesarea Formularelor. Diferențele dintre GET și POST, 2019,
Disponibil la: https://www.invata-online.ro/php-mysql/procesarea-formularelor--get--post/diferentele-dintre-get-post

[11] Sabin Buraga, Curs Tehnologii Web – Servicii Web, 2019,
Disponibil la:
https://profs.info.uaic.ro/~busaco/teach/courses/web/presentations/web11ServiciiWeb-REST.pdf

[12] Margaret Rouse, MySQL, 2019,
Disponibil la: https://searchoracle.techtarget.com/definition/MySQL

[13] Oracle, Documentația oficială, 2019,
Disponibil la: https://www.oracle.com/ro/mysql/

[14] Cosmina Ivan, Introducere în Baze de Date, 2019,
Disponibil la:
https://ftp.utcluj.ro/pub/users/civan/IBD/2_LABORATOR/10_Normalizare/IBD_Lab10.pdf

[15] Dan Pescaru, Curs Ingineria Cerințelor, 2004,
Disponibil la: http://staff.cs.upt.ro/~dan/curs/fis/Cap3_Cerinte.pdf

[16] Adriana Guran, Curs Utilizabilitatea sistemelor interactive, 2019,
Disponibil la: www.cs.ubbcluj.ro/~adriana/HCI/Curs_09.ppt

[17] SRAC, Securitatea Informațiilor ISO / CEI 27001:2013, 2013,
Disponibil la: https://www.srac.ro/ro/securitatea-informatiilor-isoiec-27001

[18] Ioan-Cosmin Mihai, Asigurarea securității informațiilor, 2019,
Disponibil la: http://www.securitatea-informatiilor.ro/solutii-de-securitate-it/asigurarea-securitatii-informatiilor/

[19] Ioan Salomie, Curs Sisteme Distribuite, 2019,
Disponibil la: http://www.coned.utcluj.ro/~salomie/DS_Lic/4_Slides/

[20] Florentina Suter, Curs Evaluarea performanțelor sistemelor informatice,
Disponibil la: http://dld.fmi.unibuc.ro/IS/EvalPerfSistInform/CURS1.pdf

[21] Ștefan Stăncescu, Lecu Radu, Tică Andra, Vidrașcu Mihai, Curs Implementarea, testarea, verificarea și validarea produselor software, 2011,
Disponibil la:
http://stst.elia.pub.ro/news/IS/Teme%20IS%202011_12/Lecu%20Tica%20Vidrascu%20IMPLVER%20442A.pdf

[22] UPB, Curs Noțiuni despre testarea software și soluții disponibile, 2019,
Disponibil la: http://www.euroqual.pub.ro/wp-content/uploads/C10-Testare-sw.pdf

Anexa 1

Anexa 2

Abrevierea	**Definiția**
MIS	Management Information System
POM	Project Object Model
ORM	Object-relational mapping
WS	Web Service
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
GNU	Gnu's Not Unix
SGBD	Sistemele de gestiune a bazelor de date
LAMP	Linux, Apache, MySQL, PHP
CRUD	Create, Read, Update, Delete
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
SMTP	Simple Mail Transfer Protocol
JDK	Java Development Kit
FN	Forma Normală