



UNIVERSITATEA DIN BUCUREŞTI



FACULTATEA DE
MATEMATICĂ ŞI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de disertație

**ATP Tour
PLATFORMĂ WEB PENTRU
GESTIUNEA CIRCUITULUI DE
TENIS MASCULIN**

Absolvent

Gherghe Ioana-Delia

Coordonator științific

Lect. Dr. Mihai Gabriela

București, iunie-iulie 2024

Cuprins

INTRODUCERE	7
I BAZE DE DATE.....	9
I.1 Baze de date relaționale.....	9
I.1.1 Modelul relațional	9
I.1.2 Structura bazelor de date relaționale	10
I.1.3 Consistența datelor în modelul relațional.....	11
I.1.4 Blocarea și concurența în bazele de date.....	11
I.1.5 Avantajele bazelor de date relaționale	12
I.1.6 Baza de date relațională a viitorului	13
I.2 Baza de date Oracle 19c	13
I.2.1 Istoria bazelor de date Oracle.....	14
I.2.2 Caracteristici noi în Oracle 19c	15
I.3 Tipuri de date geospațiale în Oracle	17
I.3.1 Elemente geometrice	18
I.3.2 Tipul de obiect SDO_GEOOMETRY	19
I.3.3 Sistem de coordonate	22
I.3.4 Toleranță.....	22
I.3.5 Indexarea datelor spațiale.....	24
I.3.6 Metode SDO_GEOOMETRY	24
I.3.7 Operatori, funcții și proceduri spațiale	25
II TEHNOLOGII UTILIZATE.....	27
II.1 Oracle Application Express	27
II.1.1 Page Designer	28
II.1.2 SQL Workshop	29

II.1.3 Servicii RESTful în Oracle APEX.....	30
II.2 React Native	33
II.2.1 Dezvoltare <i>cross-platform</i>	33
II.2.2 <i>Hook-uri</i> React	34
II.2.3 Componente de bază în React Native	34
II.3 Expo	35
II.4 Typescript.....	37
III FORMULAREA PROBLEMEI ȘI PROIECTAREA BAZEI DE DATE	38
III.1 O scurtă istorie a tenisului	38
III.2 Formularea problemei.....	38
III.3 Diagramele bazei de date.....	40
IV DEZVOLTAREA APLICAȚIEI WEB	43
IV.1 Descrierea aplicației	43
IV.2 Autentificare și autorizare	47
IV.3 Gestiunea informațiilor.....	49
IV.4 Jucători.....	51
IV.5 Turnee.....	54
IV.6 Planificarea călătoriilor	59
IV.7 Bilete.....	61
IV.8 Quiz-uri.....	65
IV.9 Forum de discuții	66
V DEZVOLTAREA APLICAȚIEI MOBILE	67
V.1 Descrierea aplicației.....	67
V.2 Implementarea REST API-ului.....	67
V.3 Autentificarea.....	69
V.4 Ecranele aplicației	70

V.5 Profilul jucătorilor.....	71
V.5.1 Google Places API.....	71
V.5.2 Actualizarea profilului	74
CONCLUZII.....	76
BIBLIOGRAFIE	77

Listă de figuri

Figura I.1 – Cronologia versiunilor Oracle	15
Figura I.2 – Capabilitățile de utilizare a bazei de date Oracle 19c	16
Figura I.3 – Tipuri geometrice.....	18
Figura II.1 – Page Designer	29
Figura II.2 – SQL Workshop.....	30
Figura II.3 – RESTful Services	32
Figura II.4 – Pagina principală Expo Go	36
Figura III.1 – Entitățile identificate	40
Figura III.2 – Diagrama entitate-relație	41
Figura III.3 – Diagrama conceptuală.....	42
Figura IV.1 – Diagrama UML <i>use case</i> 1	44
Figura IV.2 – Diagrama UML <i>use case</i> 2	45
Figura IV.3 – Diagrama UML secvențială	46
Figura IV.4 – Pagina de creare a contului	47
Figura IV.5 – Pagina principală.....	48
Figura IV.6 – Pagina de atribuire a rolurilor	49
Figura IV.7 – Pagina de administrare	49
Figura IV.8 – Pagina de gestiune a meciurilor în funcție de turneu	50
Figura IV.9 – Formularul de adăugare/editare a unui meci.....	51
Figura IV.10 – Pagina clasamentului mondial	52
Figura IV.11 – Pagina detaliilor despre jucători.....	53
Figura IV.12 – Pagina scorului din meciurile directe.....	53
Figura IV.13 – Pagina statisticilor unui jucător.....	54
Figura IV.14 – Pagina programului turneeelor	55
Figura IV.15 – Pagina calendarului turneeelor	55
Figura IV.16 – Pagina cu detaliile unui turneu.....	56
Figura IV.17 – Pagina listei de participanți	57
Figura IV.18 – Pagina tabloului de concurs	57
Figura IV.19 – Pagina rezultatelor meciurilor încheiate	57
Figura IV.20 – Pagina programului meciurilor	58

Figura IV.21 – Pagina programelor TV ce transmit turneul.....	58
Figura IV.22 – Pagina planificării călătoriilor.....	59
Figura IV.23 – Alegerea locației pe hartă.....	60
Figura IV.24 – Pagina turneeelor rezultate	61
Figura IV.25 – Pagina biletelor unui turneu	62
Figura IV.26 – Pagina de adăugare a biletelor în coș	63
Figura IV.27 – Pagina coșului de cumpărături	64
Figura IV.28 – Pagina biletelor achiziționate	65
Figura IV.29 – Pagina întrebărilor.....	65
Figura IV.30 – Pagina comentariilor unui subiect.....	66
Figura V.1 – Pagina de autentificare	69
Figura V.2 – Pagina principală	70
Figura V.3 – Pagina de schimbare a temei	70
Figura V.4 – Funcționalitatea de completare automată a locației	73
Figura V.5 – Selectarea intervalului orar.....	73
Figura V.6 – Ecranul profilului	74
Figura V.7 – Pagina de vizualizare a profilurilor pentru administratori	75

Listă de tabele

Tabel I.1 – Valorile identificatorului TT din atributul SDO_GTYPE	20
Tabel I.2 – Metode SDO_GEOMETRY	25
Tabel I.3 – Operatori spațiali	26

Rezumat

Aplicația web ATP Tour dezvoltată în această lucrare cu ajutorul tehnologiei Oracle APEX gestionează cele mai importante activități din cadrul circuitului de tenis masculin, precum performanțele jucătorilor, calendarul competițional și detaliile despre turnee, meciurile programate, rezultatele partidelor și vânzarea biletelor. De asemenea, se pune accent pe divertismentului și interacțiunea dintre utilizatori prin intermediul forumului de discuții și a competiției trivie anuale.

Spre deosebire de alte produse similare existente pe piață, aplicația include un modul dedicat asistării fanilor în planificarea călătoriilor în funcție de cele mai apropiate turnee. În plus, aceasta este acompaniată de aplicația mobilă ITIA-App dezvoltată în React Native destinată organizării programelor și locațiilor sportivilor pentru testările anti-doping.

În funcție de rolurile utilizatorilor, există cinci niveluri de acces: administratori, editori, useri, jucători și publicul larg. Primele patru dintre acestea necesită crearea unui cont în aplicația web, în timp ce aplicația mobilă poate fi accesată doar de tenismeni.

The ATP Tour web application developed in this paper using Oracle APEX manages the most important aspects of the men's tennis circuit, such as player performances, competition calendar and tournament details, match schedule, results and ticket sales. Moreover, the app focuses on user entertainment and interaction through the discussion forum and the annual trivia contest.

Unlike other similar products already existing on the market, the software includes a module dedicated to planning future trips based on the nearest tournaments. In addition, the website is accompanied by the ITIA-App mobile application developed in React Native which helps with the organization of players' schedules and locations used within the anti-doping program.

Depending on the users' roles, there are five existing access levels: administrators, editors, authenticated users, players and the general public. The first four mentioned require an account created through the web application, while the mobile app may only be used by the athletes.

INTRODUCERE

Contextul lucrării

Obiectivul principal al industriei oricărui sport din ziua de astăzi este de a atrage cât mai mulți fani care să urmărească competițiile atât la televizor, cât și din tribune și să fie implicați în discuții legate de acestea pe rețelele de socializare. Pentru a putea promova cu succes jucătorii și turneele și a oferi utilizatorilor un mediu accesibil de regăsire a tuturor informațiilor relevante în același loc, astfel de organizații au nevoie de aplicații web eficiente și ușor de utilizat.

Motivație

Tenisul se află printre cele mai populare sporturi din lume din punct de vedere al numărului de fani și practicanți. Având cel mai încărcat sezon competițional cu peste șaizeci de turnee de nivel înalt, tenisul atrage în fiecare lună zeci sau chiar sute de mii de spectatori și milioane de telespectatori.

Pasiunea pentru tenis și ultimii zece ani în care am urmărit îndeaproape desfășurarea întrecerilor profesioniste și evoluția jucătorilor m-au ajutat în cunoașterea în detaliu atât a structurii circuitului, cât și a informațiilor biografice și tehnice ale tenismenilor. Aceste aspecte m-au determinat să realizez o aplicație web ce ajută în gestiunea și furnizarea tuturor caracteristicilor importante legate de circuitul de tenis ATP.

Principalele elemente incluse în aplicație sunt: jucătorii alături de sponsorii și antrenorii acestora, programul și rezultatele meciurilor din cadrul turneeelor și vânzarea biletelor. În plus, se dorește facilitarea unei metode de găsire rapidă a competițiilor pe baza diverselor proprietăți ale acestora și a distanței față de locațiile utilizatorilor pentru planificarea viitoarelor participări la meciuri.

De asemenea, este rezolvată problema organizării detaliilor de localizare a sportivilor în cadrul programului anti-doping cu ajutorul unei aplicații mobile ce funcționează atât pe dispozitivele Android, cât și iOS. În cadrul acesteia jucătorii își vor putea seta zilnic adresa și intervalul orar în care sunt disponibili pentru testare.

Structura lucrării

Capitolul 1 cuprinde informații despre tipul bazei de date utilizate în lucrare, versiunea acesteia și tipurile de date geospațiale disponibile.

În cadrul capitolului 2 sunt descrise tehnologiile folosite pentru realizarea celor două aplicații, punându-se accent pe principalele funcționalități și pe comunicarea dintre platformele

destinate dezvoltării web și cele destinate programării dispozitivelor mobile. Sunt prezentate componentele și acțiunile oferite de Oracle Application Express și modul de combinare al limbajului Typescript cu tehnologiile React Native și Expo pentru crearea produselor software *cross-platform*.

Capitolul 3 include formularea și modelarea subiectului lucrării prin intermediul transpunerii regulilor de *business* în diagramele bazei de date.

În capitolul 4 este realizată o trecere completă prin aplicația web a circuitului de tenis masculin pentru a ilustra toate paginile și funcționalitățile implementate. Acestea sunt, de asemenea, prezentate cu ajutorul diagramelor UML.

Capitolul 5 descrie modalitățile de dezvoltare și utilizare a aplicației mobile pentru actualizarea profilului de doping al tenismenilor. Sunt incluse ecranele aplicației și metodele în care este accesat serviciul RESTful creat în APEX și API-ul Places din Google Cloud.

În ultimul capitol sunt prezentate concluziile lucrării și caracteristicile ce pot fi implementate în versiunile viitoare ale aplicațiilor.

I BAZE DE DATE

Bazele de date reprezintă componenta principală a administrării informației din majoritatea sistemelor digitale din zilele noastre, permitând stocarea, prelucrarea și manipularea eficientă a datelor. Fie că este vorba de aplicații bancare, rețele de socializare sau platforme de gestiune a înregistrărilor medicale dintr-un spital, bazele de date oferă un mediu sigur și performant pentru depozitarea și accesarea unui volum din ce în ce mai mare de informații.

Înainte de dezvoltarea produselor software, este important ca persoanele responsabile de proiectarea logică a bazei de date să cunoască în detaliu structura și modul în care operează domeniul modelat pentru o implementare optimă și veridică a schemei. De asemenea, analiza funcționalităților aplicației și anticiparea manierei în care utilizatorii vor interacționa cu aceasta sunt utile în alegerea corectă a tipului de bază de date (relațională, non-relațională, orientată obiect, graf, multidimensională, etc), a sistemului de gestiune a bazei de date (Oracle Database, MySQL, Microsoft SQL Server, MongoDB, Cassandra, etc.) și a tipurilor de date sub care vor fi stocate informațiile.

I.1 Baze de date relaționale

O bază de date relațională este un tip de bază de date care stochează și oferă acces la date corelate între ele prin intermediul relațiilor. În astfel de sisteme toate informațiile sunt reținute în tabele și pot fi identificate în mod unic prin numele entității, valoarea cheii primare și denumirea atributelor. [3]

Fundamentele bazelor de date relaționale au fost puse de Edgar F. Codd în 1970 prin definirea modelului relațional ce permite crearea de legături între oricare două tabele prin folosirea unor attribute comune. Contrafolosirii unei structuri ierarhice de organizare a datelor, acest model propune o alternativă în care datele sunt stocate, accesate și conectate într-un mod ce evită necesitatea unei reorganizări ulterioare a arhitecturii. [17]

I.1.1 Modelul relațional

În anii de început ai bazelor de date, aplicațiile aveau propriile structuri de stocare a datelor, fapt ce reprezenta un impediment pentru dezvoltatorii care trebuiau să cunoască în amănunt fiecare nou sistem ca să le poată accesa. Pentru a rezolva problema numeroaselor organizări arbitrară și a

optimiza metodele de întreținere și prelucrare a informațiilor, a fost introdus modelul relațional, un standard intuitiv și simplu de reprezentare a datelor. [16]

Principalul concept ce stă la baza modelului relațional îl constituie tabelele. Acestea sunt alcătuite din rânduri ce reprezintă înregistrări/tupluri și coloane ce reprezintă câmpuri/attribute. Ideea de normalizare ce reduce problema redundanței din date împreună cu operatorii algebrei relaționale (selectie, proiecție, join, operatori pe mulțimi) oferă un mijloc eficient și flexibil de stocare și prelucrare a datelor structurate. [3]

În timp, un alt punct forte al utilizării modelului relațional a apărut odată cu dezvoltarea limbajului *SQL* (*Structured Query Language*) pentru manipularea și interogarea datelor. Bazat pe algebra relațională, acesta pune la dispoziție o sintaxă declarativă pentru accesarea și gestiunea performantă a informațiilor dintr-o bază de date relațională. [17]

I.1.2 Structura bazelor de date relaționale

Modelul relațional oferă o separare între structura logică a datelor (tabele, vizualizări materializate, indecsi, etc.) și nivelul fizic de stocare. Această distincție pune la dispoziție administratorilor de baze de date o metodă accesibilă de gestiune a stocării fizice a datelor fără a afecta disponibilitatea acestora. [16]

Pentru ca accesibilitatea și corectitudinea datelor să fie asigurată în permanență, bazele de date relaționale urmează anumite reguli de integritate. Principalele tipuri de integritate ce împiedică eventuala introducere a informațiilor eronate sunt:

- **integritatea entității** – implementată prin intermediul cheii primare ce identifică înregistrările dintr-un tabel în mod unic;
- **integritatea referențială** – implementată prin intermediul cheilor externe ce creează legături între tabele și facilitează anumite practici de inserare și ștergere a datelor ce asigură corespondența între datele celor două tabele;
- **integritatea domeniului** – se referă la limitele sau regulile pe care trebuie să le respecte valorile unei coloane;
- **integritatea definită de utilizator** – descrie seturile de reguli specifice conceptelor modelate și cerințelor de business.

I.1.3 Consistența datelor în modelul relațional

Există patru proprietăți importante ce definesc tranzacțiile din bazele de date relaționale, acestea fiind, de obicei, referite prin intermediul acronimului ACID: [17]

- **Atomicitate** – toate modificările datelor se realizează ca fiind o singură unitate (fie toate operațiile dintr-o tranzacție sunt efectuate, fie niciuna);
- **Consistență** – informațiile stocate respectă întotdeauna regulile și constrângările impuse (dacă o tranzacție determină date invalide, atunci baza de date revine la starea anterioară ce conține date coerente);
- **Izolare** – asigură faptul că stările intermediare din cadrul unei tranzacții nu sunt vizibile din cadrul altora, astfel încât tranzacțiile ce rulează concurent par a fi serialized;
- **Durabilitate** – odată ce tranzacția este permanentizată, modificările realizate devin persistante și nu pot fi anulate nici măcar în cazul unei defecțiuni a sistemului.

Modelul relațional este cel mai performant în menținerea coereneței datelor între aplicații și diferite instanțe ale bazei de date. Alte tipuri de sisteme, precum cele *NoSQL*, întâmpină dificultăți în asigurarea unei consistențe imediate atunci când se lucrează cu o cantitate mare de date. Astfel, acestea sunt nevoite să recurgă la un principiu numit coerentă eventuală (*"eventual consistency"*) ce presupune scurgerea unei perioade de timp până când toți utilizatorii ce accesează simultan baza de date să poată vedea conținutul actualizat al acesteia. Cu toate că abordarea menționată poate funcționa fără probleme majore în cadrul anumitor cazuri de utilizare, operațiunile de afaceri, precum tranzacțiile din coșul de cumpărături, folosesc ca standard bazele de date relaționale. [16]

I.1.4 Blocarea și concurența în bazele de date

Atunci când mai mulți utilizatori sau aplicații încearcă să modifice anumite date în același timp, pot avea loc diverse conflicte. Tehnicile de blocare și concurență reduc potențialul apariției acestora, menținând totodată integritatea datelor.

Blocarea are rolul de a împiedica alți utilizatori să acceseze informații ce sunt în curs de actualizare. În timp ce anumite sisteme recurg la procedeul de blocare a întregului tabel, ceea ce creează un impact negativ asupra performanței, alte baze de date relaționale, precum Oracle Database, aplică blocări la nivel de înregistrare, lăsând disponibile celelalte linii ale tabelului ce nu sunt afectate de tranzacțiile curente.

Concurența gestionează activitatea atunci când utilizatori mulți invocă simultan interogări asupra bazei de date. Această capacitate oferă în mod corect accesul la informații în conformitate cu politicile definite pentru controlul datelor. [16]

I.1.5 Avantajele bazelor de date relaționale

Principalul beneficiu adus de tratarea relațională a proiectării și implementării bazei de date îl constituie posibilitatea creării unor asocieri între date prin intermediul legăturilor dintre tabele. De asemenea, capacitatea de a realiza rapoarte complexe cu ajutorul funcțiilor de agregare, grupare, filtrare și ordonare puse la dispoziție de limbajul *SQL* face din abordarea relațională cel mai des utilizat instrument de interogare în afacerile din zilele noastre.

Față de celelalte tipuri de baze de date, modelul relațional prezintă următoarele avantaje: ușurința utilizării, reducerea redundanței și facilitatea backup-ului și a recuperării datelor.

Ușurința utilizării

Datorită istoriei îndelungate a bazelor de date relaționale, există o comunitate dezvoltată în jurul acestora și nenumărate resurse ce facilitează implementarea unei astfel de soluții pentru organizații de diferite dimensiuni. Cu toate că din punct de vedere istoric modelul relațional a fost privit ca fiind mai rigid din cauza schemei impuse, opțiunile *DBaaS* (*Database as a Service*) și migrarea către mediile *cloud* apropie din ce în ce mai mult flexibilitatea bazelor de date relaționale de cea a bazelor de date de tip *NoSQL*.

Reducerea redundanței

Bazele de date relaționale pot elimina redundanța în două moduri. Modelul relațional în sine minimizează redundanța din date prin aplicarea pașilor de normalizare ce asigură faptul că informațiile nu sunt duplicate în mod inutil. O altă metodă de reducere a proceselor repetitive este folosirea subprogramelor stocate, utile în gestionarea controlului accesului și în eficientizarea muncii dezvoltatorilor prin furnizarea unor secvențe de cod reutilizabile.

Facilitatea backup-ului și recuperării în caz de dezastru

Datorită naturii tranzacționale a bazelor de date relaționale, acestea garantează faptul că starea întregului sistem este consistentă în orice moment. Opțiunile de export și import al datelor, inclusiv în timp ce baza de date rulează, ușurează restaurarea în caz de eșec. Bazele de date relaționale moderne puse la dispoziție de furnizorii serviciilor de *cloud* oferă oglindire continuă, ceea ce reduce la ordinul secundelor intervalul de timp în care datele pot fi inaccesibile din cauza

unei deficiențe. De asemenea, există posibilitatea creării unor replici ce permit doar citirea datelor, acestea putând fi promovate la instanțe de citire/scriere în cazul unui dezastru. [17]

I.1.6 Baza de date relațională a viitorului

De-a lungul timpului bazele de date au devenit din ce în ce mai bune, puternice și eficiente. Cu toate acestea, complexitatea lor a ajuns la nivelul la care administrarea s-a transformat într-o muncă îndelungată și costisitoare.

În ziua de astăzi, tehnologia autonomă bazată pe punctele forte ale modelului relațional, infrastructura *cloud* și învățarea automată oferă un nou tip de baze de date relaționale. Baza de date autonomă menține avantajele modelului relațional, însă utilizează inteligență artificială și automatizarea pentru a monitoriza și îmbunătăți performanța interogărilor și sarcinile de gestionare. Spre exemplu, baza de date autonomă poate formula ipoteze și testa îndești pe cont propriu pentru a crește rapiditatea interogărilor și a le trimite pe cele mai eficiente în producție.

Tehnologia autonomă scutește dezvoltatorii de sarcinile uzuale de administrare, accelerând astfel procesul de dezvoltare a produselor software. De asemenea, cu ajutorul unei baze de date autonome cerințele de infrastructură, resursele de calcul și stocare pot fi extinse sau reduse cu ușurință în funcție de nevoile companiei. [16]

I.2 Baza de date Oracle 19c

Oracle Database 19c reprezintă rezultatul îndelungatului progres tehnologic în domeniul sistemelor de gestiune a bazelor de date din industrie. Lansat în anul 2019 de compania Oracle, acest produs constituie actuala versiune pe termen lung și oferă cel mai înalt nivel de stabilitate și fiabilitate de pe piață.

Oracle 19c aduce o multitudine de caracteristici și îmbunătățiri menite să asigure organizațiilor avantajul unor standarde ridicate atunci când vine vorba de securitate și automatizare. De asemenea, arhitectura bazei de date este concepută pentru a se integra perfect cu mediile *cloud*, permitând dezvoltatorilor să valorifice puterea implementărilor hibride și *multi-cloud* pentru flexibilitate și scalabilitate maximă. [10]

Este recomandă folosirea acestei versiuni pentru a beneficia de cele mai noi actualizări și îmbunătățiri de performanță. La momentul actual data încheierii suportului pentru Oracle 19c este stabilită la 30 aprilie 2027.

I.2.1 Istoria bazelor de date Oracle

În anul 1977 Larry Ellison, Bob Miner și Ed Oates au fondat compania de consultanță Software Development Laboratories, care a devenit Relational Software, Inc. (RSI). În 1983 RSI și-a schimbat numele în Oracle Systems Corporation și mai apoi în Oracle Corporation.

Lansarea în 1979 a Oracle V2 (Version 2), prima bază de date bazată pe SQL disponibilă în comerț, a marcat un moment de cotitură în istoria bazelor de date relaționale. Oracle Version 3 a reprezentat prima bază de date relațională ce rulează pe *mainframe*, minicalculatoare și calculatoare personale. Aceasta era scrisă în C, ceea ce oferea portabilitate pe diverse platforme.

Oracle 4 a introdus consistența la citire prin reținerea de versiuni multiple ale datelor pe parcursul tranzacțiilor, în timp ce versiunea 5 a venit cu posibilitatea efectuării computațiilor de tip client-server și metode de distribuire a bazei de date. Oracle 6 a adus îmbunătățiri în procesele de citire/scriere pe disk, blocarea la nivel de înregistrare și opțiuni de backup și recuperare. În Oracle 7 au fost introduse funcționalități noi ale limbajului PL/SQL, precum subprograme stocate și *trigger-i*. Versiunea 8 a bazei de date Oracle punea la dispoziție o multitudine de noi tipuri de date, precum obiectele, și posibilitatea partaționării tabelelor de mari dimensiuni.

Odată cu punerea în circulație a Oracle 8i în 1999 ce oferea suport nativ pentru protocolele de internet, implementarea bazelor de date putea fi efectuată într-un mediu pe mai multe niveluri. În continuare, versiunea Oracle 9i a introdus *Oracle RAC* (*Real Application Clusters*) ce permitea mai multor instanțe să acceseze o singură bază de date simultan.

În 2003, odată cu lansarea bazei de date Oracle 10g, a fost dezvoltat conceptul de *grid computing* care asigura organizațiilor mijloace de virtualizare a resurselor de calcul prin construirea unei infrastructuri de tip grid bazată pe servere ce au costuri reduse. Oracle Database 11g a introdus o serie de caracteristici noi ce au facilitat adaptarea rapidă a administratorilor și programatorilor la cerințele în continuă schimbare ale afacerilor. Simplificarea infrastructurii sistemului informatic prin frecventa utilizare a automatizării a reprezentat un punct cheie în conceperea acestei versiuni.

În 2013 a fost pusă în circulație prima bază de date proiectată pentru *cloud*, Oracle 12c. Aceasta oferea o nouă arhitectură *multi-tenant* bazată pe containere și suport pentru documentele de tip JSON. Oracle Database 18c a simplificat integrarea cu servicii de directoare, precum *Microsoft Active Directory*, și a adus funcționalități de exploatare a memoriei pentru un acces foarte rapid la înregistrări. Obiectivele actualei versiuni pe termen lung, Oracle 19c, sunt consolidarea stabilității și îmbunătățirea serviciului de replicare prin *Active Data Guard*. [10]

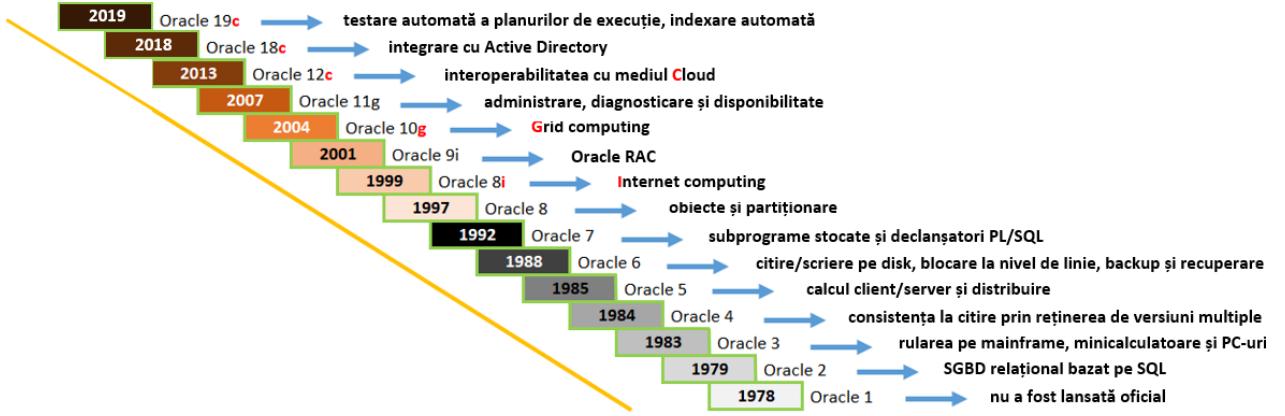


Figura I.1 – Cronologia versiunilor Oracle

I.2.2 Caracteristici noi în Oracle 19c

Oracle Database 19c face parte din familia de produse ce conține versiunile 12c și 18c datorită accentului pus pe dezvoltarea în mediul *cloud* și infrastructura *Oracle Cloud*. Acest produs vine cu îmbunătățiri substanțiale față de versiunile precedente, oferind caracteristici noi, de la suport pentru dezvoltarea aplicațiilor bazate pe date, până la învățare automată și o mai bună agilitate folosind arhitectura containerizată de tip *multi-tenant*.

Oracle 19c poate fi implementat pe *Oracle Autonomous Database*, *Oracle Cloud Infrastructure*, *Oracle Exadata*, *Oracle Database Appliance* și alte medii locale. [4]

În continuare vom ilustra câteva dintre principalele funcționalități noi pe care le pune la dispoziție Oracle Database 19c și care ajută organizațiile în utilizarea unei baze de date sigure și eficiente.

Indexare automată

Indecșii bazelor de date sunt utilizati pentru a mări viteza de interogare și preluare a informațiilor în schimbul creșterii numărului de operații de scriere și a spațiului de stocare.

În Oracle 19c există capacitatea de a efectua indexare automată prin intermediul pachetului DBMS_AUTO_INDEX ce va genera indecsi candidat pentru tabelele ce ar putea beneficia de aceștia. Oracle Database va testa un potențial index pentru a se asigura că performanța interogărilor este într-adevăr îmbunătățită, caz în care acesta va fi definit în aplicații. În caz contrar, el nu va fi vizibil pentru utilizatorii finali și va fi în cele din urmă eliminat. [9]

Spre exemplu, pentru a configura și porni indexarea automată putem folosi comanda:

```
exec dbms_auto_index.configure('AUTO_INDEX_MODE', 'IMPLEMENT');
```

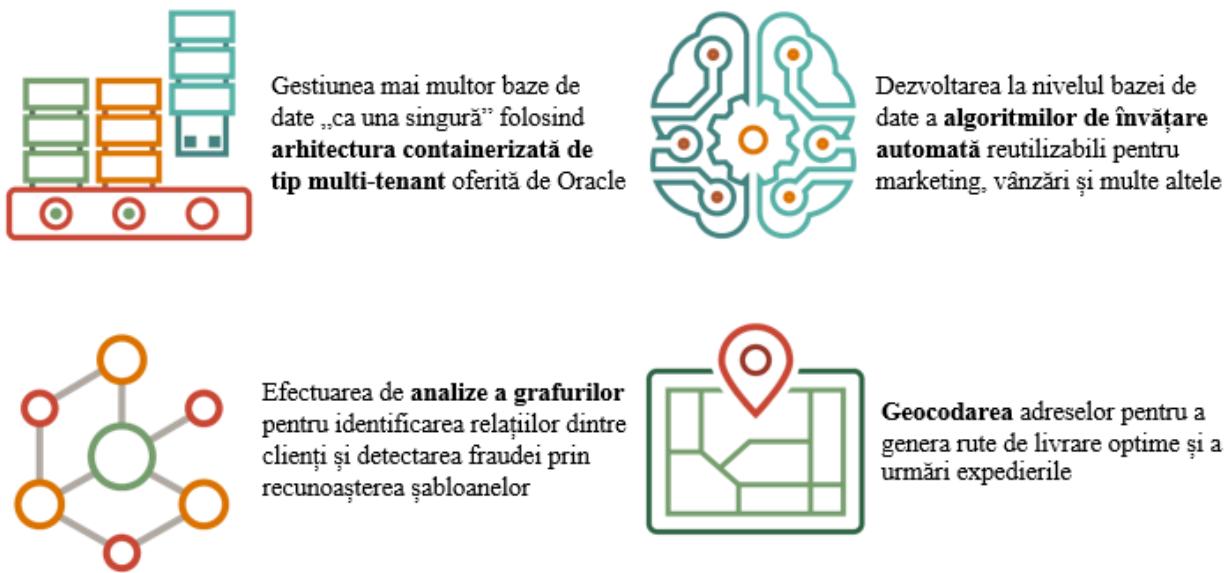


Figura I.2 – Capabilitățile de utilizare a bazei de date Oracle 19c

Îmbunătățiri *Oracle Data Guard*

Oracle Data Guard este un program ce îi ajută pe utilizatori să mențină replici secundare „în aşteptare” ale bazelor de date principale din producție. Deși *Data Guard* constituie de mult timp o caracteristică a sistemului de gestiune a bazelor de date Oracle, versiunea 19c aduce numeroase îmbunătățiri precum:

- schimbarea dinamică a bazei de date „standby” în procesul de *Fast-Start Failover*, versiunile anterioare necesitând dezactivarea acestuia înainte de modificarea replicii;
- gestionarea mai ușoară a parametrilor bazei de date prin intermediul utilitarului Oracle SQL*Plus;
- suport pentru redirecționarea limbajului de manipulare a datelor (LMD) în *Active Data Guard*. [9]

„Carantine” SQL

Dacă o cerere SQL consumă prea multe resurse CPU și I/O, *Oracle Database Resource Manager* o va opri automat, prevenind astfel scăderea eficienței și a vitezei bazei de date. În versiunile anterioare, însă, nu exista o metodă ce împiedica utilizatorii să execute ulterior aceeași interogare, cauzând din nou probleme de performanță.

În Oracle 19c pachetul DBMS_SQLQ oferă posibilitatea setării unor praguri de consum al resurselor ce vor determina comenzi SQL costisitoare să fie puse în „carantină”, fapt ce împiedică execuția lor repetată. [9]

Partiționare hibridă

Partiționarea externă a tabelelor a fost introdusă în versiunea 12.2 și presupune folosirea unui driver extern (ORACLE_LOADER sau ORACLE_DATAPUMP) pentru a obține datele dintr-un fișier sau dintr-o sursă *cloud* (DBMS_CLOUD).

Oracle 19c pune la dispoziție opțiunea tabelelor partiționate hibrid ce conțin atât partiții externe cât și normale. Scopul principal al acestei metode de partiționare este ca partițiiile utilizate frecvent să rămână la nivelul bazei de date, în timp ce partițiiile accesate rar să fie externalizate către sisteme mai ieftine de stocare.

Cele două versiuni apărute după Oracle Database 19c sunt Oracle 21c și Oracle 23c. Prima dintre acestea a fost menită să prezinte potențialele inovații din bazele de date ulterioare, nebeneficind de stabilitatea versiunilor pe termen lung. Oracle 23c reprezintă noua versiune pe termen lung, însă timpul scurt de la lansare și imposibilitatea rulării pe sistemul de operare Windows fără utilizarea mașinilor virtuale sau a containerelor Docker constituie dezavantaje în fața fiabilității, flexibilității și îndelungatei testări a bazei de date Oracle 19c.

I.3 Tipuri de date geospațiale în Oracle

Oracle *Spatial and Graph* reprezintă o suită de caracteristici avansate pentru lucrul cu date geospațiale și aplicații ce descriu structuri fizice, logice sau sociale sub forma grafurilor. Funcționalitățile spațiale oferă o schemă specială și metode ce facilitează stocarea, actualizarea și interogarea aspectelor geospațiale dintr-o bază de date Oracle. *Spatial and Graph* este alcătuit din următoarele componente:

- schema MDSYS care prevede stocarea, sintaxa și semantica tipurilor de date geometrice acceptate;
- un mecanism de indexare spațială;
- operatori, funcții și proceduri pentru efectuarea interogărilor ce conțin analize spațiale legate de zonele de interes;
- un model pentru lucrul cu date ce descriu nodurile, muchiile și fețele dintr-o topologie;
- un model de date de tip rețea pentru reprezentarea obiectelor descrise de nodurile și muchiile unui graf. [13]

I.3.1 Elemente geometrice

O geometrie reprezintă o succesiune ordonată de vârfuri conectate prin segmente de linie dreaptă sau arce circulare. Semantica unei geometrii este determinată de tipul acesteia, categoriile bidimensionale primitive și compuse acceptate fiind:

- puncte și grupuri de puncte;
- șiruri de linii;
- poligoane;
- șiruri de arce;
- poligoane de arce;
- poligoane compuse;
- șiruri de linii compuse;
- cercuri;
- dreptunghiuri optimizate. [8]

Punctele bidimensionale sunt alcătuite din două coordonate, X și Y , corespunzând adesea longitudinii și latitudinii dintr-un sistem de coordonate. Șirurile de linii sunt compuse din una sau mai multe perechi de puncte ce definesc segmente de dreaptă. Poligoanele sunt formate din șiruri de linii conectate ce formează o figură închisă, nefiind posibile auto intersecții sau auto suprapunerile. De exemplu, un punct poate reprezenta locația unei clădiri, un șir de linii poate corespunde unui drum sau unei rute de zbor, iar un poligon poate ilustra un stat, oraș, sector sau complex rezidențial. [13]

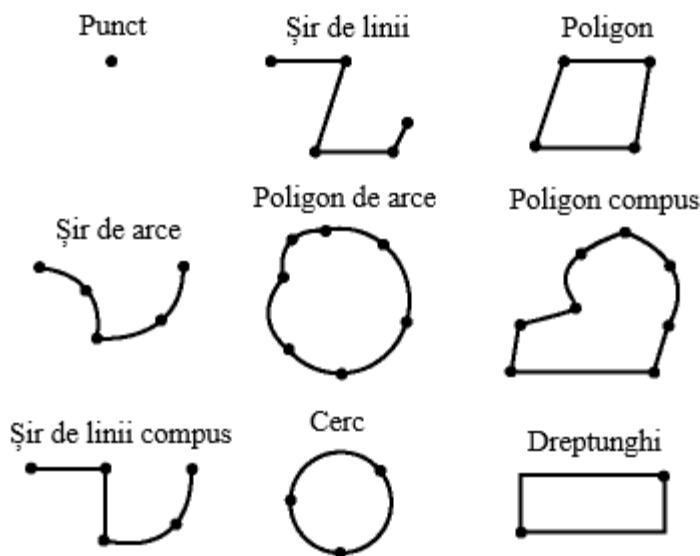


Figura I.3 – Tipuri geometrice

Sistemul Oracle acceptă, de asemenea, stocarea și indexarea tipurilor geometrice tridimensionale sau cu patru dimensiuni, unde trei, respectiv patru coordonate sunt utilizate pentru a descrie fiecare punct al obiectului definit.

I.3.2 Tipul de obiect SDO_GEOmetry

Spatial and Graph folosește modelul obiect-relațional pentru reprezentarea datelor spațiale, stocând figurile geometrice în tipul nativ Oracle pentru date vectoriale, SDO_GEOmetry. Un tabel poate conține una sau mai multe coloane de tip SDO_GEOmetry care este definit astfel:

```
CREATE TYPE sdo_geometry AS OBJECT (
    SDO_GTYPE NUMBER,
    SDO_SRID NUMBER,
    SDO_POINT SDO_POINT_TYPE,
    SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
    SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Tipurile SDO_POINT_TYPE, SDO_ELEM_INFO_ARRAY și SDO_ORDINATE_ARRAY au fost create în următorul mod:

```
CREATE TYPE sdo_point_type AS OBJECT (
    X NUMBER,
    Y NUMBER,
    Z NUMBER);
CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
CREATE TYPE sdo_coordinate_array AS VARRAY (1048576) of NUMBER;
```

SDO_GTYPE

Atributul SDO_GTYPE determină tipul geometriei corespunzător celor din lista obiectelor geometrice valide ale specificației *OGIS (Open Geodata Interoperability Specification)* pentru SQL. Valoarea sa este alcătuită din 4 cifre în formatul DLTT unde:

- D descrie numărul de dimensiuni;
- L identifică dimensiunea măsurii pentru un obiect tridimensional de tip *LRS (Linear Referencing System)*, mai exact, ce dimensiune (3 sau 4) conține valoarea măsurii (pentru obiectele geometrice ce nu sunt de tip LRS se specifică 0);
- TT reprezintă tipul geometric (poate lua valori de la 01 la 09, intervalul 10-99 fiind rezervat pentru utilizări viitoare). [14]

Semnificațiile identificatorilor TT sunt descrise în tabelul de mai jos.

Valoare	Tip geometric OGIS	Descriere
<i>DL00</i>	UNKNOWN_GEOMETRY	<i>Spatial and Graph</i> ignoră acest tip
<i>DL01</i>	POINT	Geometria conține un singur punct
<i>DL02</i>	LINE sau CURVE	Geometria conține un sir de linii compus din segmente, arce de cerc sau ambele
<i>DL03</i>	POLYGON sau SURFCAE	Geometria conține un poligon cu sau fără spații goale sau o suprafață compusă din unul sau mai multe poligoane
<i>DL04</i>	COLLECTION	Geometria este o colecție eterogenă de elemente, fiind un superset ce include toate celelalte tipuri
<i>DL05</i>	MULTIPOINT	Geometria conține unul sau mai multe puncte
<i>DL06</i>	MULTILINE sau MULTICURVE	Geometria conține unul sau mai multe șiruri de linii
<i>DL07</i>	MULTIPOLYGON sau MULTISURFACE	Geometria poate conține mai multe poligoane sau suprafețe disjuncte
<i>DL08</i>	SOLID	Geometria constă în mai multe suprafețe și este complet închisă într-un spațiu tridimensional
<i>DL09</i>	MULTISOLID	Geometria poate avea mai multe corpuri disjuncte

Tabel I.1 – Valorile identificatorului TT din atributul SDO_GTYPE

De exemplu, o valoare a atributului SDO_GTYPE de 2003 reprezintă un poligon bidimensional (cifra 2 indică 2 dimensiuni, iar valoarea 3 indică forma poligonală). Orice coloană de tip SDO_GEOMETRY trebuie să conțină același număr de dimensiuni pentru toate elementele.

SDO_SRID

Atributul SDO_SRID poate fi utilizat pentru a asocia un sistem de coordonate cu obiectele geometrice stocate în baza de date. Dacă valoarea este nenulă, aceasta trebuie să corespundă uneia dintre opțiunile din coloana SRID a tabelului SDO_COORD_REF_SYS și să fie inserată în coloana SRID a vizualizării USER_SDO_GEOM_METADATA. [14] Valoarea SRID pentru coordonatele WGS84/GPS ce descriu latitudinea și longitudinea de pe suprafața Pământului este 4326.

Toate obiectele dintr-o coloană geometrică trebuie să aibă același SDO_SRID dacă se dorește crearea unui index spațial pe aceasta.

SDO_POINT

Proprietatea SDO_POINT este descrisă folosind attributele *X*, *Y* și *Z* ale tipului SDO_POINT_TYPE. Dacă vectorii SDO_ELEM_INFO și SDO_ORDINATES sunt ambii nuli,

iar atributul SDO_POINT este nenul, atunci valorile X , Y și Z sunt considerate a fi coordonatele pentru un element geometric de tip punct. În caz contrar, *Spatial and Graph* ignoră proprietatea SDO_POINT.

Este recomandat ca toate punctele să fie stocate în acest atribut pentru o performanță optimă a scrierii, actualizării și interogării datelor. [14]

SDO_ELEM_INFO și SDO_ORDINATES

Celealte forme geometrice pot fi stocate în atributele de tip vector SDO_ELEM_INFO și SDO_ORDINATES.

În SDO_ORDINATES sunt reținute coordonatele tuturor elementelor ce definesc conturul obiectului spațial.

Atributul SDO_ELEM_INFO specifică unde în vectorul SDO_ORDINATES începe un nou element (SDO_STARTING_OFFSET), cum este conectat de următorul (prin linii drepte sau arce) (SDO_INTERPRETATION) și dacă este punct, linie sau poligon (SDOETYPE). De exemplu, o geometrie cu SDO_ELEM_INFO egal cu (1, 1003, 3) și SDO_ORDINATES egal cu (1, 1, 5, 7) reprezintă un poligon interior (coordonatele sunt enumerate în sensul invers al celor de ceasornic) conform valorii 1003. [8] Datorită cifrei 3 din primul vector, poligonul are formă dreptunghiulară, fiind suficient să fie specificate doar coordonatele colțurilor din stânga-jos și dreapta-sus ce vor fi enumerate în SDO_ORDINATES începând cu prima poziție. Cel de-al doilea vector descrie coordonatele celor două vârfuri ca fiind (1, 1) și (5, 7) pentru stânga-jos, respectiv dreapta-sus.

În aplicația din această lucrare a fost nevoie de stocarea coordonatelor geografice (latitudine și longitudine) ale diverselor locații în care se află jucătorii sau complexurile sportive unde au loc turnee de tenis. În tabelul *LOCATIE* avem o coloană *poziție* de tip SDO_GEOmetry ce reține elemente geometrice de tip punct:

```
CREATE TABLE locatie (
    id_locatie NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    descriere VARCHAR2(200) NOT NULL,
    pozitie SDO_GEOmetry NOT NULL,
    id_oras NUMBER REFERENCES oras(id_oras) ON DELETE CASCADE
);
```

Un exemplu de inserare a unei linii în acest tabel este:

```
INSERT INTO locatie(descriere, pozitie, id_oras)
VALUES ('All England Lawn Tennis and Croquet Club',
        sdo_geometry(2001, 4326, sdo_point_type(-0.2144, 51.4343, NULL),
        NULL, NULL), 3);
```

I.3.3 Sistem de coordonate

Un sistem de coordonate (numit și sistem de referință spațială) reprezintă un mijloc de atribuire a coordonatelor unei locații și de stabilire a relațiilor între ele. Orice date spațiale au un sistem de coordonate ce poate fi sau nu georeferențiat (legat de o reprezentare specifică a globului). Dacă sistemul este georeferențiat, acesta are o unitate de măsură implicită (precum metri), putând seta *Spatial and Graph* să returneze automat rezultatele într-o altă unitate specificată, cum ar fi mile.

Categoriile de sisteme de coordonate sunt:

- coordonate carteziene – măsoară poziția unui punct de la o origine definită de-a lungul axelor perpendiculare în spațiul bidimensional sau tridimensional reprezentat (dacă nu este asociat niciun sistem de coordonate, atunci se presupune că acesta este cartezian);
- coordonate geodezice (geografice) – coordonate unghiulare (latitudine și longitudine), strâns legate de coordonatele polare sférice, definite în raport cu o anumită reprezentare a formei Pământului;
- coordonate proiectate – coordonate carteziene plane rezultate din efectuarea unei mapări matematice de la un punct de pe suprafața Pământului la un plan;
- coordonate locale – coordonate carteziene într-un sistem negeoreferențiat, folosite adesea în aplicații de proiectare și sondaje locale. [13]

I.3.4 Toleranță

Toleranța este folosită pentru a asocia un nivel de precizie datelor spațiale, reflectând distanța care poate exista între două puncte astfel încât acestea să fie considerate identice. Valoarea toleranței trebuie să fie un număr strict pozitiv, semnificația acesteia variind dacă datele sunt sau nu asociate cu un sistem de coordonate geodezice.

Pentru datele geodezice, valoarea toleranței reprezintă numărul de metri dintre două puncte (se recomandă folosirea unei valori mai mici ca 10), în timp ce pentru datele non-geodezice descrie numărul de unități asociate cu sistemul de coordonate definit. În ambele cazuri, cu cât toleranța este mai mică, cu atât precizia coordonatelor este mai mare.

Cele două cazuri în care toleranța este utilizată sunt:

- definiția metadatelor geometriei pentru un strat;
- parametrii anumitor funcții. [13]

Vizualizări pentru metadatele unei geometrii

Metadatele geometriei ce descriu dimensiunile, limitele inferioare și superioare și toleranța sunt stocate într-un tabel global detinut de MDSYS. Fiecare utilizator are la dispoziție în schema sa vizualizarea USER_SDO_GEOM_METADATA în care trebuie inserate pentru fiecare coloană de tip SDO_GEOGRAPHY numele acesteia, numele tabelului din care face parte, SRID-ul asociat sistemului de coordonate folosit și informații despre fiecare dimensiune. Tipul de obiect SDO_DIM_ELEMENT conține attributele SDO_DIMNAME (numele dimensiunii), SDO_LB (limita inferioară), SDO_UB (limita superioară) și SDO_TOLERANCE (toleranța) care definesc caracteristicile dimensiunilor. [14]

În exemplul de mai sus care ilustră coloana *poziție* a tabelului *LOCATIE* ce stocă latitudinea și longitudinea, limitele sunt [-180, 180] pentru dimensiunea X/longitude și [-90, 90] pentru dimensiunea Y/latitudine, înregistrarea inserată în vizualizare arătând astfel:

```
INSERT INTO user_sdo_geom_metadata(table_name, column_name, diminfo,
                                     srid)
VALUES ('locatie', 'pozitie', SDO_DIM_ARRAY(SDO_DIM_ELEMENT('X', -180,
                                                               180, 0.005), SDO_DIM_ELEMENT('Y', -90, 90, 0.005)), 4326);
```

Oracle Spatial and Graph asigură faptul că vizualizarea ALL_SDO_GEOM_METADATA ce conține informații despre toate coloanele geospațiale reflectă modificările efectuate asupra vizualizării USER_SDO_GEOM_METADATA.

Parametrii de toleranță

Multe funcții spațiale acceptă un parametru de toleranță care, dacă este specificat, suprascrie valoarea implicită a toleranței pentru stratul setat în vizualizarea USER_SDO_GEOM_METADATA.

Dacă distanța dintre două puncte este mai mică sau egală cu valoarea toleranței, sistemul consideră că cele două puncte sunt unul și același. Astfel, toleranța reflectă nivelul de precizie pe care utilizatorii îl doresc de la date și rezultatele analizelor acestora.

I.3.5 Indexarea datelor spațiale

Integrarea capabilităților de indexare spațială în motorul de baze de date Oracle reprezintă o funcționalitate cheie a produsului *Spatial and Graph*. Ca orice alt index, un index spațial are rolul de a limita căutările dintr-o interogare pentru îmbunătățirea performanței. Mecanismele indecșilor spațiali se bazează pe criterii precum intersecția sau incluziunea. Aceștia pot fi utilizați pentru:

- regăsirea unui obiect dintr-un spațiu indexat ce interacționează cu un anumit punct sau zonă de interes;
- regăsirea unor perechi de date din spații indexate ce au diverse legături între ele.

Indecșii spațiali sunt de tip *R-tree*. Similar cu indecșii de tip *B-tree*, aceștia reprezintă arbori de căutare balansați ce organizează datele în blocuri și sunt destinați stocării pe disk. Fiecare bloc/nod poate conține un număr maxim de elemente, fiind stabilit și un număr minim ce trebuie îndeplinit pentru o performanță optimă. În timp ce indecșii *B-tree* garantează un prag minim de 50%, datorită complexității informațiilor multidimensionale, pragul pentru indecșii *R-tree* este, de obicei, între 30% și 40%. Indecșii *R-tree* pot lucra cu date spațiale până la patru dimensiuni, aproximând fiecare geometrie printr-un sigur dreptunghi numit *MBR* (*Minimum bounding rectangle*) ce înglobează minim obiectul. Pentru un strat/coloană de tip geometric, un index *R-tree* constă într-un index ierarhic asupra *MBR*-urilor tuturor obiectelor. [13]

Indecșii *R-tree* sunt stocați în tabloul *SDO_INDEX_TABLE* al vizualizării *USER_SDO_INDEX_METADATA*, menținându-se, de asemenea, un obiect de tip secvență în *SDO_RTREE_SEQ_NAME* pentru a fi asigurată posibilitatea actualizărilor simultane realizate de către utilizatori diferiți.

Pentru exemplul din aplicație indexul spațial a fost creat cu ajutorul următoarei comenzi:

```
CREATE INDEX locatie_idx ON locatie(pozitie)
INDEXTYPE IS MDSYS.SPATIAL_INDEX_V2;
```

I.3.6 Metode SDO_GEOmetry

Tipul de obiect *SDO_GEOmetry* conține o serie de metode folosite pentru extragerea informațiilor despre elementele geometrice. Principalele metode sunt prezentate în următorul tabel.

Nume	Tip de date returnat	Descriere
Get_Dims	NUMBER	Numărul de dimensiuni ale obiectului, precum este specificat în valoarea atributului SDO_GTYPE
Get_GeJson	CLOB	Reprezentarea GeoJSON a obiectului geometric
Get_GType	NUMBER	Tipul geometric al obiectului, precum este specificat în valoarea atributului SDO_GTYPE
ST_IsValid	NUMBER	Returnează 1 sau 0 dacă definiția obiectului este validă sau nu

Tabel I.2 – Metode SDO_GEOMETRY

Un exemplu pentru afișarea numărului de dimensiuni, tipului și validității locației cu id-ul egal cu 3 este:

```
SELECT l.pozitie.Get_Dims() dimensiuni, l.pozitie.Get_GType() tip,
       l.pozitie.ST_IsValid() valid
  FROM   locatie l
 WHERE  id_locatie = 3;
```

I.3.7 Operatori, funcții și proceduri spațiale

Interfața de programare PL/SQL a aplicațiilor *Spatial and Graph* pune la dispoziție un număr mare de operatori, proceduri și funcții. Operatorii spațiali trebuie specificați în clauza WHERE a cererilor și oferă performanțe optime atunci când folosesc indecsi. Primul parametru al oricărui operator corespunde coloanei SDO_GEOMETRY asupra căreia se face căutarea, iar cel de-al doilea specifică o fereastră de interogare. [15] Principalii operatori spațiali sunt prezenți în tabelul de mai jos.

Operator	Descriere
SDO_FILTER	Specifică ce obiecte geometrice pot interacționa cu o anumită geometrie
SDO_JOIN	Efectuează un join spațial pe baza uneia sau mai multor relații topologice
SDO_NN	Determină cele mai apropiate n obiecte de un element geometric dat
SDO_NN_DISTANCE	Determină distanța unui obiect returnat de operatorul SDO_NN

SDO_POINTINPOLYGON	Preia un set de tupluri ce conțin pe prima coloană coordonata x a unui punct și pe cea de-a doua coloană coordonata y și returnează acele rânduri ce se află în interiorul unui poligon specificat
SDO_RELATE	Determină dacă două geometrii interacționează sau nu într-un anumit mod
SDO_WITHIN_DISTANCE	Determină dacă două obiecte se află la o distanță specificată unul de celălalt

Tabel I.3 – Operatori spațiali

Procedurile și funcțiile spațiale sunt furnizate ca subprograme în pachetele SDO_GEOM, SDO_CS și SDO_LRS. Acestea nu necesită definirea unui index spațial și pot fi utilizate atât în clauza WHERE cât și în subcereri. Cele mai des folosite subprograme sunt SDO_DISTANCE, SDO_AREA și SDO_VOLUME din SDO_GEOM pentru calculul distanței, ariei, respectiv volumului unui obiect geometric.

Un exemplu în care se dorește afișarea numelui, a duratei în zile și a distanței celor mai apropiate 5 turnee de Facultatea de Matematică și Informatică din cadrul Universității București (ale cărei latitudini și longitudini sunt 44.43578061626134 și 26.099565239048783) poate fi:

```

SELECT t.nume, t.data_sfarsit - t.data_inceput + 1 nr_zile,
       ROUND(SDO_NN_DISTANCE(1), 2) distanta
FROM   turneu t, locatie l
WHERE  t.id_locatie = l.id_locatie
AND    SDO_NN(l.pozitie,
              sdo_geometry(2001, 4326,
              sdo_point_type(26.099565239048783, 44.43578061626134,
                             null), null, null),
              'sdo_num_res=5 unit=KM', 1) = 'TRUE'
ORDER BY distanta;

```

II TEHNOLOGII UTILIZATE

În dezvoltarea aplicației web și a aplicației mobile propuse în această lucrare au fost utilizate diverse tehnologii adecvate fiecărei situații, ce comunică între ele pentru realizarea unei soluții eficiente și intuitive pentru utilizator.

II.1 Oracle Application Express

Oracle Application Express (APEX) reprezintă un mediu de dezvoltare dedicat implementării și trimiterii în producție a aplicațiilor web centrate pe baze de date. Motorul APEX are capacitatea de a reda aplicațiile în timp real din datele aflate în baza de date. Atunci când o aplicație este creată sau extinsă, Oracle Application Express adaugă sau modifică metadatele stocate în tabele, citirea ulterioară a acestora făcând posibilă afișarea informațiilor prin intermediul interfeței grafice. De asemenea, APEX oferă un mecanism transparent de gestionare a sesiunilor prin folosirea substituției și a variabilelor de legătură din SQL.

Oracle APEX se instalează împreună cu o versiune *Enterprise* a bazei de date Oracle și constă în date din tabele și cod PL/SQL. Procesul urmat la fiecare afișare sau trimitere a paginilor aplicației este compus din: lansarea din browser a unei solicitări URL ce este transpusă în apelul PL/SQL corespunzător, execuția acestuia și returnarea în *browser* a rezultatelor sub formă de cod HTML. În spate, pe lângă redarea și procesarea paginilor, motorul APEX se ocupă și de următoarele sarcini:

- management-ul sesiunilor;
- servicii de autentificare;
- servicii de autorizare;
- controlul fluxului paginilor;
- procesarea validărilor.

Oracle APEX poate fi instalat local sau utilizat prin intermediul unui serviciu Oracle Cloud precum: Oracle APEX Application Development, baza de date autonomă pentru procesarea tranzacțiilor sau baza de date autonomă pentru analiză și Data Warehousing.

În funcție de gradul de permisiuni și privilegii asupra aplicațiilor realizate în APEX, mediul de dezvoltare instalat poate fi:

- mediu de rulare – utilizatorii pot rula aplicații, dar nu le pot modifica;

- mediu de dezvoltare completă – utilizatorii pot dezvolta, modifica, rula și șterge aplicațiile. [7]

Principalul avantaj al acestei tehnologii constă în accelerarea procesului de dezvoltare prin punerea la dispoziție a componentelor predefinite ce pot fi ușor adaptate la logica și cerințele oricărui tip de aplicație pentru obținerea unor funcționalități utile și a unui aspect vizual plăcut. Spațiul de lucru pentru configurarea paginilor și a legăturilor dintre acestea, secțiunea dedicată implementării și testării codului SQL și PL/SQL și posibilitatea creării și comunicării cu servicii externe reprezintă caracteristici importante ce ajută în realizarea unor aplicații complexe și eficiente.

II.1.1 Page Designer

Page Designer este un mediu de dezvoltare integrat (IDE) ce include o bară de instrumente și mai multe panouri utile pentru definirea și stilizarea diverselor componente ale unei pagini. [7]

În partea centrală a IDE-ului se află zona de *Layout* ce constituie o reprezentare vizuală a poziționării componentelor în pagină. Regiunile, elementele și butoanele pot fi adăugate, mutate, duplicate sau redimensionate în funcție de nevoie, înfățisarea și proporțiile dorite de dezvoltator. Principalele tipuri de regiuni puse la dispoziție de Oracle APEX sunt: conținut static, listă, raport, grafic, formular, calendar și hartă. Există, de asemenea, o serie de tipuri de elemente ce pot fi utilizate în formulare pentru interacțiunea cu utilizatorul, precum: input de tip text, listă de selecție, *date picker*, element ascuns, butoane radio, rating cu steluțe sau încărcare de fisier.

În partea din stânga există o secțiune în care componente din pagină sunt reprezentate în mod ierarhic și este posibilă definirea de acțiuni asociate acestora. La încărcarea paginii pot fi create computații ce populează anumite elemente cu valori, în timp ce procesele sunt folosite pentru execuția unor linii de cod înainte de randarea regiunilor, la apăsarea unui buton sau după trimiterea unei pagini. De asemenea, se pot adăuga acțiuni dinamice ce determină comportamentul unor componente sau rularea unei bucăți de cod în funcție de modul în care utilizatorul interacționează cu un anumit element. O altă caracteristică importantă o reprezintă definirea validărilor ce împiedică introducerea de date eronate ce încalcă regulile de business și constrângerile declarate la nivelul bazei de date. Cu ajutorul acestora poate fi impus formatul datelor de intrare, fiind asociate mesaje de eroare sugestive ce ajută utilizatorii în completarea corectă a formularelor. Nu în ultimul rând, din această secțiune poate fi accesată zona componentelor partajate unde sunt configurate schemele de autentificare și autorizare, meniul aplicației și alte componente globale.

Partea din dreapta a mediului Page Designer este rezervată proprietăților componentelor. Odată cu selectarea unei componente, atributele specifice tipului acesteia vor apărea în editor grupate pe baza funcționalității. Astfel, poate fi introdus codul SQL, PL/SQL sau JavaScript pentru preluarea datelor din baza de date sau definirea comportamentului computațiilor, proceselor și validărilor. De asemenea, poate fi utilizat cod CSS pentru stilizare sau aplicată o schemă de autorizare pentru protejarea componentelor pe baza rolului utilizatorului.

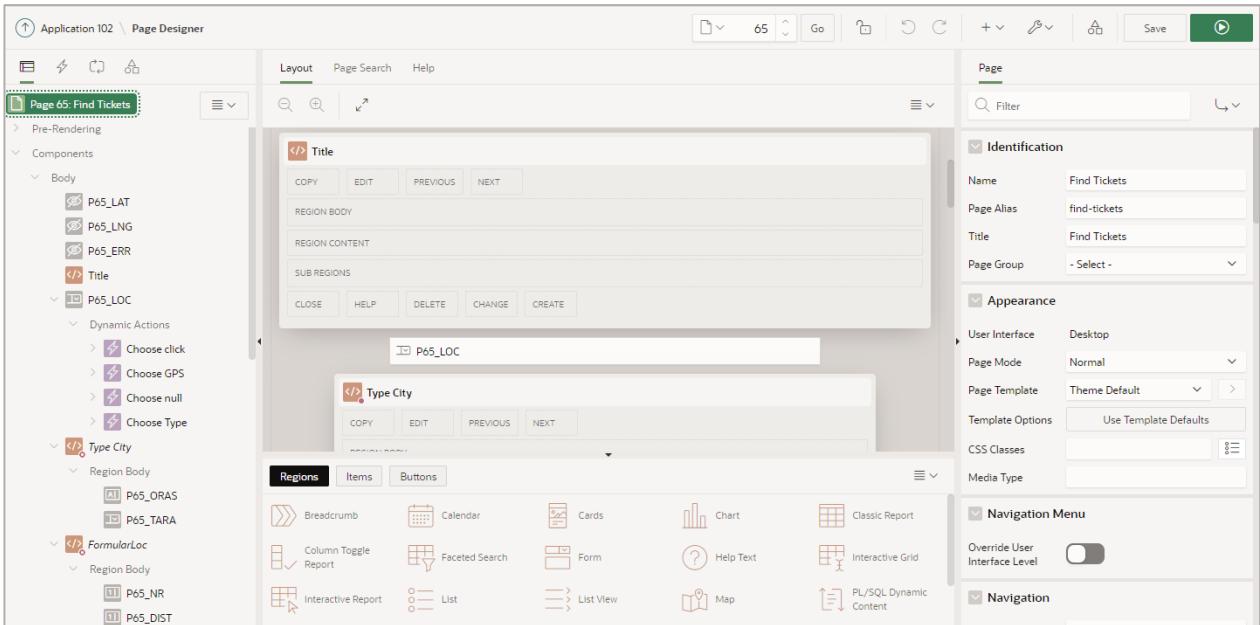


Figura II.1 – Page Designer

II.1.2 SQL Workshop

SQL Workshop reprezintă zona din Oracle Application Express ce pune la dispoziție instrumente pentru vizualizarea și gestionarea obiectelor bazei de date. Fiecare sesiune de lucru în SQL Workshop este asociată unei scheme definite asupra căreia vor fi efectuate operațiile și unde vor fi căutate obiectele.

Secțiunea Object Browser este dedicată managementului obiectelor bazei de date și modificării proprietăților acestora. Pot fi create tabele, indecsi, vizualizări, pachete, subprograme, trigger-i, vizualizări materializate, legături de baze de date, sinonime sau colecții SODA (Simple Oracle Document Access), o serie de API-uri de tip NoSQL folosite pentru stocarea colecțiilor de documente în baza de date Oracle. Panoul obiectului selectat oferă funcționalități precum:

- gestiunea privilegiilor asupra acestuia;
- încărcarea de date în tabele;

- generarea de statistici pe baza datelor;
- generarea codului de creare a obiectului;
- generarea unor exemple de cereri. [6]

Secțiunea SQL Commands furnizează un editor de cod unde pot fi executate comenzi SQL și PL/SQL. Acest mediu este util pentru testarea și optimizarea cererilor ce vor fi folosite în rapoarte și procese, făcând posibilă vizualizarea planului de execuție al acestora. De asemenea, există posibilitatea salvării și păstrării istoricului anumitor interogări executate. Pentru gruparea și reținerea secvențelor complexe de cod este utilizată secțiunea SQL Scripts unde pot fi încărcate, create, salvate și rulate fișiere .sql.

Zona de utilități oferă instrumente pentru crearea rapidă a comenzi SQL, generarea cererilor LDD, vizualizarea rapoartelor despre obiecte, gestionarea valorilor implicate ale interfeței grafice, restaurarea obiectelor bazei de date, compararea mai multor scheme sau monitorizarea bazei de date. [6] Nu în ultimul rând, specificarea declarativă a serviciilor utilizate pentru accesarea bazei de date este realizată în regiunea RESTful Services.

The screenshot shows the Oracle SQL Workshop interface. At the top, there are five tabs: Object Browser, SQL Commands, SQL Scripts, Utilities, and RESTful Services. Below these tabs is a grid of recent activity:

Recently Created Tables	Recent SQL Commands	Recent SQL Scripts
STATISTICA	2 days ago SELECT count(*) from statistica;	2 days ago Creare functii statistici
PROFIL_DOPING	5 weeks ago select * from statistica;	2 days ago Creare tabel statistica
BILET	6 weeks ago SELECT count(*) from statistica;	2 days ago Creare tabel tenismen
BILET_ABONAMENT	6 weeks ago select ID_MECHI, nvl2(id_juc...	2 days ago Creare tabel profil_doping si trigger
BILET_PARTIDA	6 weeks ago SELECT * FROM statistica;	2 days ago Creare tabel profil_soping si trigger
MECI	2 months ago SELECT * FROM Jucator where clasame...	2 days ago Creare biletete
TABLOU	2 months ago SELECT nume, data_sfarsit-data_ince...	2 days ago Creare tabel si trigger comentariu
PARTICIPARE	2 months ago SELECT nume, data_sfarsit-data_ince...	2 days ago Creare tabel subiect
TEREN	2 months ago SELECT nume, data_sfarsit-data_ince...	2 days ago Creare tabel oras
TUR	2 months ago SELECT * from profil_doping;	2 days ago Creare tabel turneu
ATRIBUIRE	2 months ago SELECT l.pozitie.Get_Dimst() dimensi...	2 days ago Creare sevența comenzi_seq
CONTRACT_POST_TV	3 months ago SELECT * FROM profil_doping;	2 days ago Creare functie verifica_juhn

On the right side, there are sections for About, Schema (with a dropdown set to ATP_TOU), Create Object (with options for Table, View, Index, Sequence, Type, Package, Procedure, Function, Trigger, Database Link), and a link to Learn More.

Figura II.2 – SQL Workshop

II.1.3 Servicii RESTful în Oracle APEX

Representational State Transfer (REST) este o modalitate de a oferi interoperabilitate între sisteme informatici prin intermediul internetului. Serviciile RESTful pot fi definite pentru a permite interogarea și manipularea datelor fără a fi nevoie de acces direct la zona de stocare a

acestora. Un serviciu web se numește RESTful atunci când este în conformitate cu principiile REST. Caracteristicile de bază ale unor astfel de servicii sunt evidențiate în ceea ce urmează.

- Resursele ce modeleză un serviciu RESTful sunt identificate prin URL-uri și accesate prin protocolele web `http` sau `https`.
- Pentru manipularea resurselor se folosesc operațiile prestabilite `GET`, `POST`, `PUT`, `DELETE`.
- Nu este menținută nicio conexiune între client și sistemul informatic unde se află serviciul.
- Cererile către un serviciu RESTful obțin întotdeauna un răspuns, acesta fiind de cele mai multe ori într-unul dintre formatele `JSON`, `XML` sau `HTML`.
- Răspunsurile furnizează detalii despre modul în care datele au fost modificate, eventuale mesaje de eroare și *link*-uri către alte resurse conexe. [6]

Serviciile RESTful din Oracle APEX permit crearea unui set de API-uri bazate pe standardele REST ce oferă sistemelor externe posibilitatea de a interacționa în siguranță cu datele disponibile într-un spațiu de lucru APEX fără a fi necesară conectarea directă la baza de date.

Pentru crearea API-urilor, schemele implicate trebuie înregistrate în ORDS (Oracle REST Data Services), o aplicație dezvoltată în Java ce oferă abilitatea publicării serviciilor web RESTful pentru interacțiunea cu datele și subprogramele stocate dintr-o bază de date Oracle. Componentele din care sunt alcătuite serviciile RESTful din Oracle Application Express sunt: module, şablonane și *handler*-e.

Module de resurse

Un modul de resurse reprezintă un container ce grupează împreună un set de servicii RESTful înrudite. Acestea au atât rolul de a identifica grupul în mod unic, cât și de a defini calea de bază utilizată pentru adresele URL ale resurselor.

Un modul de resurse ce permite accesul la informații despre jucători și profilurile de doping ale acestora este `atp.tour.service`, având calea de bază `/itiaservice/`.

Şabloane de resurse

Un modul de resurse conține mai multe şablonane ce definesc servicii individuale care pot fi apelate. Fiecare şablon de resurse constituie un model URI care poate fi accesat și implementeză cel puțin un *handler*. Un model URI reprezintă o cale *case sensitive* atașată rutei către server și căii de bază utilizată pentru solicitarea diferitelor resurse. Aceasta poate include variabile de legătură adăugate după simbolul “`/`” și al căror nume este prefixat de două puncte “`:`” folosite pentru

transmiterea valorilor. Dacă o variabilă din URI este urmată de un semn de întrebare, atunci aceasta este optională, putând lipsi din cale. [6]

Spre exemplu, şablonul `players/:id?` poate fi folosit pentru preluarea informațiilor despre un jucător cu un anumit id, sau despre toți jucătorii dacă niciun id nu este specificat.

Handler-e de resurse

Fiecare *handler* corespunde unei singure operații HTTP pentru şablonul căruia îi este asociat. Implementarea unui *handler* este mapată la una dintre operațiile tradiționale CRUD:

- GET – preia reprezentarea unei resurse, fiind echivalentă cu o instrucțiune SQL de tip SELECT;
- POST – creează o resursă nouă sau adaugă o resursă la o colecție, fiind echivalentă cu o instrucțiune SQL de tip INSERT;
- PUT – actualizează valorile unei resurse existente, fiind echivalentă cu o instrucțiune SQL de tip UPDATE;
- DELETE – șterge o resursă existentă, fiind echivalentă cu o instrucțiune SQL de tip DELETE. [6]

În exemplul precedent, trimitera unei cereri de tip GET la adresa URL `http://localhost:8088/ords/atptour/itiaservice/players/50` va întoarce detalii despre jucătorul cu id-ul 50 sub forma unui obiect JSON în care numele cheilor corespund coloanelor tabelului și valorile corespund conținutului liniei rezultate de interogarea `SELECT * FROM jucator WHERE id_jucator = 50;`.

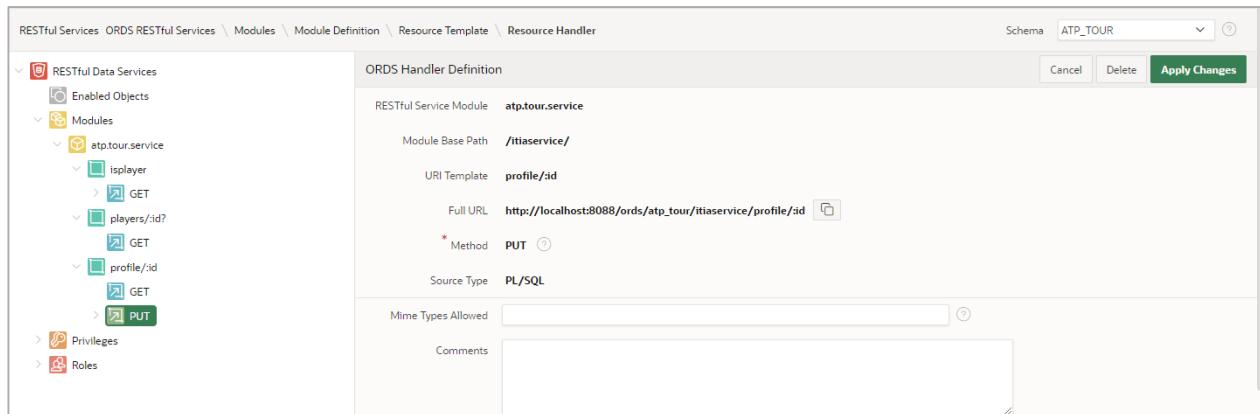


Figura II.3 – RESTful Services

Oracle APEX pune la dispoziție prin intermediul facilității REST Enabled Objects generarea automată a unor API-uri ce implementează toate cele patru tipuri de operații pentru anumite tabele. Cu toate că această opțiune poate fi utilă pentru cazuri standard, lipsa flexibilității și a adaptării la funcționalități mai complexe reprezintă un impediment ce poate fi rezolvat doar prin definirea manuală a fiecărui şablon și *handler* de către dezvoltatori.

II.2 React Native

React Native (cunoscut și ca RN) este un framework bazat pe JavaScript pentru dezvoltarea aplicațiilor mobile atât pentru iOS, cât și pentru Android, folosind același cod sursă. Aceasta a fost lansat pentru prima dată de Facebook în anul 2015, devenind în doar câțiva ani una dintre soluțiile preferate pentru dezvoltarea mobilă. În prezent, aplicații precum Instagram, Facebook și Skype folosesc React Native. [2]

La baza bibliotecii React Native se află framework-ul React (ReactJS) destinat dezvoltării aplicațiilor web. În timp ce React folosește limbajele HTML și CSS pentru definirea și stilizarea interfeței, React Native utilizează componente ce sunt transformate la compilare în elemente specifice iOS sau Android, în funcție de tipul dispozitivului pe care rulează aplicația.

II.2.1 Dezvoltare *cross-platform*

Dezvoltarea *cross-platform* este practica de construire a produselor software compatibile cu multiple platforme hardware. De exemplu, o aplicație web *cross-platform* poate rula pe cel puțin două din sistemele de operare Windows, Linux sau macOS.

React Native a apărut ca un instrument puternic pentru dezvoltarea pe mai multe platforme, permitând programatorilor să creeze aplicații mobile ce oferă performanță și aspectul aplicațiilor native Android sau iOS, timpul și costurile fiind reduse semnificativ.

Principalele caracteristici ale dezvoltării *cross-platform* sunt:

- creșterea numărului de utilizatori ce pot folosi aplicațiile;
- cod reutilizabil;
- accelerarea procesului de dezvoltare, nefiind nevoie de realizarea unei aplicații pentru fiecare sistem în parte;
- reducerea costurilor;
- un timp mai lent de lansare a funcțiilor noi, actualizările din aplicațiile native realizându-se mai rapid. [2]

II.2.2 Hook-uri React

Hook-urile reprezintă o funcționalitate introdusă în versiunea React 16.8 pentru crearea și utilizarea unor funcții speciale ce permit conectarea la ciclul de viață al unei aplicații pentru gestiunea stării și a efectelor secundare. Principalele *hook-uri* predefinite sunt: useState, useEffect și useContext.

Hook-ul useState este folosit în adăugarea de stări componentelor funcționale pentru declararea și gestionarea variabilelor. Acesta primește o valoare inițială și returnează două elemente: starea curentă și o funcție de tip *setter* prin intermediul căreia variabila poate fi actualizată. Principalul avantaj al acestei funcții îl reprezintă faptul că odată ce variabilele asociate stărilor sunt modificate, schimbările se reflectă instant în cadrul componentelor ce le utilizează, inclusiv la nivelul interfeței grafice.

Hook-ul useEffect este folosit pentru gestionarea efectelor secundare ale componentelor funcționale, precum preluarea datelor, actualizarea conținutului afișat pe ecran sau configurarea unor cronometre. Sintaxa implică furnizarea a doi parametrii: o funcție ce definește comportamentul acestuia și o listă optională de dependințe ce declanșează execuția metodei. Dacă al doilea parametru nu este menționat, funcția va fi executată la fiecare randare a componentei, în timp ce transmiterea unei liste goale va determina execuția doar la prima randare, echivalentă, de obicei, cu momentul navigării către ecranul respectiv. Lista poate conține proprietăți și stări ce vor cauza executarea funcției ori de câte ori una dintre acestea își modifică valoarea.

Hook-ul useContext este utilizat pentru accesarea contextului React, o metodă de a transmite elemente de la o componentă părinte către componentele copil fără a fi nevoie de furnizarea acestora ca proprietăți în fiecare nivel al arborelui. De obicei, acesta este folosit ca o zonă globală de depozitare a valorilor folosite frecvent în întreaga aplicație. Câteva exemple de cazuri în care contextul este des întâlnit sunt reținerea utilizatorului curent sau salvarea temei de culori selectate de acesta. [12]

II.2.3 Componete de bază în React Native

Componetele React Native Core reprezintă elementele fundamentale utilizate pentru crearea interfeței grafice într-o aplicație React Native. Acestea sunt predefinite, având posibilitatea de a fi extinse și combinate pentru o complexitate sporită și un aspect adecvat dispozitivelor mobile.

Componenta *View* este structura principală pentru proiectarea UI-ului, aceasta putând fi utilizată și configurață să răspundă la anumite interacțiuni cu utilizatorul. *View-urile* sunt

transformate în componente echivalente de pe platforma unde rulează aplicația, fie acestea de tip *UIView* (iOS), *<div>* (web) sau *android.view* (Android). Un *View* este proiectat să poată fi imbricat în alte *View-uri*, putând avea zero, unul sau mai multe elemente copil de orice tip.

Componentele *Text* și *Image* sunt utilizate pentru afișarea informațiilor de tip text, respectiv imagine, în timp ce elementele de tip *Button* sunt utilizate pentru navigarea între ecrane sau declanșarea execuției unor funcții la apăsare.

Deoarece butoanele nu oferă flexibilitate din punct de vedere al stilizării, acestea sunt des înlocuite de componenta *TouchableOpacity*. Aceasta permite includerea de elemente precum iconițe sau imagini și își schimbă aspectul odată cu atingerea utilizatorului, estompându-se.

Componenta de bază pentru preluarea informațiilor introduse de utilizator prin intermediul tastaturii este *TextInput*. Aceasta poate conține proprietăți pentru setarea corectării automate a cuvintelor, textului substituent și tipului de tastatură. De obicei, unui *TextInput* îi este asociată o variabilă de stare a cărei valoare este modificată la fiecare adăugare sau ștergere a unui caracter.

FlatList este o componentă utilă în redarea eficientă a listelor de elemente prin furnizarea unui id unic pentru fiecare. Aceasta suportă definirea unui antet sau a unui subsol, posibilitatea derulării în cazul unei listei de mari dimensiuni, orientarea verticală sau orizontală și configurarea unei componente pentru delimitarea elementelor consecutive. [12]

II.3 Expo

Expo este un instrument construit în jurul *framework-ului React Native* menit să simplifice și să accelereze procesul de dezvoltare al aplicațiilor mobile. Una dintre caracteristicile sale cheie este reprezentată de biblioteca extinsă de componente și API-uri predefinite ce acoperă o gamă largă de funcționalități, de la elemente ale interfeței grafice până la funcții ale dispozitivului precum accesul la cameră și localizarea geografică. Pe lângă acestea, SDK-ul Expo oferă posibilitatea utilizării unor API-uri native ce permit programatorilor să profite de toate capabilitățile platformei de bază, precum accesarea senzorilor sau a sistemului de fișiere al telefonului.

Expo include, de asemenea, o suiată de mecanisme pentru simplificarea dezvoltării aplicațiilor. Acestea constau într-un *CLI (Command Line Interface)* puternic pentru crearea, rularea și trimiterea în producție a produselor software și un instrument robust pentru identificarea și remedierea erorilor în timp real. În plus, Expo este conceput pentru a se integra cu cele mai populare editoare de cod și sisteme de control al versiunilor.

La rularea unui proiect prin intermediul Expo CLI este pornit un server pe portul implicit 8081 pe care un client îl poate folosi pentru a interacționa cu *bundler*-ul creat la compilare prin combinarea tuturor fișierelor, modulelor și dependințelor incluse în aplicație. Interfața apărută după lansarea proiectului conține un cod QR pentru adresa URL a serverului de dezvoltare și o listă de comenzi rapide ce pot fi utilizate pentru:

- deschiderea aplicației pe un dispozitiv iOS sau Android conectat;
- deschiderea aplicației pe un simulator iOS sau Android;
- deschiderea aplicației într-un browser web;
- reîncărcarea aplicației pe dispozitivul conectat;
- deschiderea meniului de comenzi speciale pentru dezvoltatori. [11]

Proiectele Expo pot fi rulate, de asemenea, direct pe telefon prin intermediul aplicației Expo Go atât timp cât dispozitivul mobil și cel pe care este realizată dezvoltarea sunt conectate la aceeași rețea Wi-Fi și la același cont Expo.

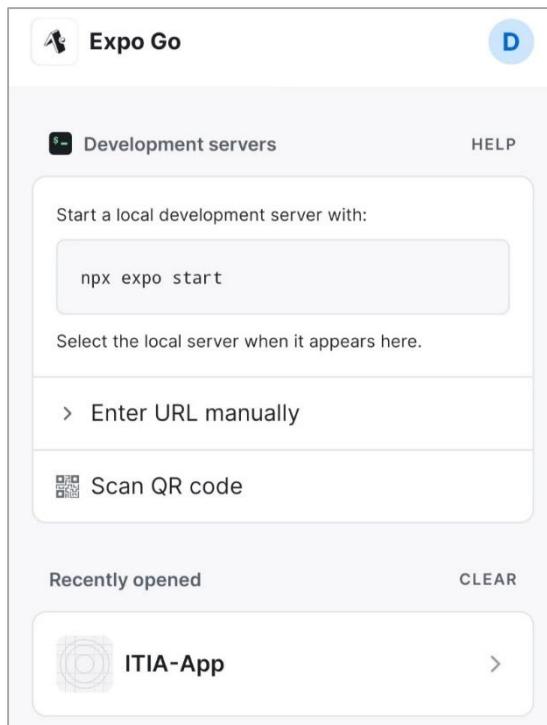


Figura II.4 – Pagina principală Expo Go

Spre deosebire de web, React Native nu oferă compatibilitate cu sistemele anterioare, fiind necesară instalarea versiunilor exacte ale pachetelor adăugate prin intermediul instrumentelor *npm/npx* sau *Yarn*. Astfel, Expo CLI oferă o modalitate de regăsire automată a combinațiilor de versiuni potrivite prin adăugarea la instalare a comenзii *expo* înaintea numelor pachetelor. [11]

II.4 Typescript

Typescript este un limbaj de programare creat și întreținut de Microsoft ce extinde JavaScript pentru o dezvoltare mai ușoară și sigură a aplicațiilor la scară largă. Typescript pune la dispoziție un sistem de module, clase și interfețe pentru definirea tipurilor de date ale variabilelor dintr-un program. Acest strat suplimentar oferă dezvoltatorilor posibilitatea unei organizări mai eficiente a codului și detectarea timpurie a potențialelor erori.

Declararea statică a tipurilor pentru variabile, parametrii funcțiilor și valorile returnate reprezintă principalul avantaj al limbajului, identificarea erorilor legate de tipurile de date putându-se realiza în timpul dezvoltării și nu în momentul execuției. [1]

Typescript acceptă utilizarea funcționalităților moderne JavaScript, inclusiv a celor introduse în ECMAScript 2015 (ES6) și versiunile ulterioare. Astfel, programatorii pot folosi concepte precum *arrow functions*, clase sau module ce vor fi compilate în cod JavaScript compatibil. De asemenea, deoarece Typescript este construit pe baza limbajului JavaScript, codul JavaScript existent poate fi migrat treptat la cod Typescript fără a necesita modificări majore. Dezvoltatorii pot începe prin adăugarea de adnotări de tip la programul scris și îl pot refactoriza pentru a profita din plin de caracteristicile Typescript.

Pe lângă declararea statică, Typescript dispune de un sistem puternic de inferență ce deduce automat tipurile de date în funcție de context. Acest lucru reduce necesitatea adnotărilor explicite, oferind codului un aspect mai curat și concis. În plus, față de tipurile primitive, limbajul permite definirea interfețelor și genericelor pentru descrierea structurii obiectelor, respectiv crearea de componente și funcții reutilizabile ce lucrează cu o varietate mare de tipuri de date. [1]

Typescript este compatibil cu diverse medii de dezvoltare integrate (IDE), cum ar fi *Visual Studio Code*, *Sublime Text* și *WebStorm*. Aceste instrumente oferă facilități precum completarea automată a codului, verificarea tipurilor și suport pentru refactorizare, îmbunătățind productivitatea programatorilor.

Proiectele React Native și Expo pot fi inițializate pentru a folosi Typescript ca limbaj de programare. Astfel, fișierele *.js* și *.jsx* din aplicații se vor transforma în documente *.ts*, respectiv *.tsx*, iar proprietățile, metodele, variabilele și stările din componente funcționale vor avea asociate tipuri de date. Câteva dintre cazurile de utilizare des întâlnite sunt: inferența tipurilor variabilelor de stare din valorile inițiale, definirea formatului proprietăților transmise la navigarea între ecrane și impunerea structurii obiectelor folosite în aplicație.

III FORMULAREA PROBLEMEI ȘI PROIECTAREA BAZEI DE DATE

III.1 O scurtă istorie a tenisului

Având cele mai îndepărtate origini în diverse civilizații mediteraneene antice, tenisul a fost popularizat în Europa în secolele XI-XII când a devenit o activitate recreațională uzuale pentru aristocrații francezi. Meritul pentru inventarea tenisului de câmp modern îi este oferit ofițerului galez al armatei britanice Walter Clopton Wingfield, care în anul 1873 a venit cu ideea transformării sportului dintr-unul jucat în interior într-unul jucat în aer liber. De asemenea, acesta a proiectat mingea de cauciuc care este folosită și în zilele noastre. Regulile jocului au fost enunțate în anul 1874, urmând ca primul campionat să se desfășoare în 1877 la Wimbledon.

Cuvântul românesc „tenis” provine din englezescul „tennis”, care la rândul său își are originea din franceza veche, prin termenul anglo-normand „tenez”, care poate fi tradus prin „țineți!”, „primiți!” sau „ia！”, un strigăt de la un jucător către adversarul său indicând faptul că este pe cale să servească.

Tenisul se află acum în primele 5 cele mai populare sporturi din lume cu aproximativ un miliard de persoane jucând-ul sau urmărind-ul la televizor.

III.2 Formularea problemei

Circuitul ATP (Association of Tennis Professionals) reprezintă circuitul profesionist de tenis masculin și a fost înființat în anul 1990. Pentru a modela desfășurarea acestuia și interacțiunea cu spectatorii și sportivii, au fost enunțate următoarele reguli:

- de-a lungul unui sezon se desfășoară mai multe turnee, fiecare dintre acestea aparținând unei categorii (Grand Slam, ATP Masters 1000, etc.) în funcție de care sunt acordate punctele și premiile în bani;
- orice turneu se află într-o anumită locație descrisă de numele complexului sportiv în cadrul căruia are loc competiția;
- o țară conține mai multe orașe care la rândul lor pot cuprinde multiple locații;
- persoanele implicate în circuit sunt: jucători, antrenori și arbitri;

- un jucător poate fi antrenat de mai mulți antrenori și un antrenor poate lucra cu mai mulți tenismeni;
- un jucător poate fi sponsorizat de mai multe firme și un sponsor poate avea parteneriate cu numeroși sportivi;
- oricare doi jucători au un scor în meciurile directe precedente dintre aceștia;
- la un turneu se înscriu jucători pe baza clasamentului actual, cei mai buni 4/8/16/32 primind statutul de cap de serie în funcție de dimensiunea tabloului de concurs;
- fiecare competiție conține mai multe terenuri caracterizate de capacitate și tururi al căror preț este cunoscut;
- mai mulți arbitri sunt repartizați fiecărui turneu pentru a arbitra diverse partide din cadrul acestuia;
- într-un turneu se desfășoară meciuri ce au loc între exact doi jucători distincți, într-un anumit tur, pe unul din terenurile turneului și sunt oficiale de un arbitru de scaun;
- pe parcursul unui turneu, odată cu încheierea meciurilor, tabloul de concurs este actualizat;
- un jucător are înregistrat un set de statistici pentru fiecare meci jucat, precum numărul de ași, duble greșeli, lovitură câștigătoare sau erori neforțate;
- turneele sunt transmise de anumite posturi de televiziune ce, în funcție de tipul contractului, acoperă toate meciurile din concurs sau doar confruntările ce au loc pe terenul central al complexului;
- utilizatorii pot cumpăra bilete la turnee, acestea fiind de două tipuri: bilete speciale pentru fiecare meci și bilete abonament ce permit accesul la toate partidele din turneu sau dintr-o anumită zi a competiției;
- fiecare jucător, antrenor, arbitru, utilizator și post tv provine dintr-o țară;
- fanii pot deschide subiecte de discuție și comenta pe forum pentru a interacționa cu alte persoane pasionate de tenis;
- pe parcursul sezonului competițional are loc un concurs în care utilizatorii rezolvă lunar câte un quiz cu cinci întrebări pentru șansa de a câștiga la finalul anului mingi, manșete sau rachete cu autograful jucătorilor;
- pe lângă utilizatorii uzuali, tenismenii își pot crea conturi ce vor fi asociate informațiilor deja înregistrate pentru jucători;
- orice cont de tenismen are un profil de doping ce conține locația și intervalul orar în care acesta este disponibil pentru testare.

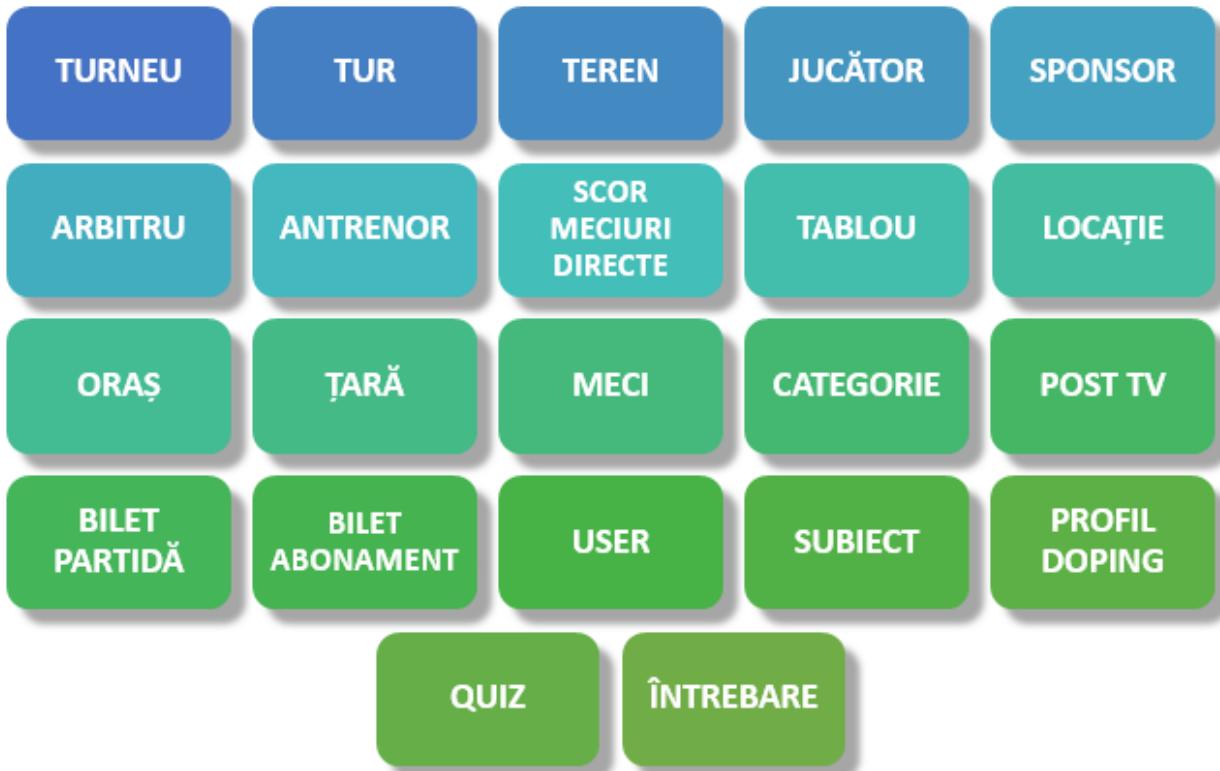


Figura III.1 – Entitățile identificate

III.3 Diagramele bazei de date

Odată ce entitățile participante în modelarea problemei sunt identificate pe baza regulilor descrise precedent, pasul următor constă în conectarea lor prin intermediul diagramei entitate-relație (E/R). În cadrul acesteia se ilustrează tipurile de relații (*one-to-one*, *one-to-many*, *many-to-many*) dintre entități și cardinalitățile minime și maxime ale legăturilor.

Din diagrama E/R va fi construită diagrama conceptuală ce va conține toate atributele entităților și cheile externe ce descriu relațiile *one-to-one* și *one-to-many* dintre tabele. De asemenea, relațiile *many-to-many* precum SPONSOR sponsorizează JUCĂTOR, ANTRENOR antrenează JUCĂTOR, POST TV transmite TURNEU, JUCĂTOR participă la TURNEU, ARBITRU este atribuit la TURNEU, JUCĂTOR are statistici la MECHI, USER comentează la SUBIECT și USER răspunde la QUIZ sunt transformate în tabelele asociative CONTRACT SPONSOR, CONTRACT ANTRENOR, CONTRACT TV, PARTICIPARE, ATRIBUIRE, STATISTICĂ, COMENTARIU și SCOR QUIZ. Această reprezentare va constitui fundamentul implementării bazei de date din aplicație.

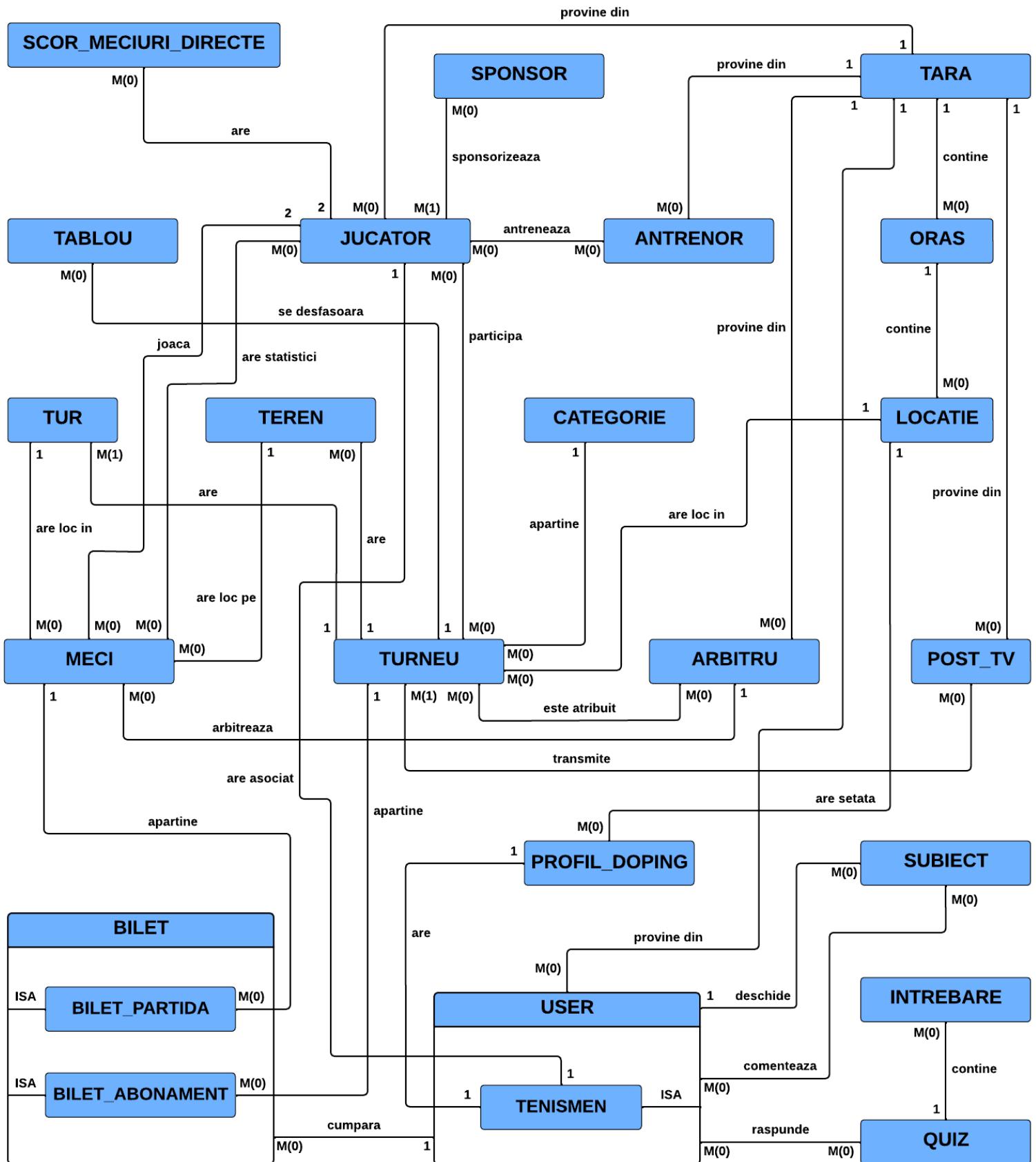


Figura III.2 – Diagrama entitate-relatie

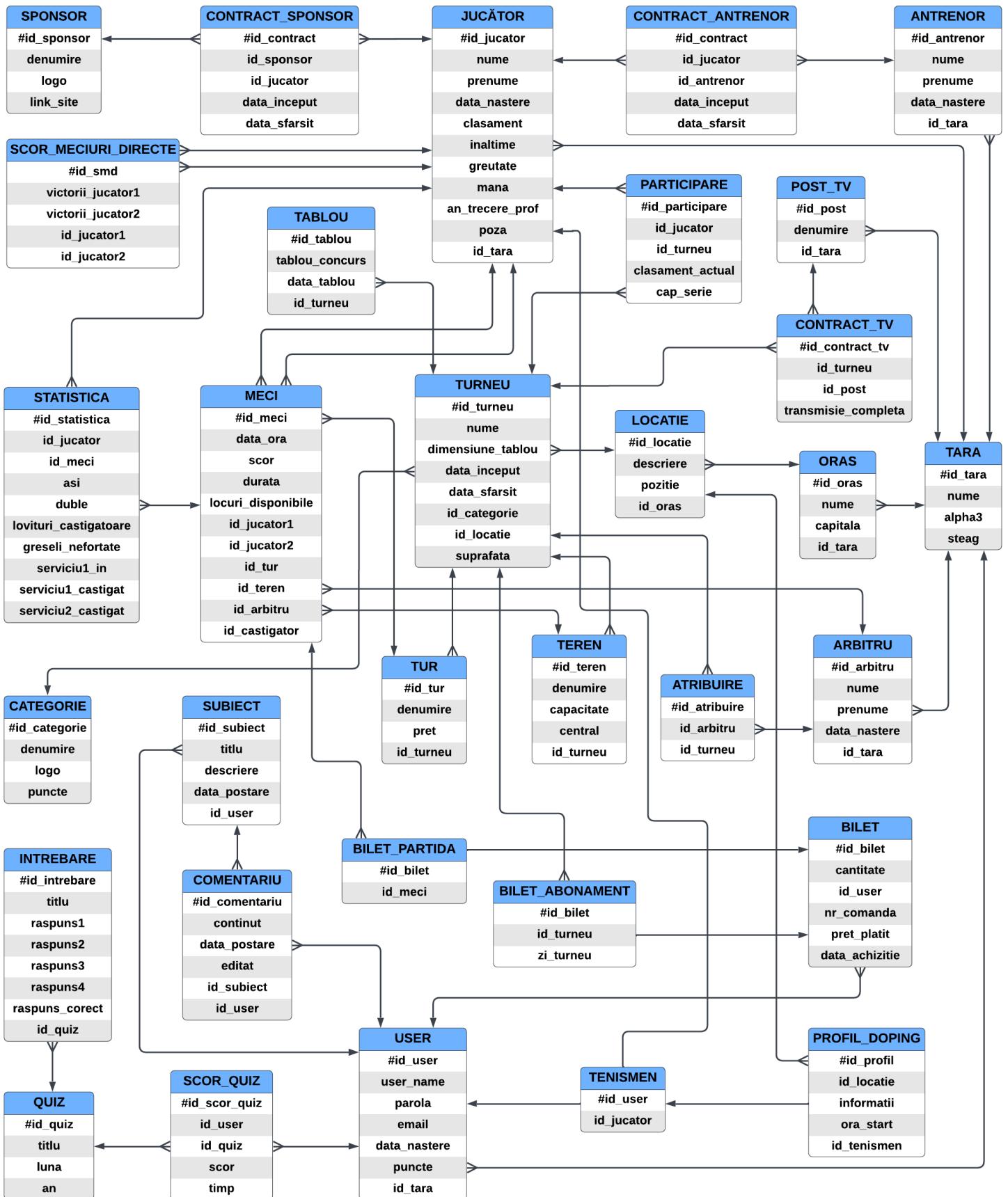


Figura III.3 – Diagrama conceptuală

IV DEZVOLTAREA APLICAȚIEI WEB

IV.1 Descrierea aplicației

Aplicația web ATP Tour dezvoltată în Oracle APEX are scopul de a oferi fanilor tenisului un loc accesibil unde pot să găsească toate informațiile utile legate de circuit, să își planifice viitoarele participări la turnee, să achiziționeze bilete la meciuri și să interacționeze cu alte persoane pasionate de sport. De asemenea, statisticile loviturilor sportivilor prezente în aplicație pot fi utile jucătorilor ce nu își permit angajarea unui analist de date pentru studierea profilurilor de joc ale adversarilor și pregătirea tacticilor înainte de meci.

Utilizatorii pot folosi o mare parte din aplicație fără a fi nevoie de crearea unui cont. Acțiunile nepersonalizate precum vizualizarea detaliilor despre turnee, jucători sau meciuri și filtrarea competițiilor pe baza locației sau caracteristicilor sunt disponibile în întregime publicului larg. Odată logat, un utilizator poate dispune de unul sau mai multe dintre cele patru roluri existente: USER, EDITOR, ADMINISTRATOR sau JUCĂTOR.

Rolul implicit acordat la crearea unui cont este cel de USER care deblochează funcționalități precum cumpărarea biletelor, rezolvarea quiz-urilor lunare sau accesul la forumul de discuții. EDITORII au sarcina și permisiunea de a gestiona informațiile din aplicație astfel încât să reflecte desfășurarea turneelor și evoluția jucătorilor din circuit. În plus, aceștia se ocupă de crearea quiz-urilor și au grijă să modereze postările și comentariile inadecvate de pe forum.

Rolul de ADMINISTRATOR conține cele mai multe privilegii, permitând schimbarea rolurilor utilizatorilor din aplicație și ștergerea conturilor acestora în situația în care au încălcat în mod repetat regulile comunității. [5]

Nu în ultimul rând, rolul de JUCĂTOR este folosit în accesarea aplicației mobile pentru setarea informațiilor profilului de doping. Detaliile profilurilor tuturor jucătorilor pot fi vizualizate în aplicația web de utilizatorii cu rolul de ADMINISTRATOR.

Cerințele funcționale ale produsului software sunt modelate prin intermediul unor diagrame UML (Unified Modeling Language) ce includ clasele, actorii, acțiunile și rolurile participante. Pentru ilustrarea corespondențelor dintre diferitele rolului ale utilizatorilor și principalele funcționalități au fost folosite diagrame de tip *use case*. Pentru redarea trecerii complete a unui USER prin aplicație se utilizează o diagramă secvențială ce descrie interacțiunea dintre actori și obiectele sistemului.

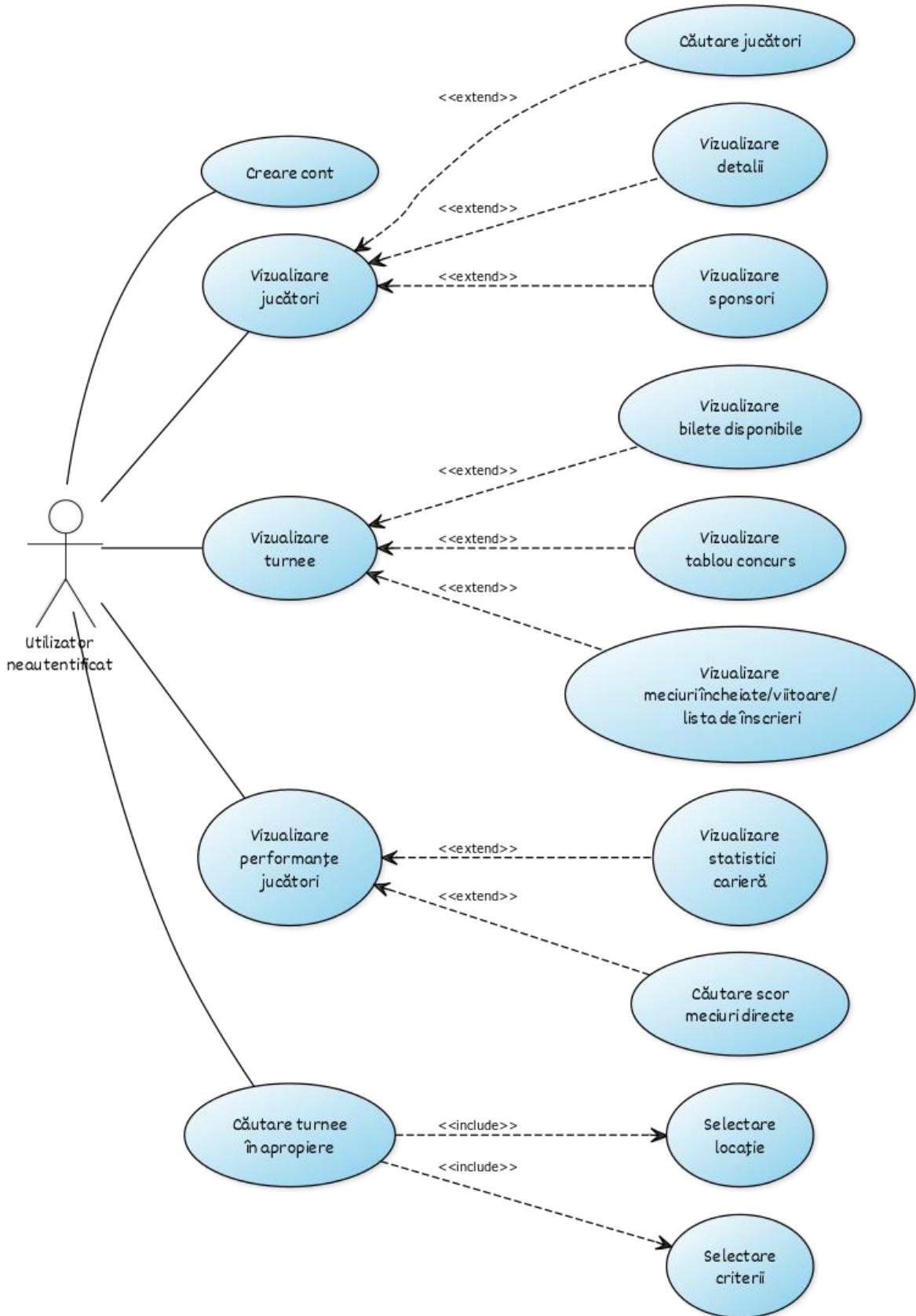


Figura IV.1 – Diagrama UML use case 1

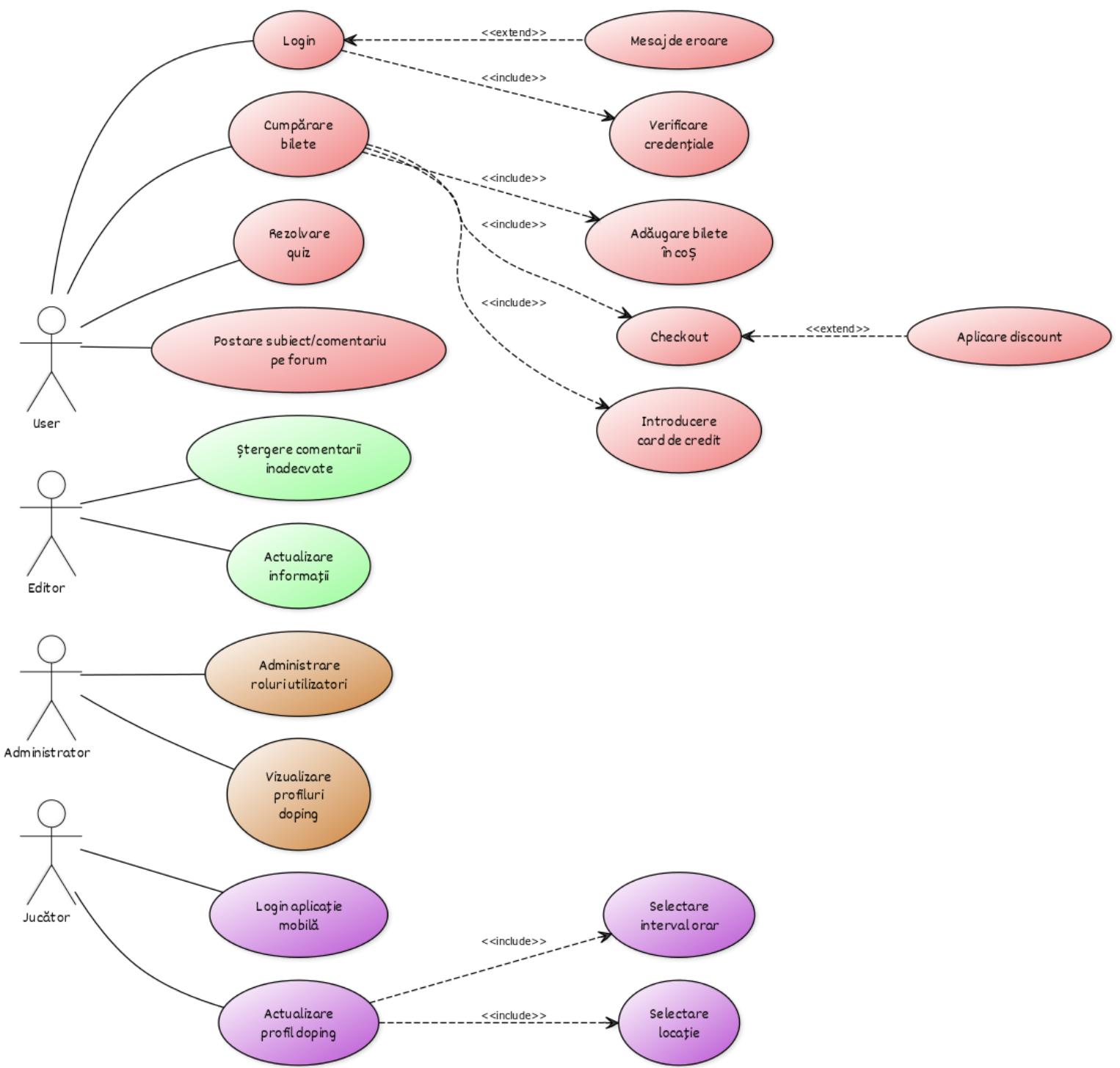


Figura IV.2 – Diagrama UML use case 2

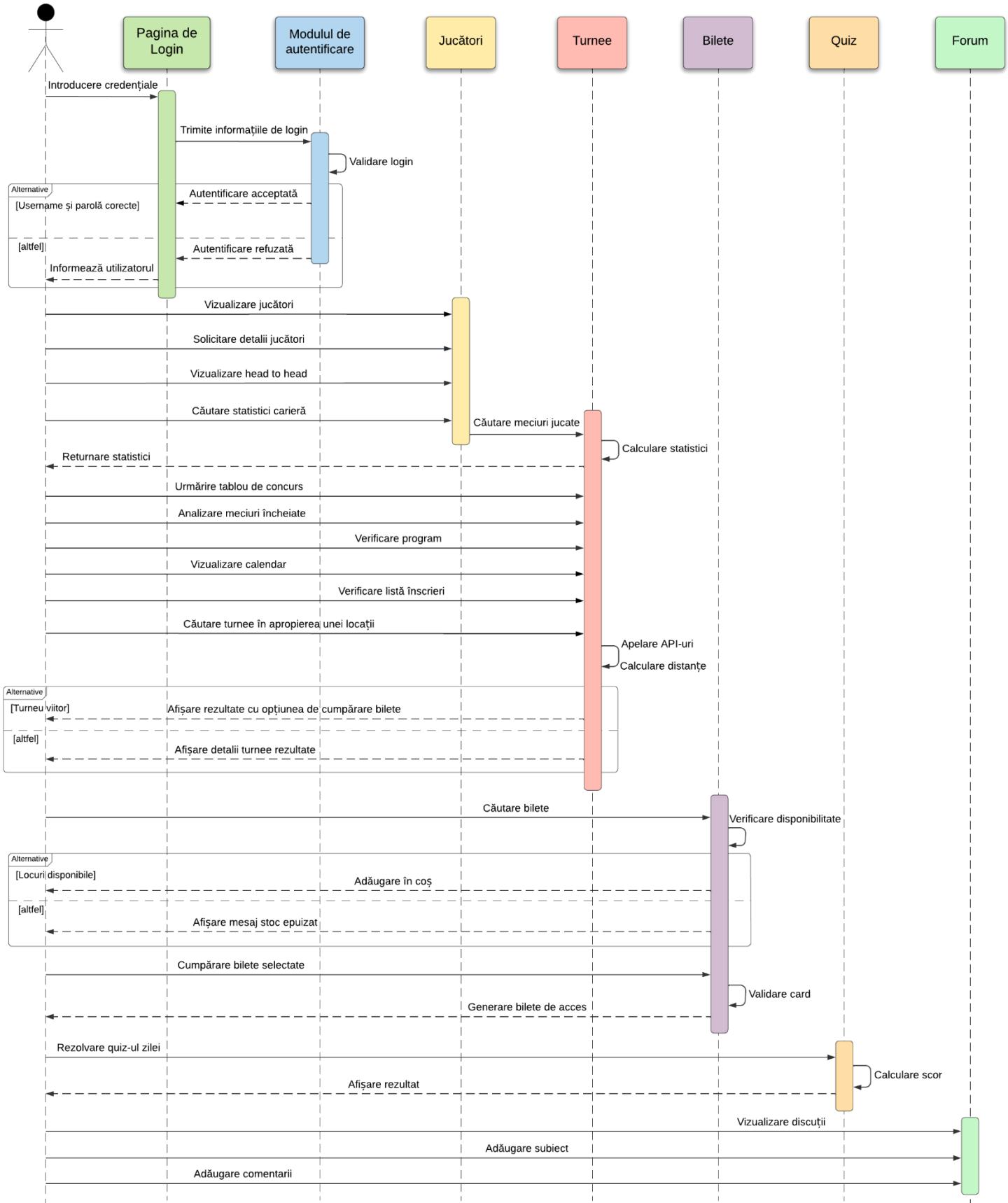


Figura IV.3 – Diagrama UML secvențială

IV.2 Autentificare și autorizare

Pentru crearea unui cont este nevoie de introducerea unui nume de utilizator unic la nivelul aplicației, unei adrese de email, datei de naștere, țării de proveniență și parolei. Formularul conține validări ce asigură:

- formatul corect al adresei de email cu ajutorul unei expresii regulate;
- complexitatea parolei (minim 8 caractere dintre care cel puțin o literă mare, un caracter numeric și unul special);
- unicitatea numelui de utilizator și a adresei de email;
- data nașterii corespunzătoare unei persoane cu o vârstă de cel puțin 12 ani;
- parola și confirmarea acesteia conțin aceeași valoare.

De asemenea, utilizatorii își aleg țara de proveniență prin intermediul unei liste de selecție ce conține toate statele lumii.

Securitatea parolelor este asigurată de stocarea în mod criptat a acestora în baza de date cu ajutorul algoritmului de *hashing* SHA-256 pus la dispoziție de pachetul DBMS_CRYPTO.

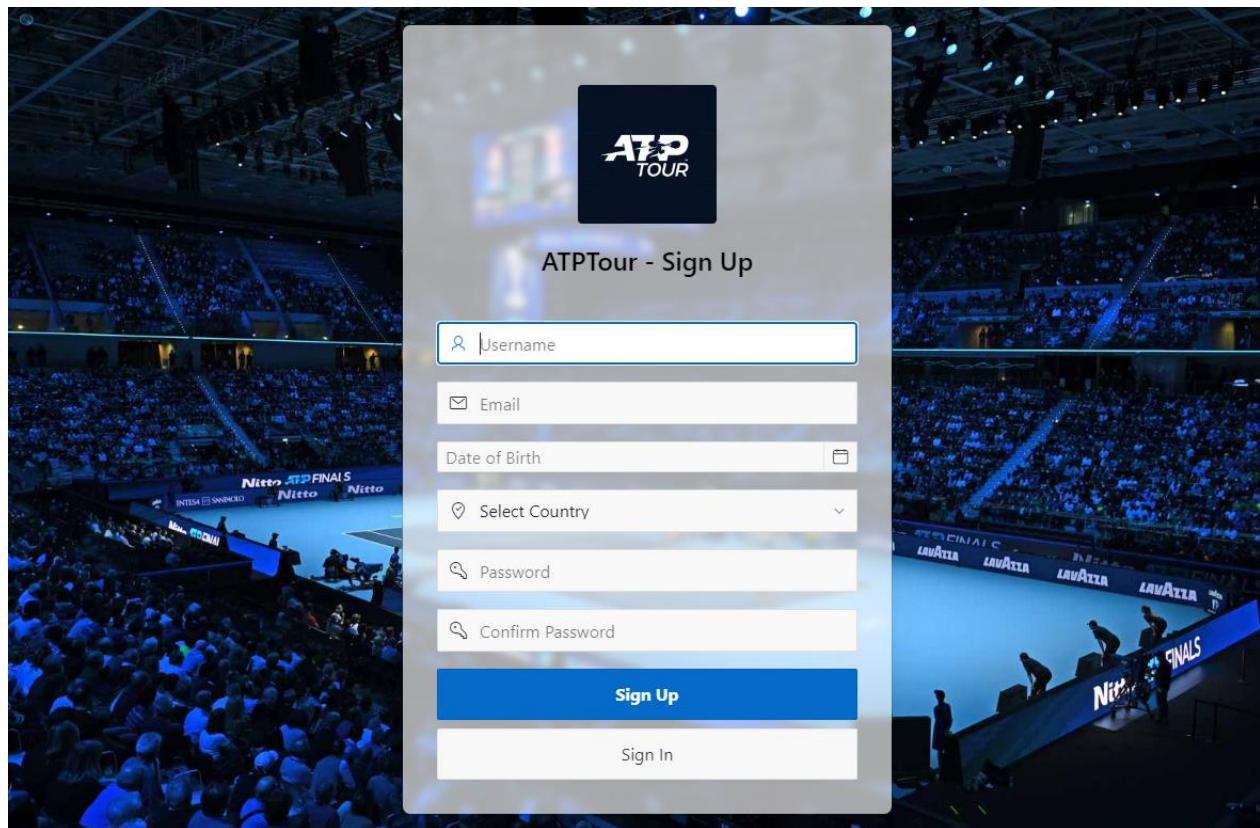


Figura IV.4 – Pagina de creare a contului

Autentificarea se realizează pe baza numelui de utilizator și a parolei, utilizatorii fiind redirecționați către pagina principală. Aceasta conține informații generale despre turnee și clasament și o regiune de tip carusel ce derulează logourile principalilor sponsori ai circuitului. Dacă o persoană neautentificată încearcă să acceseze o pagină ce necesită deținerea unui cont, după logarea cu succes a utilizatorului, acesta va fi trimis automat către pagina respectivă.

The screenshot shows the ATP Tour website. At the top right, there are icons for a user profile, shopping cart, and search. The main header is "ATP Tour". Below it, a tagline reads: "ATP is the global governing body of men's professional tennis. We entertain a billion fans and showcase the game's greatest players on its greatest stages." To the right is the ATP TOUR logo. The left side features a large image of Novak Djokovic holding a trophy. The right side contains sections titled "WHERE WE PLAY?" and "WHAT ARE THE ATP RANKINGS?", each with descriptive text and small images.

Figura IV.5 – Pagina principală

Pentru managementul utilizatorilor administratorii au acces la o pagină din cadrul căreia pot atribui roluri sau șterge conturi. Dacă se dorește atribuirea rolului de jucător unui utilizator, atunci, odată cu bifarea căsuței corespunzătoare acestuia, va apărea o listă de selecție din care trebuie ales sportivul stocat în tabelul *JUCATOR* ce va fi asociat contului.

Autorizarea constă în securizarea paginilor, regiunilor, componentelor și butoanelor astfel încât să nu poată fi vizualizate sau accesate de către utilizatorii ce nu dețin suficiente privilegii. Aceasta este creată prin intermediul schemelor de autorizare puse la dispoziție de APEX. Este important ca autorizarea paginilor să fie implementată atât la nivelul interfeței grafice prin ascunderea butoanelor și a *link-urilor* către ele, cât și la nivelul backend-ului pentru a împiedica accesul prin URL. Dacă un utilizator autentificat încearcă să introducă adresa unei pagini la care

au acces doar editorii sau administratorii, atunci acesta va fi redirectionat către o pagină de eroare ce afișează un mesaj de tipul „Privilegii insuficiente!”. Administratorii au disponibil un raport ce conține informații despre fiecare user, moment de timp și pagină pe care a încercat să o acceseze fără a avea privilegiile necesare.

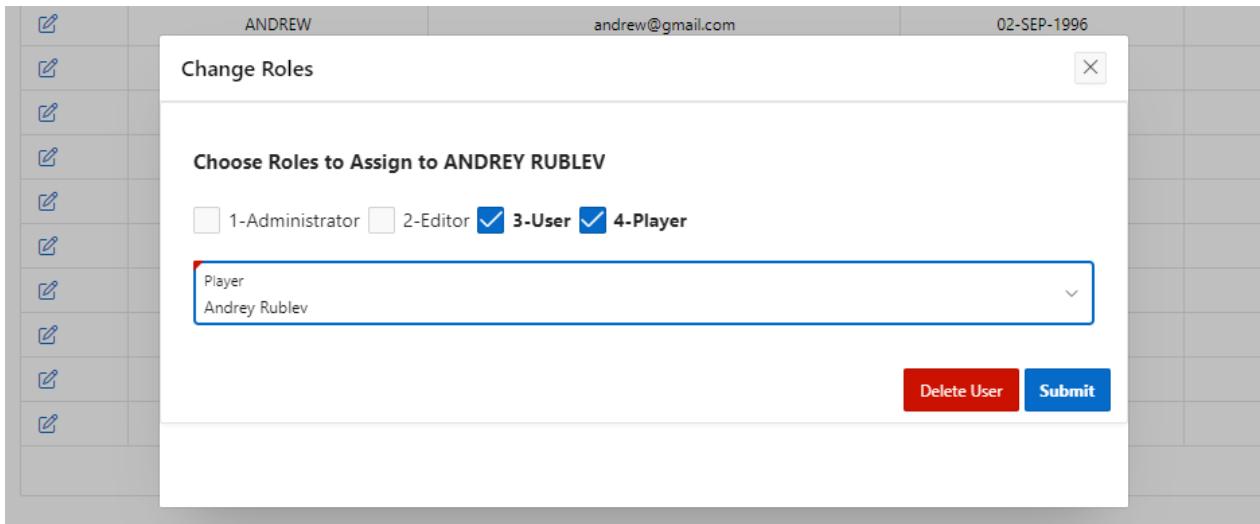


Figura IV.6 – Pagina de atribuire a rolurilor

IV.3 Gestiunea informațiilor

Pentru managementul datelor din aplicație este pusă la dispoziție pagina de administrare ce poate fi accesată de utilizatorii cu rol de EDITOR și ADMINISTRATOR. Aceasta conține o listă ale cărei intrări reprezintă *link-uri* către paginile de gestionare a datelor din majoritatea tabelelor bazei de date. [5]

Administration	
	TV Channel Management Add, edit, delete TV channels
	City Management Add, edit, delete cities
	Location Management Add, edit, delete locations
	Tournament Management Add, edit, delete tournaments
	TV Contract Management Add, edit, delete TV Contracts

Figura IV.7 – Pagina de administrare

Informațiile legate de entitățile din aplicație referitoare la desfășurarea circuitului, precum jucători, turnee, contracte sau meciuri trebuie adăugate și actualizate de editori pentru a fi în conformitate cu rezultatele și detaliile existente. În plus față de lista de management disponibilă editorilor, administratorii au acces la pagina de administrare a utilizatorilor prezentată anterior și la secțiunea de vizualizare a profilurilor de doping ale jucătorilor.

Fiecare pagină de gestionare a datelor dispune de un raport interactiv cu ajutorul căruia persoanele responsabile pot filtra, sorta și regăsi cu ușurință informațiile necesare. Pentru situațiile în care un anumit tabel este puternic dependent de altul (ex. *INTREBARE* de *QUIZ*, *MECI* de *TURNEU*), iar gestionarea datelor acestuia pe cont propriu ar fi neintuitivă, sunt folosite pagini de tip *Master Detail*. O astfel de pagină conține o regiune în care sunt enumerate liniile tabelului de tip *Master* (*QUIZ*, *TURNEU*), iar accesarea unui element al listei va deschide rapoartele corespunzătoare datelor din tabelele de tip *Detail* (*INTREBARE*, *MECI*). Această reprezentare a relațiilor de tip *one-to-many* facilitează vizualizarea datelor corelate.

Match Management											
<input type="text" value="Search..."/> ✖ Reset											
Tournament											
Name Cincinnati Open											
Start Date 12-AUG-2024											
End Date 19-AUG-2024											
Matches +											
	Player 1		Score	Player 2		Winner	Court	Round	Date and Time	Umpire	Duration
Mubadala Citi DC Open 2024	Felix Auger-Aliassime	▶	4-6 6-2 6-3	Jack Draper	▶	Felix Auger-Aliassime	Center Court	1st Round	12-08-2024 11:00	Greg Allensworth	145 min
Atlanta Open 2024	Taylor Fritz	▶	7-6(14) 6-2	Iiri Lehecka	▶	Taylor Fritz	Center Court	1st Round	12-08-2024 13:00	Miriam Bley	105 min
Generali Open 2024	Gael Monfils	▶	3-6 6-4 6-3	Cameron Norrie	▶	Gael Monfils	Center Court	1st Round	12-08-2024 15:00	John Blom	131 min
Plava Laguna Croatia Open Umag 2024	Novak Djokovic	▶	6-2 6-3	Fabian Marozsan	▶	Novak Djokovic	Center Court	1st Round	12-08-2024 19:00	Jaume Campistol	58 min

Figura IV.8 – Pagina de gestiune a meciurilor în funcție de turneu

Paginile de management al informațiilor conțin butoane de adăugare și editare a înregistrărilor existente ce deschid formularele corespunzătoare în ferestre modale. În exemplul de mai sus, în dreapta numelui fiecărui jucător există, de asemenea, un buton ce permite actualizarea statisticilor acestuia în cadrul meciului respectiv.

Formularul pentru gestionarea meciurilor conține liste de selecție pentru câmpurile corespunzătoare jucătorilor, arbitrului de scaun, terenului și turului. Deoarece id-ul turneului din care face parte meciul este transmis paginii ce conține formularul, listele de selecție vor fi compuse doar din jucătorii deja înscrisi în competiție, arbitrii, tururile și terenurile asociate turneului. Astfel, sunt prevenite eventualele erori ce pot apărea la inserarea sau actualizarea cheilor externe. În plus, prin utilizarea unei liste de valori în cascădă se impune ca învingătorul partidei să fie ales dintre cei doi jucători aflați în câmpurile *Player 1* și *Player 2*. De asemenea, există validări ce asigură formatul corect al scorului și faptul că data meciului se află în intervalul de desfășurare a turneului.

Round	Date and Time
1st Round	12-08-2024 11:00
1st Round	12-08-2024 13:00
1st Round	12-08-2024 15:00

Figura IV.9 – Formularul de adăugare/editare a unui meci

Toate celelalte entități ce necesită administrarea datelor au asociate rapoarte și formulare similare ce conțin validări corespunzătoare tuturor regulilor de business și constrângerilor din baza de date, ușurând munca editorilor.

IV.4 Jucători

Meniul dedicat jucătorilor este compus din două secțiuni: clasament și performanțe. Pagina clasamentului conține toți tenismenii ordonați în funcție de locul ocupat în ierarhia mondială la momentul actual. Sunt afișate informații precum țara de proveniență, numele, prenumele și poza oficială pe baza cărora utilizatorii pot regăsi anumiți jucători prin intermediul barei de căutare. De

asemenea, în partea din stânga se regăsește o regiune de tip *Faceted Search* cu ajutorul căreia sportivii pot fi filtrați pe baza naționalității, înălțimii, greutății sau clasamentului. [5]

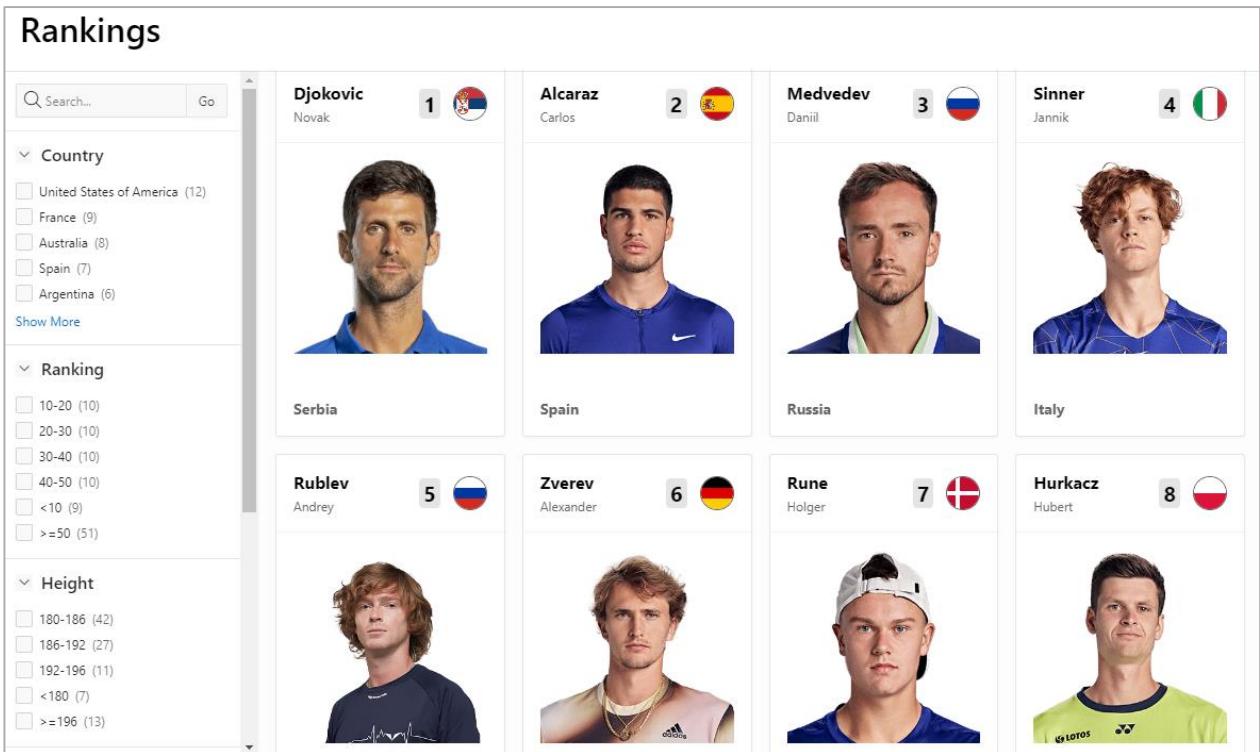


Figura IV.10 – Pagina clasamentului mondial

Pentru a putea afla mai multe detalii despre jucători, utilizatorii pot da click pe aceștia din cadrul clasamentului, accesând astfel pagina ce conține, pe lângă informațiile anterioare, data nașterii, vîrsta, anul trecerii la profesioniști și actualii antrenori. În plus, există o listă a sponsorilor ce include *link-uri* către site-urile firmelor și o regiune cu rezultatele ultimelor zece meciuri jucate.

Pagina performanțelor jucătorilor constă în scorurile din meciurile directe și statistici ale jocului. Ambele regiuni oferă inputuri de tip *autocomplete* ce ușurează introducerea numelor. [5]

Fereastra apărută la căutarea victoriilor dintre doi jucători este alcătuită dintr-o comparație a informațiilor despre aceștia și numărul de partide câștigate de fiecare contra celuilalt. Deoarece numărul de scoruri din meciurile directe ce trebuie aduse în aplicație este mare, a fost creat un robot RPA în platforma UiPath ce preia automat pe baza unui fișier Excel toate perechile distincte de jucători și introduce numele acestora în secțiunea *Head 2 Head* de pe site-ul oficial ATP. Prin intermediul selectorilor CSS și DOM-ului HTML, robotul apasă automat pe butonul de căutare și extrage valorile rezultate, la final generând comanda de inserare SQL. Astfel, timpul petrecut de dezvoltator pentru adăugarea datelor este redus, iar șansa erorilor umane scade semnificativ.

Novak Djokovic (SRB)



Ranking: 1

Date of birth: 22-May-1987 (36)

Plays: Right-handed

Height / Weight: 188 cm / 77 kg

Turned Pro: 2003

Coach(es): Goran Ivanisevic



Matches

Novak Djokovic (1) d. Alejandro Davidovich Fokina
6-4 6-0

2nd Round Cincinnati Open 2024

Novak Djokovic (1) d. Fabian Marozsan
6-2 6-3

1st Round Cincinnati Open 2024

Figura IV.11 – Pagina detaliilor despre jucători

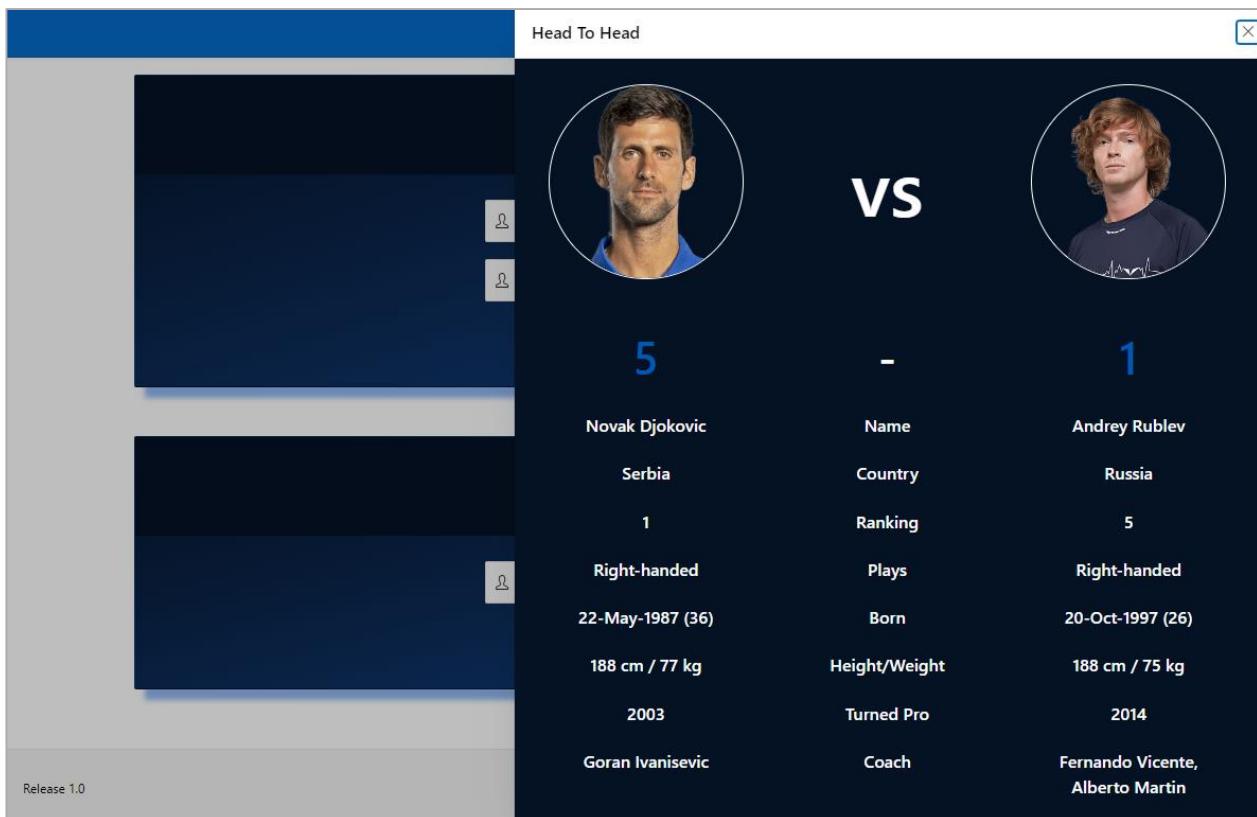


Figura IV.12 – Pagina scorului din meciurile directe

Pe baza statisticilor înregistrate pentru fiecare jucător în meciurile disputate, utilizatorii pot vizualiza și analiza profilul sportivilor prin intermediul valorilor ce descriu diversi parametri ai jocului. Astfel de cifre pot fi regăsite în fereastra statisticilor carierei unui jucător și contorizează numărul mediu de ași/duble greșeli/greșeli neforțate/lovituri câștigătoare. De asemenea, sunt înregistrate procentajele la serviciu pentru primul și al doilea serviciu pus în teren și pentru numărul de puncte câștigate pe prima mingă. Nu în ultimul rând, sunt afișate numărul total de victorii și înfrângeri alături de cele contra adversarilor de top 10 și durata medie a meciurilor jucătorului.

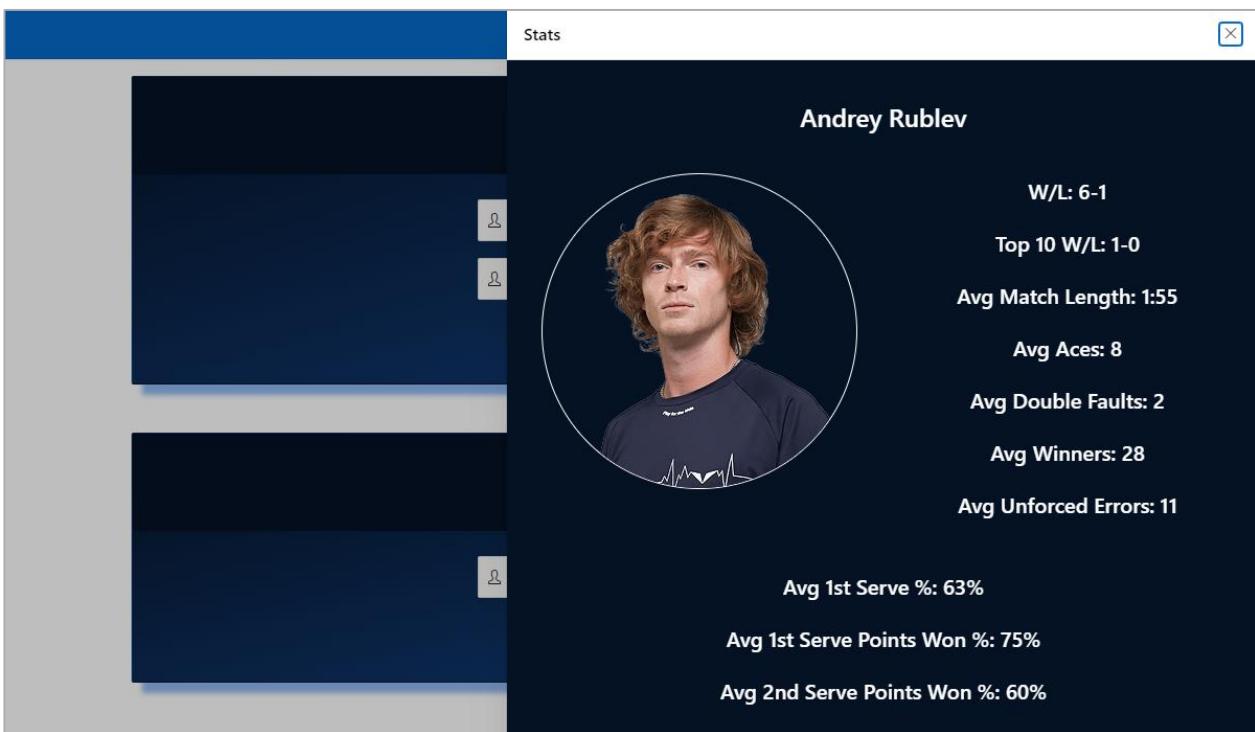


Figura IV.13 – Pagina statisticilor unui jucător

IV.5 Turnee

În secțiunea turnelor se regăsesc toate competițiile din sezonul curent ordonate în funcție de data de început. Acestea sunt grupate pe baza lunii în care se află data de început, pagina derulându-se automat la întrecerile din luna curentă. Informațiile afișate sunt: numele și categoria turneului, intervalul de timp, tipul de suprafață, orașul și țara în care au loc. Regiunile corespunzătoarele competițiilor din celelalte luni sunt ascunse, putând fi extinse ulterior de utilizatori pentru vizualizarea întregului program. De asemenea, din partea superioară a paginii se poate accesa un calendar ce ilustrează planificarea turnelor într-un mod structurat, marcându-se cu roșu competițiile încheiate, cu galben cele aflate în desfășurare și cu verde cele viitoare.

May 2024

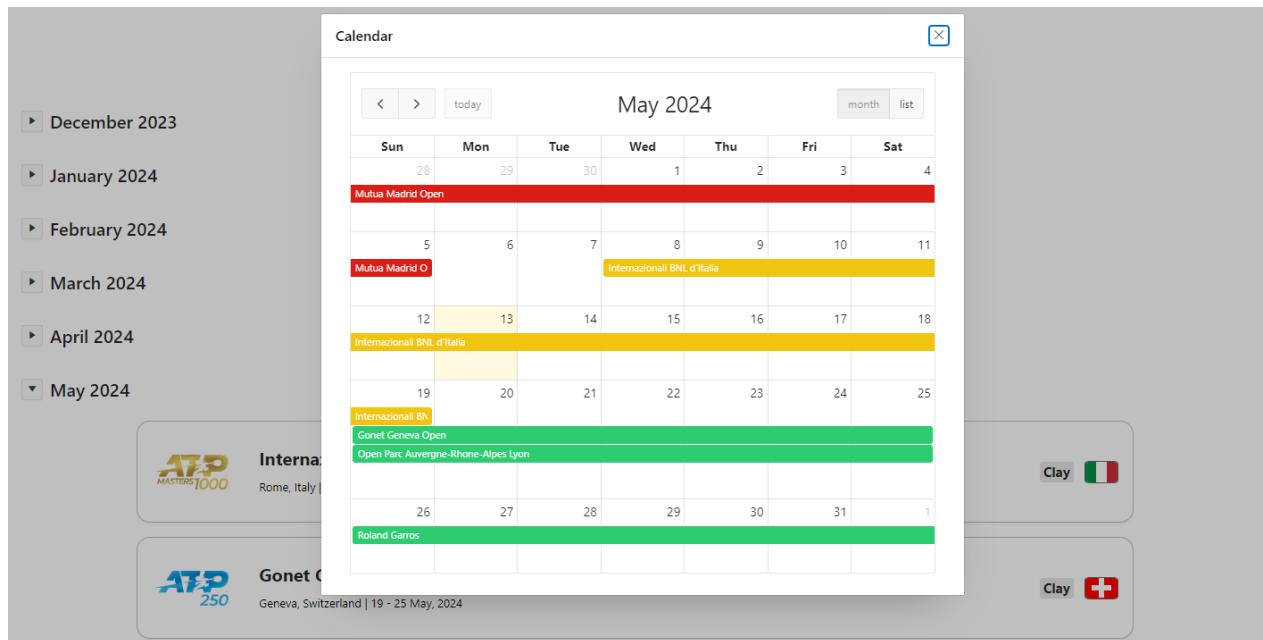
ATP Masters 1000 Internazionali BNL d'Italia
Rome, Italy | 8 - 19 May, 2024 Clay 

ATP 250 Gonet Geneva Open
Geneva, Switzerland | 19 - 25 May, 2024 Clay 

ATP 250 Open Parc Auvergne-Rhone-Alpes Lyon
Lyon, France | 19 - 25 May, 2024 Clay 

Roland Garros
Paris, France | 26 May - 9 June, 2024 Clay 

Figura IV.14 – Pagina programului turneelor



The screenshot shows a calendar for May 2024. The days of the week are labeled from Sunday to Saturday. Specific dates are highlighted in colored boxes corresponding to tournament weeks:

- May 1-4: Mutua Madrid Open (Red)
- May 5-11: Mutua Madrid Open (Red)
- May 8-12: Internazionali BNL d'Italia (Yellow)
- May 12-18: Internazionali BNL d'Italia (Yellow)
- May 19-25: Gonet Geneva Open (Green)
- May 20-26: Open Parc Auvergne-Rhone-Alpes Lyon (Green)
- May 26-June 9: Roland Garros (Green)

On the left sidebar, a list of months from December 2023 to May 2024 is shown. Below the calendar, there are two cards for the tournaments in May:

- Internazionali BNL d'Italia** Rome, Italy | 8 - 19 May, 2024
- Gonet Geneva Open** Geneva, Switzerland | 19 - 25 May, 2024

Each card includes the ATP logo, the tournament name, location, dates, surface type (Clay), and the flag of the host country.

Figura IV.15 – Pagina calendarului turneelor

La apăsarea unui turneu utilizatorii vor fi redirecționați către pagina cu detalii despre acesta ce conține, pe lângă informațiile deja prezentate, dimensiunea tabloului de concurs și poziția exactă pe glob a complexului sportiv. Apăsarea marcatorului de pe hartă deschide locația turneului în Google Maps.

În cadrul acestei pagini pot exista, în funcție de statusul turneului, anumite butoane prin intermediul căroroare sunt afișate date suplimentare despre jucătorii participanți și meciuri. Primul

link poate fi folosit de utilizatori pentru a afla detalii referitoare la biletele disponibile și prețurile acestora. Butonul *Tickets* dispare odată cu încheierea competiției respective.

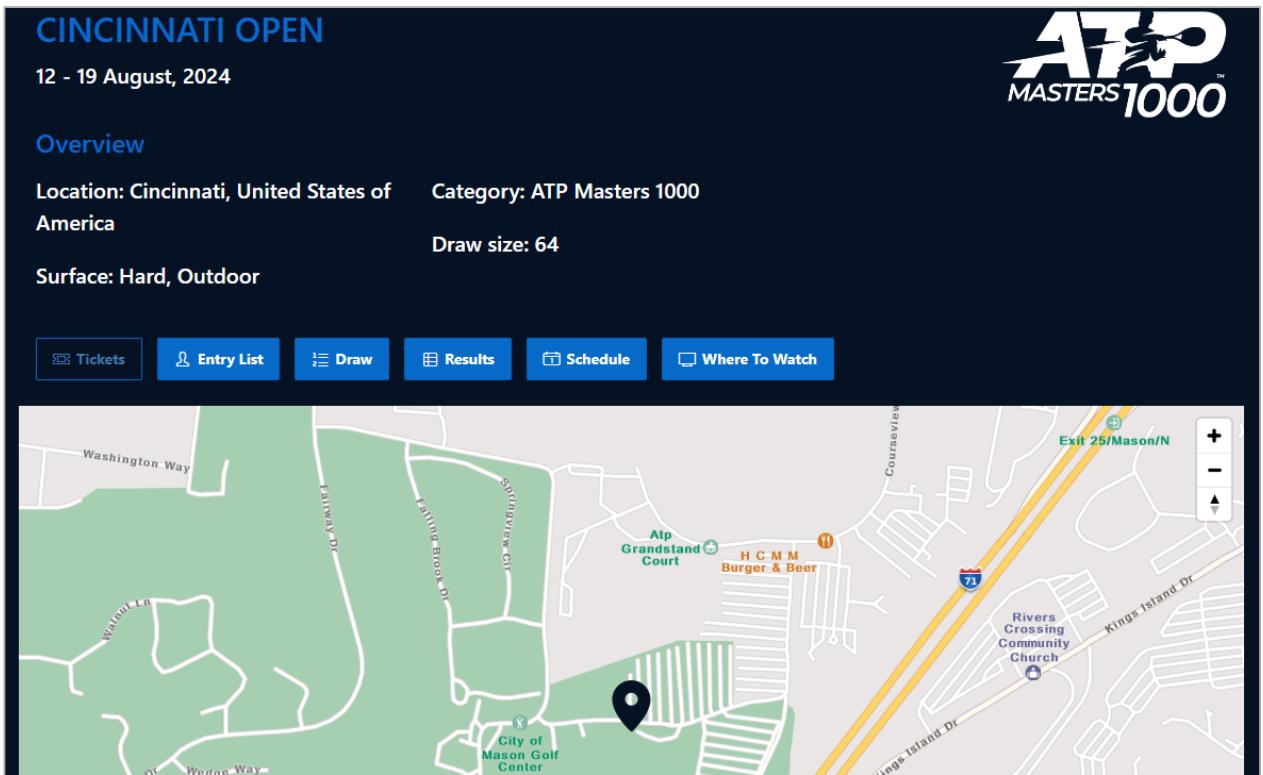


Figura IV.16 – Pagina cu detaliile unui turneu

Cu câteva săptămâni înainte de începerea unui turneu va fi afișată lista cu sportivii参
icienți ce conține clasamentul cu care au intrat în concurs și capii de serie. Aceasta poate fi accesată de utilizatori prin intermediul butonului *Entry List* și după terminarea competiției.

Odată cu realizarea tragerii la sorti, fanii pot urmări evoluția tabloului ce va fi actualizată de către utilizatorii cu rolul de EDITOR la finalul fiecărei zile competiționale. Pagina accesată prin intermediul butonului *Draw* conține informațiile generale legate de turneu, meciurile desfășurate în fiecare tur și scorurile acestora. Utilizatorii pot, de asemenea, reveni la tabloul de concurs și după finalizarea turneului pentru a analiza cu ușurință anumite rezultate.

Butonul *Results* face trimitere către o pagină ce prezintă informații despre partidele încheiate. În partea superioară există o listă în care sunt enumerate toate zilele competiției până la data curentă, iar accesarea uneia dintre intrări determină afișarea detaliilor precum câștigătorul, scorul, durata și arbitrul de scaun al meciurilor ce au avut loc. Similar cu cele două pagini descrise anterior, aceasta rămâne disponibilă pentru fiecare turneu până la finalul sezonului pentru a servi utilizatorilor interesați de performanțele de pe parcursul anului.

Seed	Name	Entry Ranking
1	Novak Djokovic	1
2	Carlos Alcaraz	2
3	Daniil Medvedev	3
4	Stefanos Tsitsipas	4
5	Casper Ruud	5
6	Holger Rune	6
7	Andrey Rublev	7
8	Jannik Sinner	8

Figura IV.17 – Pagina listei de participanți

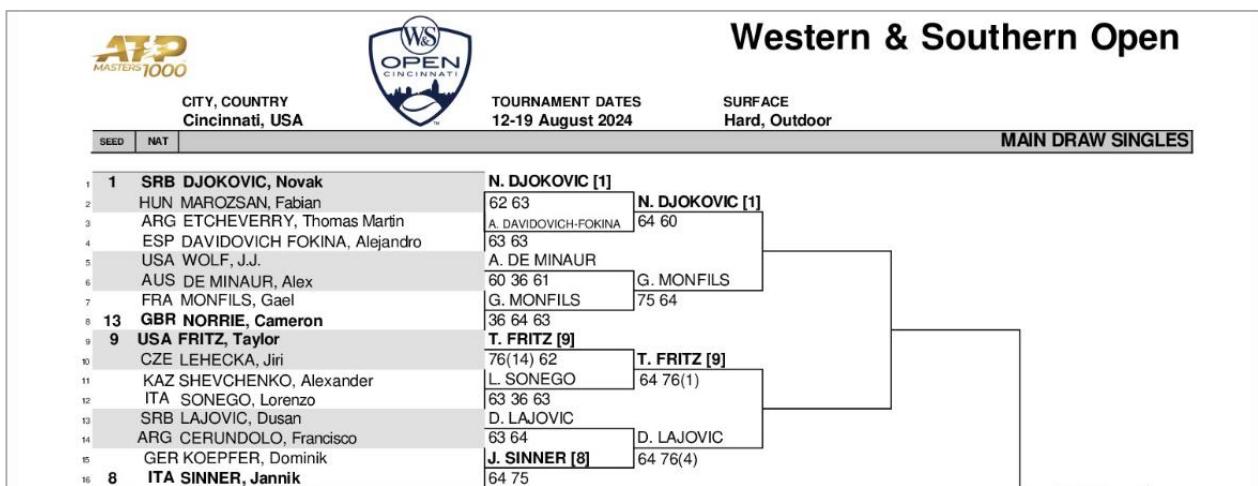


Figura IV.18 – Pagina tabloului de concurs

12-AUG-2024	13-AUG-2024	14-AUG-2024	15-AUG-2024
15-AUG-2024 - 2nd Round			
Hubert Hurkacz d. Borna Coric (15) Duration: 2:15 Umpire: Juan Zhang			5-7 6-3 6-3
Alexei Popyrin d. Nicolas Jarry Duration: 1:29 Umpire: Fergus Murphy			6-3 6-3

Figura IV.19 – Pagina rezultatelor meciurilor încheiate

Pentru vizualizarea programului de concurs utilizatorii pot accesa doar pe parcursul turneului pagina *Schedule*. Aceasta conține adversarii, turul, terenul și ora de început a meciurilor din ziua curentă și din ziua următoare odată ce acestea sunt stabilite. Datorită faptului că fanii tenisului se află în toate colțurile lumii, va fi afișată atât ora locală a partidelor, cât și cea corespunzătoare fusului orar al locației din care este folosită aplicația.

Nu în ultimul rând, în cadrul ferestrei modale apărute la apăsarea butonului *Where To Watch*, utilizatorii pot afla ce posturi de televiziune transmit fiecare turneu. În funcție de tipul de contract, anumite programe televizează toate meciurile, în timp ce altele difuzează doar confruntările de pe terenul central. [5]

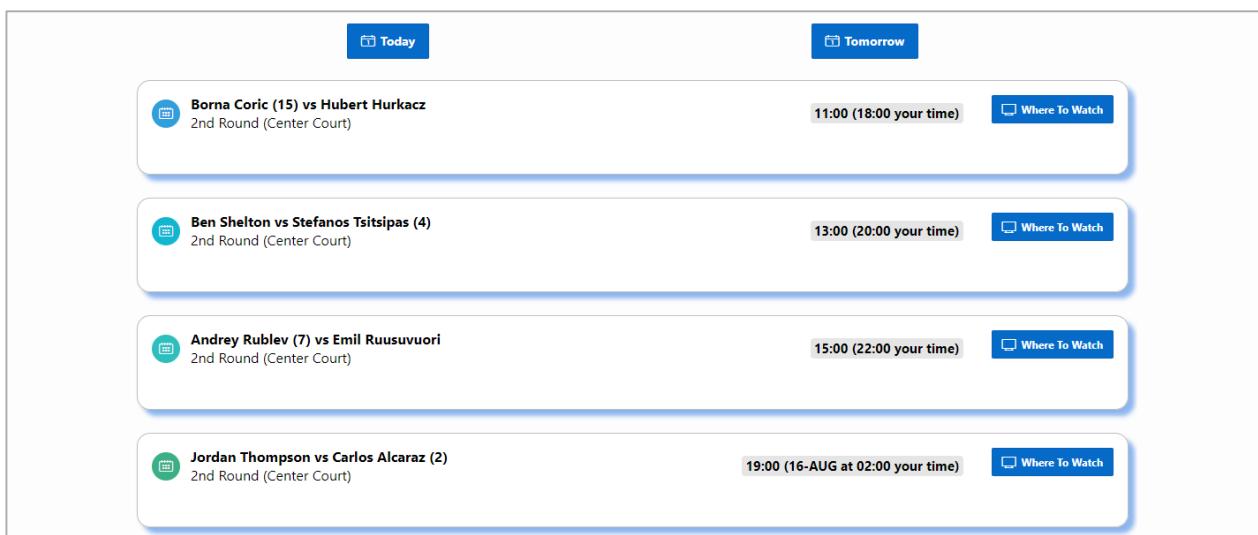


Figura IV.20 – Pagina programului meciurilor

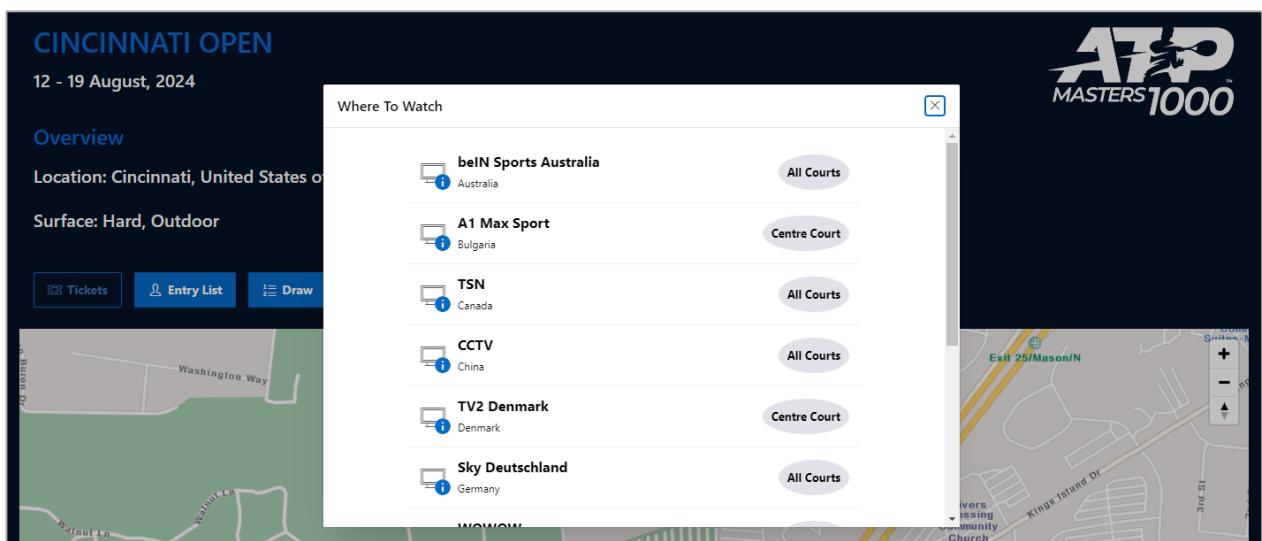


Figura IV.21 – Pagina programelor TV ce transmit turneul

IV.6 Planificarea călătoriilor

Spre deosebire de produsele similare de pe piață, aplicația web ATP Tour prezentată în această lucrare dispune de un modul destinat asistării fanilor în organizarea viitoarelor participări la turnee. Pagina *Plan Trip* oferă posibilitatea filtrării competițiilor în funcție de categorie, perioada de timp în care se desfășoară și dacă au loc într-un oraș capitală sau nu. Astfel de opțiuni ajută utilizatorii în găsirea tipurilor de competiții de care sunt interesați și care se intersectează cu intervalele de timp disponibile ale acestora.

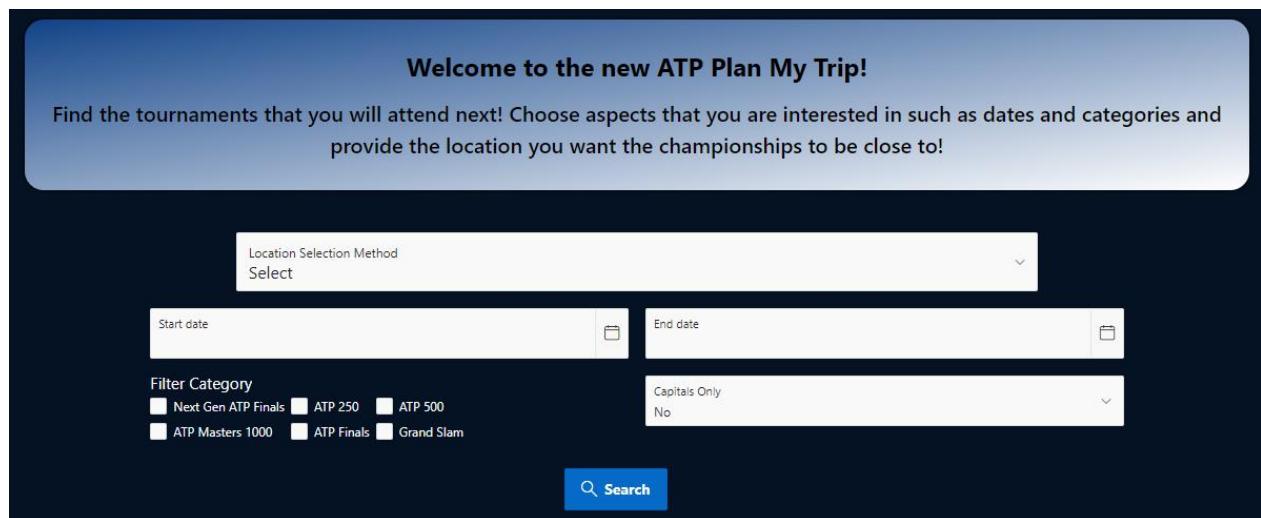


Figura IV.22 – Pagina planificării călătoriilor

Cea mai importantă caracteristică a acestei funcționalități este reprezentată de selectarea turneelor în funcție de proximitatea lor față de o anumită locație. Deoarece coloana *pozitie* a tabelului *LOCATIE* ce stochează coordonatele geografice ale complexurilor sportive este de tip SDO_GEOOMETRY, calculele distanțelor pe baza acesteia pot fi realizate într-un mod stabil și eficient. Utilizatorii pot alege ca rezultatele furnizate să se afle în cele mai apropiate x turnee față de o locație țintă, la o distanță de cel mult y kilometri de aceasta sau o combinație între cele două. În interogările SQL ce extrag informațiile corespunzătoare criteriilor introduse au fost utilizati operatorii SDO_NN, SDO_WITHIN_DISTANCE și funcția SDO_DISTANCE.

Locația față de care vor fi calculate distanțele competițiilor poate fi aleasă în trei moduri:

- preluarea locației curente a utilizatorului;
- selectarea manuală a locației pe hartă;
- introducerea numelui unui anumit oraș și a țării în care se află.

Acstea opțiuni ajută fanii atât în descoperirea turneelor celor mai apropiate de casă, cât și în situația în care au deja planificată o vacanță și doresc să afle dacă există turnee de tenis în desfășurare la care ar putea să participe.

Pentru accesarea locației curente, aplicația va cere în cadrul fiecărei sesiuni permisiunea utilizatorului. Acesta trebuie să aibă activată opțiunea de urmărire a locației pe dispozitivul personal pentru o acuratețe perfectă.

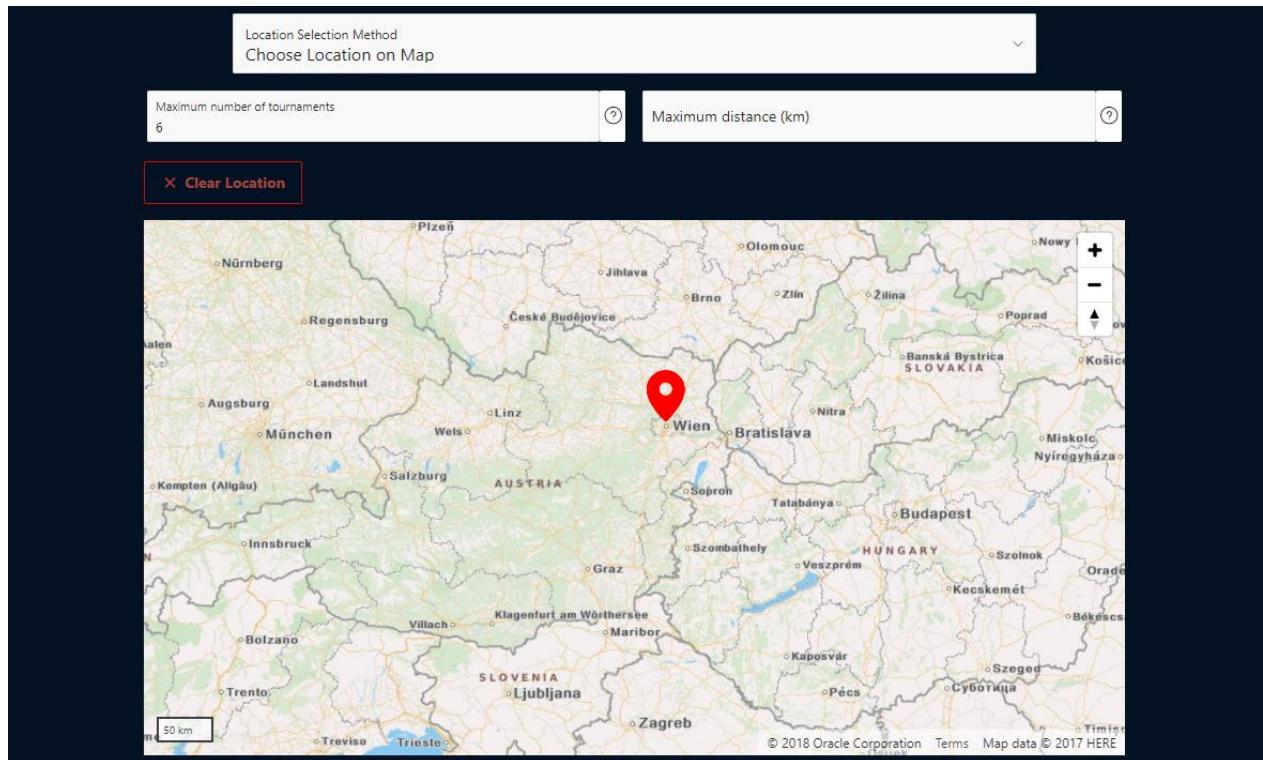


Figura IV.23 – Alegerea locației pe hartă

În cadrul metodei ce presupune furnizarea numelui locației căută este utilizat API-ul Geoapify ce oferă multiple facilități de lucru cu date geografice, printre care returnarea latitudinii și longitudinii centrului unui oraș. Pentru a putea accesa acest API este nevoie de crearea unui cont și generarea unei chei ce va fi folosită în cadrul fiecărei cereri. Deoarece există posibilitatea ca orașe cu același nume să se afle în state diferite, a fost aleasă varianta în care sunt menționate atât numele orașului, cât și cel al țării. De exemplu, apelul ce întoarce detalii despre orașul București are formatul: https://api.geoapify.com/v1/geocode/search?city=Bucharest&country=Romania&type=city&format=json&apiKey=CHEIE_API.

Pentru toate cele trei metode de alegere a locației coordonatele sunt utilizate în cadrul cererilor SQL, alături de celelalte criterii de selecție, pentru a regăsi turneele potrivite.

Rezultatele vor putea fi generate doar dacă cel puțin unul dintre câmpurile de filtrare este completat. În cazul în care o anumită locație este aleasă, utilizatorul este nevoit să completeze minim unul dintre cele două inputuri legate de distanța față de aceasta. În situația în care orașul introdus nu poate fi găsit de API sau serviciul nu funcționează, vor fi afișate mesaje de informare corespunzătoare.

Pagina unde sunt afișate rezultatele va prezenta informații precum numele competiției, perioada desfășurării, logo-ul categoriei și distanța dintre acesta și locația furnizată de utilizator. În plus, pentru turneele ce nu au avut loc până la data curentă va exista și un buton cu ajutorul căruia poate fi accesată pagina biletelor. De asemenea, fanii pot naviga către secțiunea cu detaliile turneeelor dând click pe denumirea acestora.

Tournament	Location	Distance
Erste Bank Open	Vienna, Austria	3.16 km
Generali Open	Kitzbuhel, Austria	308.07 km
BMW Open	Munich, Germany	353.2 km
Plava Laguna Croatia Open Umag	Umag, Croatia	373.54 km
BOSS OPEN	Stuttgart, Germany	535.91 km
Swiss Indoors Basel	Basel, Switzerland	657.94 km

Figura IV.24 – Pagina turneelor rezultate

IV.7 Bilete

Fanii pot achiziționa bilete la turnee urmând un proces ce presupune: adăugarea acestora în coșul de cumpărături, aplicarea unui discount (dacă este posibil) și plata prin intermediul cardului de credit.

Există trei categorii de bilete pe care utilizatorii le pot cumpăra:

- bilet de meci – permite accesul doar la partida respectivă;

- abonament de zi – permite accesul la toate meciurile desfășurate în turneu într-o anumită zi, indiferent de terenul pe care au loc;
- abonament de turneu – permite accesul pe toată perioada turneului.

Fiecare meci dintr-un anumit tur al competițiilor are asociat același preț. Utilizatorii pot economisi bani prin achiziționarea biletelor de zi în locul plății fiecărui meci în parte, abonamentul de turneu fiind cel mai profitabil. Pagina biletelor este împărțită în două secțiuni dedicate partidelor individuale, respectiv abonamentelelor.

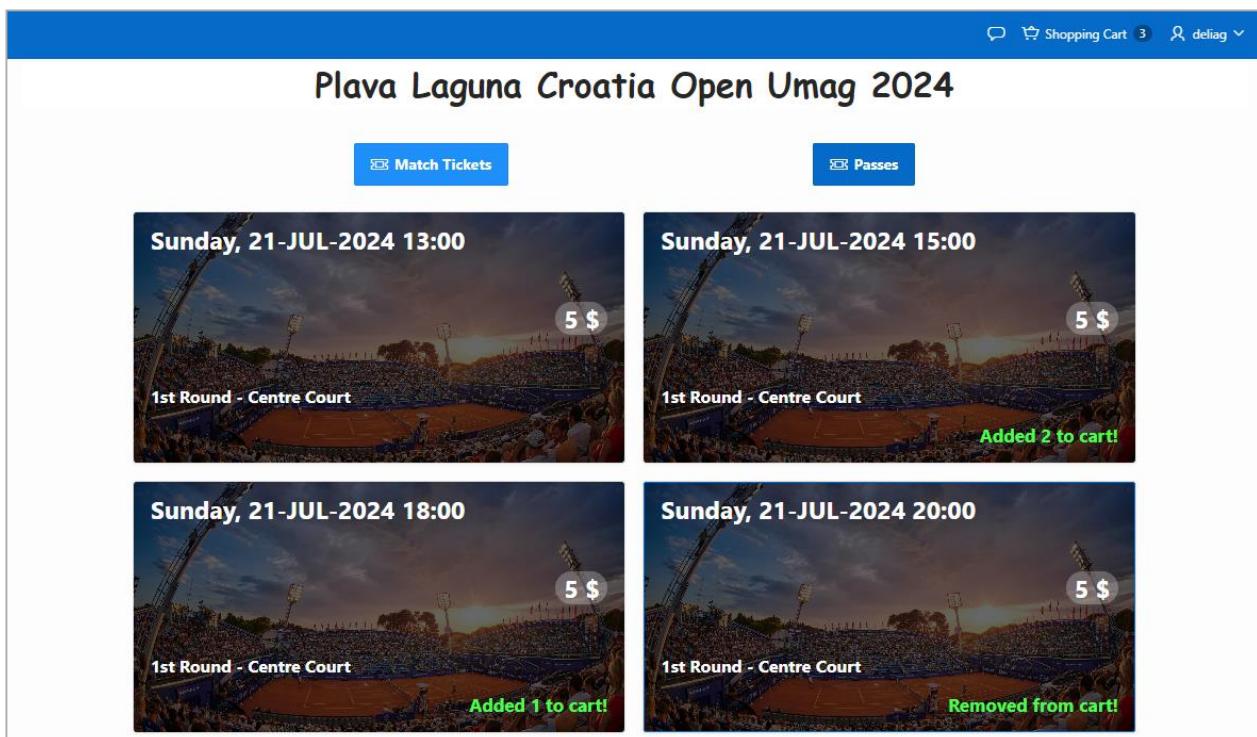


Figura IV.25 – Pagina biletelor unui turneu

În situația în care anumite tipuri de bilete au fost epuizate, acest lucru va fi indicat printr-un mesaj de tipul „Stoc epuizat” în dreptul acestuia. În caz contrar, utilizatorii autentificați pot accesa fereastra modală de adăugare în coș prin intermediul căreia pot fi selectate până la 9 bilete de același tip (limita maximă poate fi schimbată de administratori). Cantitatea acestora poate fi modificată ulterior, fanii având inclusiv opțiunea de a renunța la o parte din ele. [5]

Datorită elementelor globale SHOPPING_CART_ICON și SHOPPING_CART_ITEMS ce sunt actualizate la fiecare modificare a coșului de cumpărături, bara de navigare din partea superioară a ecranului reflectă starea și numărul total de bilete din coș. În plus, acțiunile realizate sunt afișate deasupra fiecărui bilet împreună cu actuala cantitate aleasă.

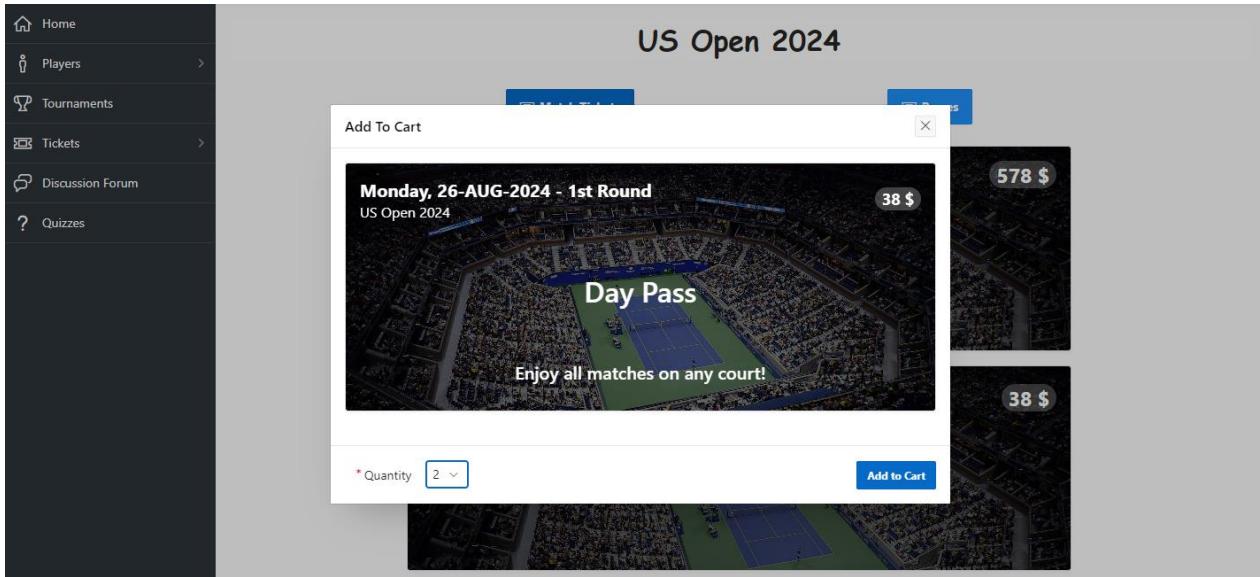


Figura IV.26 – Pagina de adăugare a biletelor în coș

Pentru reținerea biletelor alese este folosit tipul de date APEX_COLLECTION oferit de Oracle Application Express pentru stocarea temporară a informațiilor pe parcursul unei sesiuni. În cadrul aplicației va exista o colecție ce va conține:

- id-ul meciului și cantitatea (pentru biletele de meci);
- id-ul turneului și cantitatea (pentru abonamentele de turneu);
- id-ul turneului, ziua și cantitatea (pentru abonamentele zilnice).

Colecția va conține reunirea atributelor descrise mai sus (cantitatea, id-urile meciurilor și turneelor de tip NUMBER și ziua de tip DATE), coloanele ce nu corespund unui anumit tip de bilet primind valoarea *null*. Colecția este automat ștearsă în momentul în care utilizatorii se deconectează sau sesiunea curentă expiră. Toate operațiile cu ajutorul cărora se realizează managementul biletelor în aplicație sunt implementate ca subprograme într-un pachet PL/SQL. Comanda de adăugare a unui bilet în coș arată astfel:

```
apex_collection.add_member( p_collection_name => 'BILETE',
                             p_d001 => zi,
                             p_n001 => id_turneu,
                             p_n002 => id_meci,
                             p_n003 => cantitate );
```

Datorită acestei metode de memorare a obiectelor, utilizatorii pot achiziționa bilete la mai multe turnee în cadrul aceleiași comenzi, ușurând și accelerând procesul de cumpărare.

Pagina coșului de cumpărături conține suma totală a biletelor selectate și informații despre acestea, precum prețul unitar, cantitatea și subtotalul fiecărui. Utilizatorii își pot, de asemenea, edita comanda din cadrul acestei secțiuni prin modificarea numărului de bilete de un anumit tip sau golirea completă a coșului. [5]

The screenshot shows a shopping cart interface with the following details:

- Order Total: 633 \$**
- Match Tickets** section:
 - Sunday, 21-JUL-2024 15:00 - 1st Round, Centre Court: Plava Laguna Croatia Open Umag 2024. Quantity: 2, Unit Price: 5 \$, Subtotal: 10 \$. Edit button.
 - Sunday, 21-JUL-2024 18:00 - 1st Round, Centre Court: Plava Laguna Croatia Open Umag 2024. Quantity: 1, Unit Price: 5 \$, Subtotal: 5 \$. Edit button.
- Passes** section:
 - Monday, 26-AUG-2024 - 1st Round, US Open 2024, Day Pass. Quantity: 2, Unit Price: 38 \$, Subtotal: 76 \$. Edit button.
 - Rolex Paris Masters 2024, Tournament Pass. Quantity: 1, Unit Price: 542 \$, Subtotal: 542 \$. Edit button.
- Buttons: **Clear** (red) and **Proceed to Checkout →** (blue).

Figura IV.27 – Pagina coșului de cumpărături

La finalizarea comenzii, fanii trebuie să treacă printr-o serie de pași pentru a cumpăra cu succes biletele și a intra în posesia acestora.

În primul rând, utilizatorii pot beneficia de o reducere pe baza punctelor acumulate de-a lungul timpului. La crearea unui cont în aplicație, fiecare persoană începe cu zero puncte, având posibilitatea de a acumula 10% din suma plătită la orice comandă efectuată. Este nevoie de cel puțin 75 de puncte pentru a dispune de un discount de 20% la achiziția curentă, utilizatorii având posibilitatea de a decide dacă doresc sau nu să îl aplice.

Cel de-al doilea pas constă în completarea detaliilor cardului de credit. Acestea includ un număr valid din punct de vedere al algoritmului lui Luhn, o dată de expirare din viitor și codul CVV/CVC. Odată ce tranzacția a fost finalizată cu succes, toate biletele aflate în prezent în colecția APEX vor fi salvate în baza de date în tabelele corespunzătoare.

Utilizatorului îi va fi prezentat un rezumat al comenzii ce conține suma plătită, numărul total de bilete achiziționate și data curentă. Din această pagină sau din meniul general al aplicației poate fi deschisă pagina *My Tickets* unde se regăsesc toate biletele cumpărate alături de codurile QR pe baza cărora se realizează accesul la meciuri. [5]



Figura IV.28 – Pagina biletelor achiziționate

IV.8 Quiz-uri

Fanii au ocazia de a-și testa cunoștințele în cadrul competiției anuale de quiz-uri. În fiecare lună din sezon utilizatorii pot răspunde la câte cinci întrebări de tip grilă legate de diverse aspecte ale circuitului (turnee de Grand Slam, Jocurile Olimpice, ocupanții locului 1 mondial, etc.).

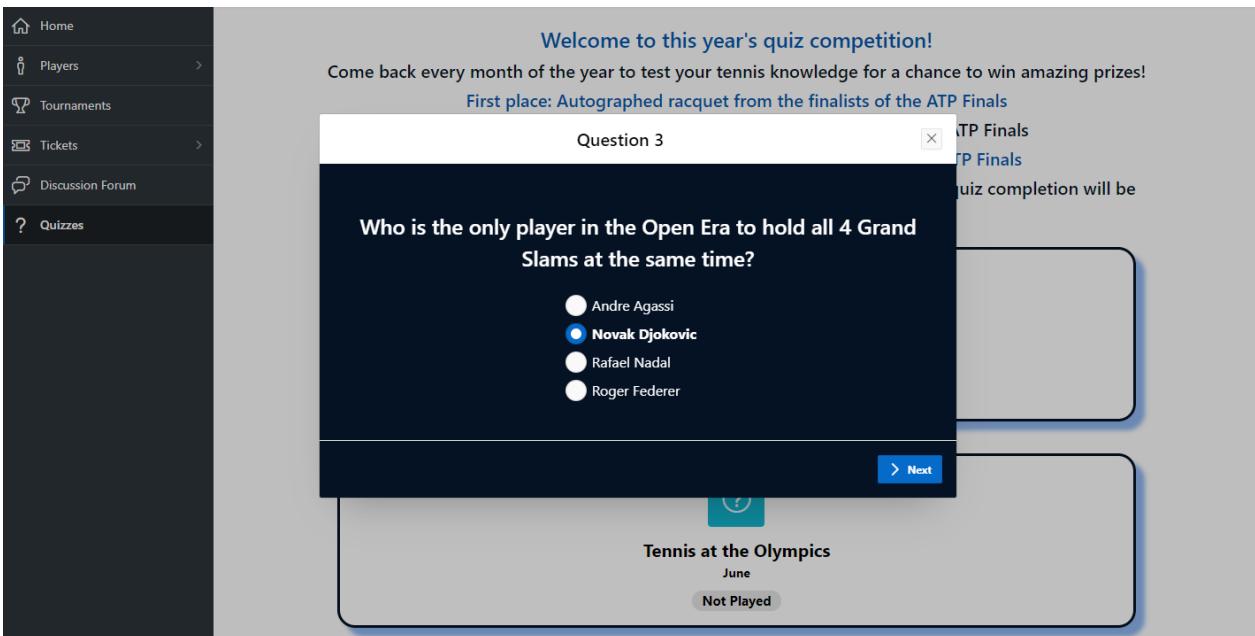


Figura IV.29 – Pagina întrebărilor

Pentru a asigura integritatea scorurilor finale, sunt reținute rezultatele intermediare astfel încât utilizatorii nu pot juca de mai multe ori același quiz. De asemenea, un test conține întrebări diferite pentru fiecare persoană, acestea fiind alese în mod aleatoriu din mulțimea totală de întrebări asociate subiectului. Datorită posibilității unor scoruri egale la sfârșitul anului, este luat în considerare și timpul de rezolvare al quiz-urilor pentru determinarea clasamentului final. [5]

IV.9 Forum de discuții

Pe forumul de discuții fanii pot citi și participa la conversații legate de rezultate și lumea tenisului. Utilizatorii pot adăuga, edita sau șterge subiectele și comentariile proprii, în timp ce editorii au permisiunea de a elimina orice postare considerată nepotrivită.

În pagina unde se regăsesc detaliiile unui subiect pot fi vizualizate comentariile acestuia afișate în ordine descrescătoare din punct de vedere al momentului postării. De asemenea, comentariile ce au fost modificate sunt marcate corespunzător, iar autorii acestora sunt afișați în partea stângă. [5]

The screenshot shows a social media post with the following details:

- Post Summary:** Novak Djokovic wins 24th Grand Slam Title!
The tennis legend extended his men's Grand Slam record by winning the 2023 US Open title and becoming the player (man or woman) with the most Slams in the Open Era.
- Author:** posted by ANDREW
- Comment Section:** A large input field labeled "Comment" with a "Post" button.
- Comments:**
 - PAUL** · 2m: I'm ready to say something I've always refrained from saying out of respect for Federer & Nadal: Novak Djokovic is the greatest tennis player of all time. Not just statistically. The greatest ever, period. For the first time, I can confidently say this won't change.
Edit
 - MA** · 6m: GOAT! 🏆
MARY · 6m
 - DE** · 8m: Novak is not only just the greatest tennis player, but the greatest athlete ever. The combination of technique, physicality, athleticism, endurance, mental toughness, tactical brilliance and longevity is simply mind blowing!

Figura IV.30 – Pagina comentariilor unui subiect

V DEZVOLTAREA APLICAȚIEI MOBILE

V.1 Descrierea aplicației

Orice sport de performanță are la bază valori precum integritatea și *fairplay*-ul. De aceea, se pune un accent important pe programele anti-doping care asigură faptul că jucătorii nu recurg la substanțe interzise ce le oferă un avantaj nedrept în fața celorlalți.

În tenis testele anti-doping pot avea loc atât în turnee, cât și în afara competiției, astfel că persoanele responsabile sunt nevoie să știe unde și când îi pot găsi pe sportivi. Pentru a rezolva această problemă, tenismenii folosesc o aplicație în care trebuie să aibă stabilite mereu o locație și o oră în care sunt disponibili pentru testare. Dacă într-o zi echipa anti-doping decide că recolteze probe de la un anumit jucător, însă acesta nu are informațiile trecute în aplicație sau nu poate fi găsit pe baza lor, atunci el va primi un avertisment. La 3 avertismente sportivul va fi suspendat din competiție timp de 18 luni.

Aplicația mobilă ITIA-App (The International Tennis Integrity Agency) dezvoltată în React Native se integrează cu aplicația web ATP Tour pentru a oferi atletilor și organizației ITIA o metodă accesibilă de gestionare a detaliilor de localizare pentru testele anti-doping.

V.2 Implementarea REST API-ului

Pentru ca aplicația mobilă să poată accesa baza de date Oracle deasupra căreia este construită aplicația web, a fost realizat un REST API ce implementează funcționalitățile necesare.

Fiecare *handler* din APEX RESTful Services preia datele prin intermediul uneia din următoarele metode:

- *Collection Query* – echivalent unei interogări SQL ce returnează un număr variabil de linii;
- *Collection Query Item* – echivalent unei interogări SQL ce returnează o singură înregistrare;
- *PL/SQL* – utilizat pentru prelucrarea complexă a datelor, fiind nevoie ca obiectul returnat să fie creat manual.

În cadrul modulului `atp.tour.service` sunt definite două şabloane: `isplayer` pentru autentificare și `profile/:id` pentru preluarea și actualizarea profilului de doping al unui jucător.

Primul dintre acestea are implementat un *handler* de tip GET ce primește prin parametrii *request*-ului un nume de utilizator și o parolă și returnează id-ul acestuia în cazul în care deține rolul de JUCĂTOR sau un mesaj de eroare în caz contrar. Codul PL/SQL ce efectuează operațiile menționate arată astfel:

```

DECLARE
    u_user_name users.user_name%TYPE;
    hash_parola VARCHAR2(1000);
    cod_user users.id_user%TYPE;
    nr NUMBER;

BEGIN
    u_user_name := UPPER(:nume_user);
    hash_parola := UPPER(:parola);

    SELECT id_user INTO cod_user
    FROM users
    WHERE user_name = u_user_name
    AND parola = hash_parola;

    SELECT COUNT(*) INTO nr
    FROM tenismen
    WHERE id_user = cod_user;

    IF nr = 0 THEN
        :mesaj := 'Not registered as a player! Please email
                    playersupport@atptour.com!';
    ELSE
        :cod := cod_user;
        :nume := u_user_name;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        :mesaj := 'Incorrect username or password!';
END;

```

Cel de-al doilea şablon conţine două *handler*-e: unul de tip GET ce returnează un *Collection Query Item* şi unul de tip PUT implementat prin intermediul unui bloc PL/SQL. *Handler*-ul de tip GET este utilizat pentru a obţine detalii despre profilul asociat utilizatorului cu id-ul transmis în URL, în timp ce *handler*-ul de tip PUT are rolul de a edita înregistrarea corespunzătoare profilului al cărui cod este dat în variabila id.

V.3 Autentificarea

În pagina de *Login* ce apare la deschiderea aplicaţiei ITIA-App jucătorii trebuie să introducă numele de utilizator şi parola aferente contului creat în aplicaţia web. Dacă aceste credenţiale sunt invalide sau aparţin unui utilizator ce nu deţine rolul de JUCĂTOR, atunci va fi afişat un mesaj de eroare corespunzător.

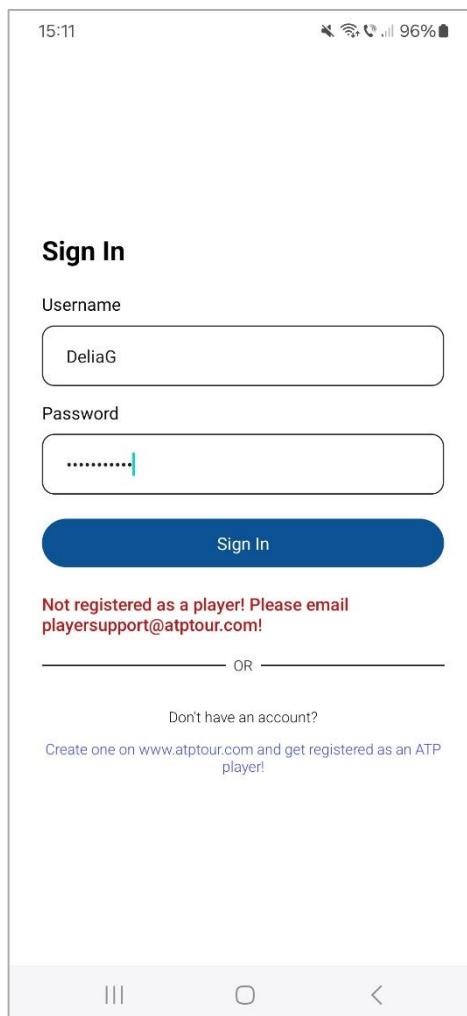


Figura V.1 – Pagina de autentificare

V.4 Ecranele aplicației

Dacă autentificarea a fost realizată cu succes, codul și numele user-ului vor fi reținute în contextul aplicației pe parcursul unei sesiuni de utilizare, iar jucătorul va fi redirecționat către pagina principală. Aceasta conține un videoclip de prezentare a circuitului ATP și patru butoane. Primul dintre acestea poate fi utilizat pentru modificarea paletelor de culori folosite în interfață grafică. Implicit, schema de culori este albastră, însă jucătorii o pot schimba cu una dintre temele roz, verde, mov sau roșu. Alegerea va fi, de asemenea, reținută în contextul aplicației, iar fiecare buton, detaliu sau iconă din UI va utiliza una dintre cele trei nuanțe predefinite ale culorii selectate.

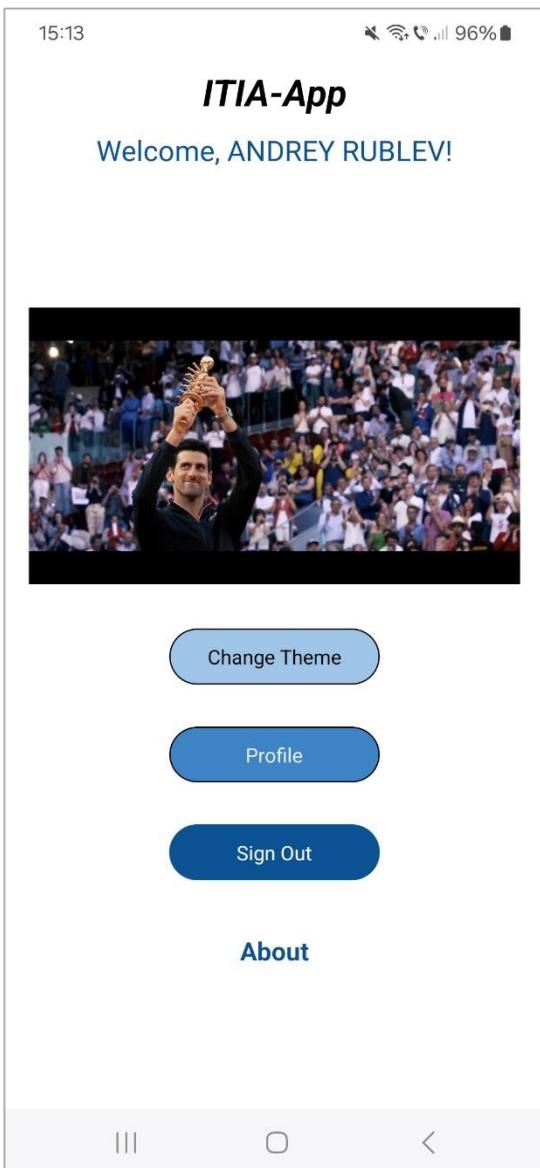


Figura V.2 – Pagina principală

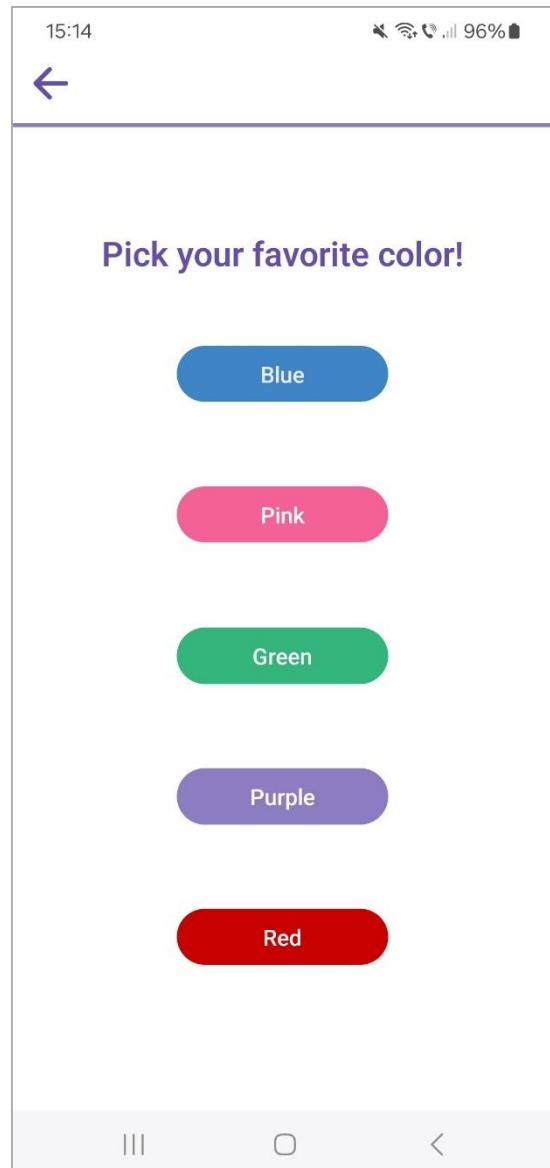


Figura V.3 – Pagina de schimbare a temei

Butonul *About* poziționat în partea de jos a ecranului principal duce către o pagină de informare unde jucătorii pot afla detalii despre modul de funcționare al aplicației și al programului anti-doping.

Pentru a se deconecta, utilizatorii pot da click pe butonul de *Sign Out* ce îi trimită înapoi la pagina de autentificare. Din motive de securitate, jucătorii sunt, de asemenea, deconectați automat la închiderea aplicației.

Nu în ultimul rând, profilul curent poate fi accesat prin intermediul butonului *Profile*.

V.5 Profilul jucătorilor

Atunci când un administrator atribuie unui utilizator rolul de JUCĂTOR, se creează automat un profil de doping asociat acestuia. La început profilul nu conține nicio informație, fiind nevoie ca sportivul să îl completeze din aplicația mobilă. Detaliile ce trebuie introduse sunt: locația, ora și, opțional, informații suplimentare despre adresa la care jucătorul poate fi găsit (numărul blocului, apartamentului, camerei de hotel, etc.). Este datoria tenismenilor să se asigure că datele furnizate sunt suficient de precise pentru ca echipa de testare să îl localizeze în mod exact. Fereastra zilnică în care jucătorii pot fi căutați este fixată la o oră. Astfel, aceștia sunt nevoiți să aleagă doar ora de început a intervalului de șaizeci de minute.

V.5.1 Google Places API

Pentru a putea extrage coordonatele locațiilor jucătorilor, a asigura corectitudinea datelor introduse de aceștia și a îi ajuta în completarea profilurilor, este folosit serviciul Places API oferit de furnizorul de *cloud* GCP (Google Cloud Platform). Aceasta se bazează pe Google Maps și pune la dispoziție detalii a peste 100 de milioane de adrese și puncte de interes.

Serviciul este integrat în aplicația mobilă prin intermediul bibliotecii *react-native-google-places-autocomplete* ce oferă, de asemenea, facilitatea de completare automată a locațiilor pe baza inputului utilizatorului și a rezultatelor returnate de API. Pentru ca acesta să funcționeze, este nevoie de o cheie generată în *cloud* corespunzătoare colecției de produse Google Maps Platform unde se regăsește și serviciul Places API. Odată ce jucătorul selectează una dintre locațiile sugerate de textul introdus, vor fi reținute temporar coordonatele și numele adresei. Modul de utilizare al acestei componente în React Native este:

```
<GooglePlacesAutocomplete
    styles={{
        container: { flex: 0 },
        textInput: {
            fontSize: 16,
            borderWidth: 1,
            borderRadius: 10,
            borderColor: colors.medium,
            marginTop: 8,
            marginBottom: 0,
            paddingVertical: 2,
        },
    }}
    nearbyPlacesAPI="GooglePlacesSearch"
    debounce={400}
    placeholder={profil?.adresa || ""}
   textInputProps={{

        editable: isUpdating,
    }}
    minLength={2}
    onPress={(data, details = null) => {
        setProfil({
            ...profil,
            adresa: data.description,
            lat: details?.geometry.location.lat,
            lng: details?.geometry.location.lng,
        });
    }}
    fetchDetails={true}
    query={{

        key: Constants.expoConfig?.extra?.apiKey,
        language: "en",
    }}
/>
```

Astfel, de fiecare dată când trec 400 de milisecunde de la ultimul caracter tastat, API-ul va genera cele mai relevante cinci locații. În plus, atunci când utilizatorul apasă pe una dintre opțiuni, va fi afișată o hartă ce prezintă poziția sa pe glob în Google Maps pentru dispozitivele Android sau în Apple Maps pentru iOS.

Pentru configurarea intervalului orar, jucătorii au la dispoziție un *Time Picker* cu ajutorul căruia pot insera sau alege ora și minutele corespunzătoare momentului de început al acestuia.

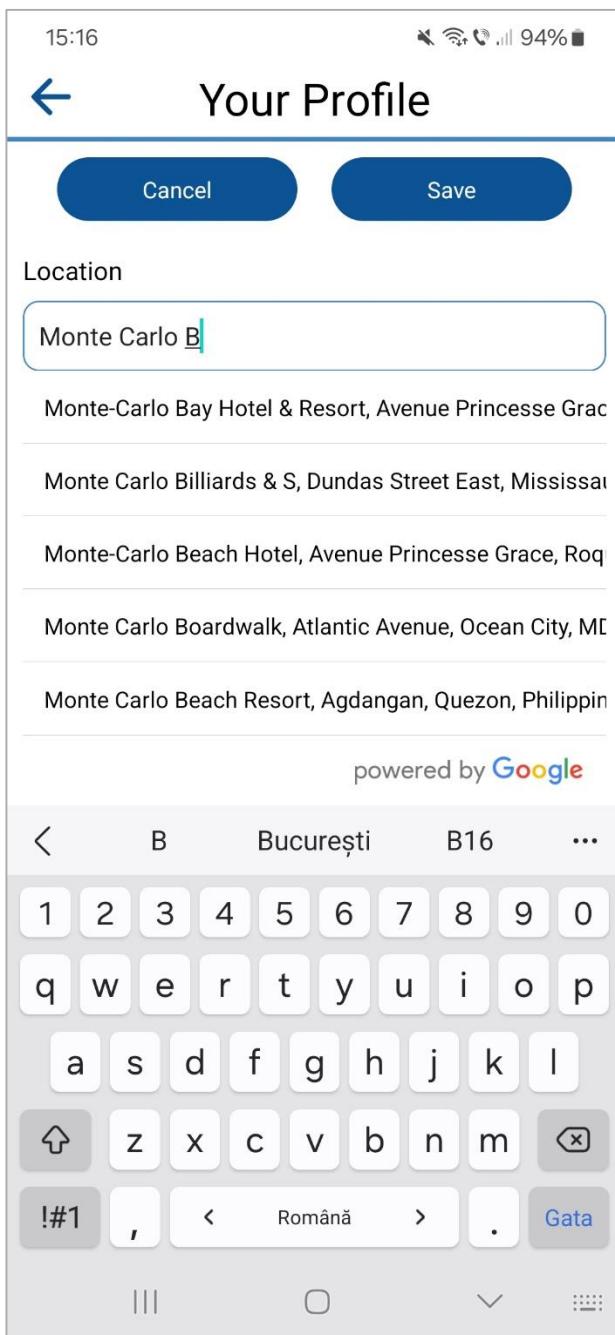


Figura V.4 – Funcționalitatea de completare automată a locației

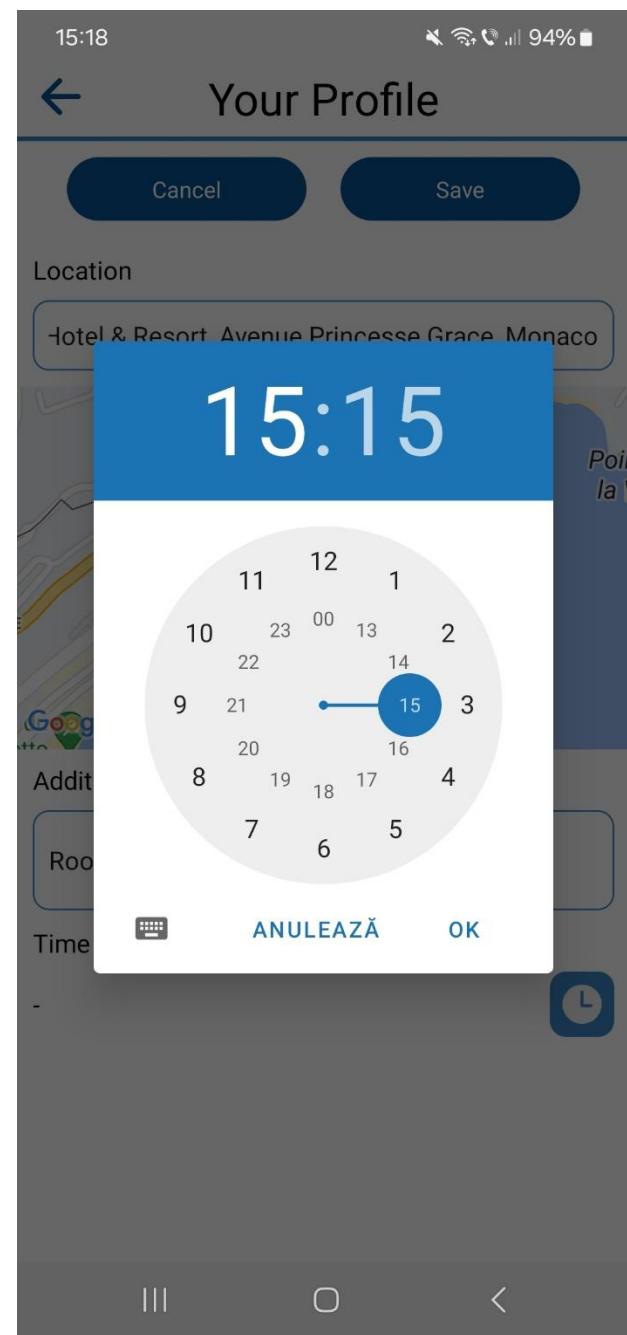


Figura V.5 – Selectarea intervalului orar

V.5.2 Actualizarea profilului

În pagina profilului jucătorii își pot vizualiza detaliile actuale și le pot edita apăsând pe butonul de *Update*. Orice schimbare va fi reținută într-o variabilă de stare și va fi afișată în interfață grafică, însă propagarea la nivelul bazei de date va fi realizată doar atunci când utilizatorul apasă pe butonul de *Save*. Astfel, există opțiunea de anulare a modificărilor, revenindu-se la informațiile precedente ale profilului.

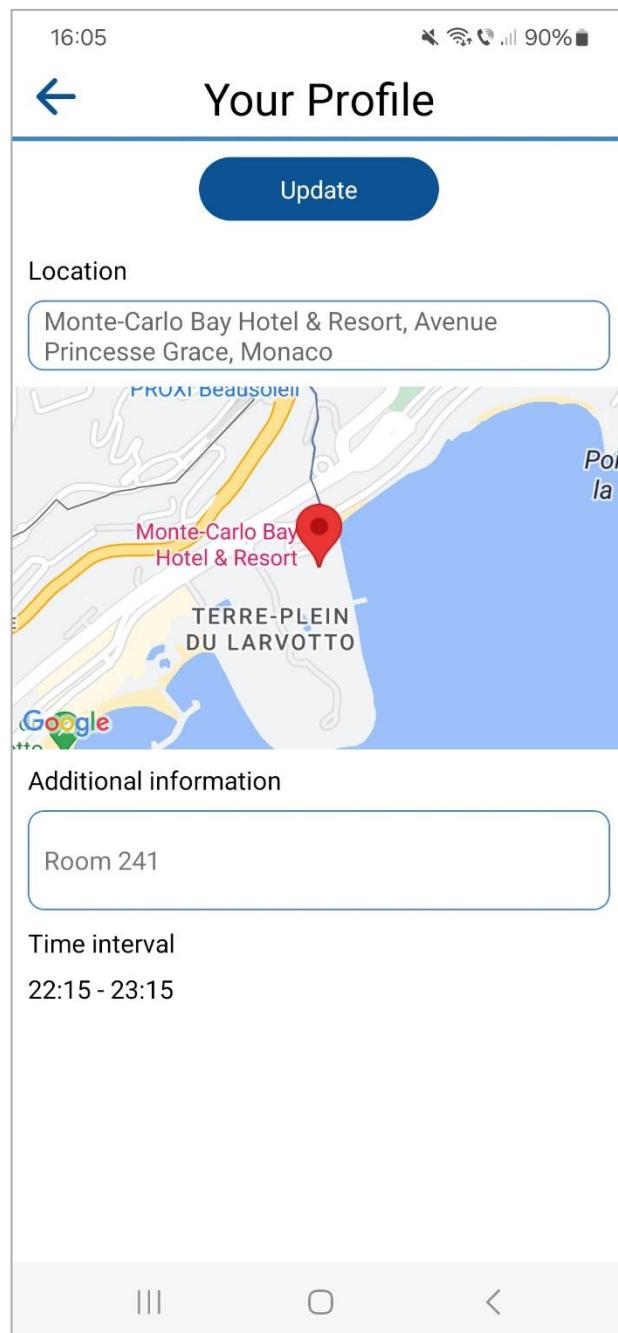


Figura V.6 – Ecranul profilului

Pentru salvarea în baza de date este folosit *handler*-ul de tip PUT definit în cadrul serviciului RESTful din Oracle APEX. Fetch API este o funcționalitate pusă la dispoziție de React Native ce permite interacțiunea cu servicii externe prin intermediul *request*-urilor URL. Apelul efectuat în momentul în care jucătorii permanentizează modificările realizate este:

```
const response = await fetch(
  `http://localhost:8088/ords/atp_tour/itiaservice/profile/${profil?.id}`
, {
  method: "PUT",
  headers: {
    Accept: "application/json",
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    adresa: profil?.adresa, latitudine: profil?.lat,
    longitudine: profil?.lng, informatii: profil?.informatii,
    ora_start: profil?.ora_start, minute_start: profil?.minute_start,
  }),
}
);
;
```

Administratorii pot vizualiza din aplicația web detaliile tuturor profilurilor și au opțiunea de a deschide în Google Maps fiecare locație setată pentru o mai ușoară gestionare a rutelor către acestea.

Player Name		Username	Adress	Latitude	Longitude	Additional Information	Interval
📍	Marton Fucsovics	MARTON	Budapest, Grand Budapest Hotel, József körút, Hungary	47.49581449999999	19.070375	Bl A2, Et 4, Ap 42	07:45 - 08:45
📍	Daniil Medvedev	DANIIL	Pullman Paris Tour Eiffel, Rue Jean Rey, Paris, France	48.8556488	2.2928584		21:00 - 22:00
📍	Andrey Rublev	ANDREY RUBLEV	Monte-Carlo Bay Hotel & Resort, Avenue Princesse Grace, Monaco	43.74881509999999	7.43872420000001	Room 241	22:15 - 23:15

Figura V.7 – Pagina de vizualizare a profilurilor pentru administratori

CONCLUZII

Aplicația web ATP Tour dezvoltată în această lucrare reușește să gestioneze cu succes componentele din care este alcătuit circuitul profesionist de tenis masculin. Informațiile despre jucători și performanțele acestora, sponsori, turnee, meciuri și programele de televiziune care le transmit sunt prezentate utilizatorilor într-un mod intuitiv și accesibil.

Procesul de achiziționare a biletelor este rapid și flexibil datorită opțiunii de a participa la mai multe partide cu ajutorul abonamentelor și a posibilității de a cumpăra bilete la diferite competiții în cadrul aceleiași comenzi. De asemenea, interacțiunea dintre utilizatori și promovarea sportului pentru publicul Tânăr sunt asigurate prin intermediul forumului de discuții și al quiz-urilor.

Modulul de planificare a călătoriilor pe baza caracteristicilor turneelor și a locațiilor acestora reprezintă o funcționalitate nouă față de aplicațiile similare, fiind menită să ajute fanii în găsirea celor mai apropiate întreceri de care sunt interesați.

Aplicația mobilă ITIA-App, realizată în completarea aplicației web, ajută în organizarea eficientă a programului de anti-doping din tenis, oferind o metodă de centralizare și actualizare a profilurilor jucătorilor. Aceasta pune la dispoziție o interfață ușor de utilizat de către sportivi pentru selectarea adresei și a intervalului orar corespunzător, modificările fiind propagate în baza de date prin intermediul unui REST API.

Cele două aplicații rezultate demonstrează faptul că tehnologiile Oracle Application Express și React Native sunt potrivite pentru dezvoltarea produselor web, respectiv mobile, și permit interoperabilitatea dintre acestea. De asemenea, baza de date Oracle 19c folosită pentru stocarea tuturor datelor, inclusiv a celor geospațiale, se dovedește a fi un instrument puternic și adecvat pentru problema propusă.

Aplicația web poate fi îmbunătățită în dezvoltările ulterioare prin adăugarea unui sistem de tip Data Warehouse dedicat vânzărilor de bilete. Astfel, vor putea fi utilizate instrumente de Business Intelligence pentru analiza și raportarea tendințelor de cumpărare în scopul elaborării unui plan de sporire a profiturilor. În viitor, poate fi, de asemenea, integrati algoritmi de învățare automată care să estimate probabilitatea de câștig a jucătorilor pe baza statisticilor înregistrate în turneu sau în competițiile anterioare. O altă funcționalitate poate fi reprezentată de existența unui modul al aplicației ITIA-App destinat personalului din programul anti-doping care să gestioneze informațiile despre probele recoltate și rezultatele acestora. Detaliile testelor ar putea fi apoi trimise și vizualizate de sportivi prin intermediul aplicației mobile prezentate în lucrare.

BIBLIOGRAFIE

- [1] Bierman, G., Abadi, M., & Torgersen, M. (2014). Understanding typescript. *ECOOP 2014 -Object-Oriented Programming: 28th European Conference, Uppsala, Sweden, July 28-August 1, 2014. Proceedings* 28 (pg. 257-281). Springer.
- [2] Budziński, M. (2024). *What Is React Native? Complex Guide for 2024*. Preluat de pe Netguru: <https://www.netguru.com/glossary/react-native>
- [3] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM* (Vol. 13, pg. 377-387). ACM New York, NY, USA.
- [4] Galiano, E. (2022, 12 Septembrie). *Oracle Database 19c - Cool Features*. Preluat de pe LiveLabs Blog: <https://blogs.oracle.com/livelabs/post/oracle-database-19c---cool-features>
- [5] Gherghe I. D. (2022). *Aplicație web pentru gestiunea unui turneu de tenis*
- [6] Godfrey, J. (2021, Decembrie). Oracle APEX SQL Workshop Guide, Release 21.2. Oracle and/or its affiliates. <https://docs.oracle.com/en/database/oracle/application-express/21.2/aeutl/oracle-apex-sql-workshop-guide.pdf>
- [7] Jennings, T. (2022, Februarie). Oracle APEX App Builder User's Guide, Release 21.2. Oracle and/or its affiliates. <https://docs.oracle.com/en/database/oracle/application-express/21.2/htmldb/oracle-apex-app-builder-users-guide.pdf>
- [8] Kothuri, R., & Ravada, S. (2008). Oracle spatial, geometries. *Encyclopedia of GIS*, 821-826.
- [9] Vanapalli, M. (2022, 19 Ianuarie). *5 Advantages of Upgrading from Oracle Database 12c to 19c*. Preluat de pe Datavail: <https://www.datavail.com/blog/5-advantages-of-upgrading-from-oracle-database-12c-to-19c/>
- [10] *Introduction to Oracle Database*. (Oracle)
<https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/introduction-to-oracle-database.html> (accesat la 25 Aprilie 2024)
- [11] *Expo CLI - Expo Documentation*. <https://docs.expo.dev/more/expo-cli/> (accesat la 9 Mai 2024)
- [12] *React Native Documentation*. <https://reactnative.dev/> (accesat la 8 Mai 2024)
- [13] *Spatial Concepts*. (Oracle) Developer's Guide:
<https://docs.oracle.com/en/database/oracle/oracle-database/19/spatl/spatial-concepts.html> (accesat la 29 Aprilie 2024)

- [14] *Spatial Data Types and Metadata.* (Oracle) Developer's Guide:
<https://docs.oracle.com/en/database/oracle/oracle-database/19/spatl/spatial-datatypes-metadata.html> (accesat la 29 Aprilie 2024)
- [15] *Spatial Operators.* (Oracle) Developer's Guide:
<https://docs.oracle.com/en/database/oracle/oracle-database/19/spatl/spatial-operators-reference.html> (accesat la 30 Aprilie 2024)
- [16] *What is a Relational Database (RDBMS)?* (Oracle)
<https://www.oracle.com/database/what-is-a-relational-database/> (accesat la 24 Aprilie 2024)
- [17] *What is a relational database?* (IBM) <https://www.ibm.com/topics/relational-databases> (accesat la 23 Aprilie 2024)