



**ACADEMIA DE STUDII ECONOMICE BUCUREȘTI  
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ**

**LUCRARE DE LICENȚĂ**  
**DEZVOLTAREA UNEI PLATFORME DE SCOUTING**  
**PENTRU IDENTIFICAREA ȘI EVALUAREA**  
**JUCĂTORILOR DE FOTBAL**

**Coordonator științific:**

**Conf. Univ. Dr. MIHAELA MUNTEAN**

**Absolvent:**

**MITU ADRIAN-CONSTANTIN**

**București**

**2024**

## Cuprins

<b>Introducere.....</b>	<b>3</b>
<b>Capitolul 1 - Stadiul cunoașterii în domeniu .....</b>	<b>5</b>
<b>Capitolul 2 - Etapa de analiză .....</b>	<b>9</b>
<b>Capitolul 3 - Etapa de proiectare.....</b>	<b>20</b>
<b>Capitolul 4 - Etapa de implementare.....</b>	<b>27</b>
<b>Capitolul 5 - Prezentarea părții aplicative.....</b>	<b>44</b>
<b>Concluzie .....</b>	<b>50</b>
<b>Bibliografie.....</b>	<b>52</b>

## Introducere

În prezent, sportul este o parte foarte importantă a culturii oricărei societăți, din mai multe motive. Oriunde ne-am uita pe glob, există o competiție oficială pentru aproape oricare din sporturi. Pe de o parte, acesta poate fi o modalitate foarte bună de relaxare sau o sursă de divertisment. Orice om obișnuit a avut, cel mai probabil, o tangență cât de mică cu sportul sau chiar are și în prezent o activitate fizică în prim-plan. Încă de mici, afară sau la școală am fost puși în situația de realiza activități sportive, sub forma unei mici competiții în care am fost puși în echipe, care ne-au adus bucurie și plăcere. De asemenea, campionatele sau cupele de anvergură atrag milioane de spectatori, care au o pasiune fără margini pentru sportivii pe care îi admiră sau pentru cluburile pe care le susțin. Fie că aleg să rămână doar fani ai diverselor competiții sau iau parte chiar ei în diverse activități ce au în mijlocul lor efortul fizic ori, ideal, se bucură de ambele, sportul ne învață anumite lucruri într-o manieră inedită: să fim disciplinați, că succesul este rezultatul unei munci continue și a unei încercări permanente de a deveni o versiune mai bună a noastră, să lucrăm în echipă, să avem curaj și să nu renunțăm până la final să realizăm ce ne-am propus.

Pe de altă parte, în zilele de astăzi sportul a ajuns să reprezinte cariera multor persoane, fie că vorbim de jucători sau de toate persoanele care sunt implicate în a susține performanța unui atlet, cum ar fi antrenorii, preparatorii fizici, scouterii, analiștii ori medicii și psihologii specializați în aceste direcții, dar și managerii și persoanele aflate în funcții de conducere ale cluburilor sau federațiilor. Bineînțeles că în prim-plan sunt jucătorii, deoarece eforturile tuturor celor care iau parte la acest fenomen din postura de persoane implicate din punct de vedere profesional au ca scop îmbunătățirea și perfecționarea prestațiilor celor dintâi. Profesioniștii au făcut din sport să fie mult mai mult decât o pasiune, pentru că această activitate a devenit scopul lor principal, nevoia primară de a evolua, de a deveni mai bun și de a învinge adversarii fiind împlinită în acest mod, destul de diferit de ceea ce face în general omul de rând. Totuși, sportivii reprezintă pentru noi modele, prin capacitățile fizice pe care le au, dar și de tehnica de care dau dovadă, ne bucură prin performanțele lor și continuă să ne inspire, indiferent că rezultatul final este o victorie sau o înfrângere.

Cel mai cunoscut, mai practicat și mai iubit sport din zilele noastre este cu siguranță fotbalul. Chiar și o persoană care nu este atrasă de fotbal în mod special de acest sport are măcar o idee despre rezultatele echipei naționale din țara sa, a auzit de câteva cluburi de succes sau a auzit de câteva competiții importante. Jucătorii cei mai buni sunt, în general, persoane publice ce ajung să fie cunoscute în întreaga lume. Competițiile finale sunt evenimente pe care țările concurează în a le găzdui, deoarece acestea atrag sute de mii de turiști din toate locurile de pe glob. De aceea, Campionatul Mondial de Fotbal, organizat o dată la 4 ani, este cel mai urmărit eveniment din lume, atrăgând în prezent aproximativ 5 miliarde de spectatori.

În altă ordine de idei, un alt fenomen de anvergură care a luat avânt în ultimele câteva decenii este apariția și dezvoltarea cu rapiditate a domeniului IT. Prezența computerelor personale, a internetului sau a telefoanelor inteligente, precum și a întregii infrastructuri necesare folosirii dispozitivelor din această sferă au schimbat radical viața noastră de zi cu zi și posibilitățile de care dispunem. Lucruri care necesită un efort, resurse sau o cantitate de timp semnificative, cum ar fi

comunicarea la distanță sau partajarea de documente au devenit acțiuni pe care avem capacitatea să le facem la costuri aproape inexistente.

Tehnologia și-a făcut deja simțit impactul în lumea sportului, în diferite maniere. Astăzi ne putem bucura de luxul de a putea vedea mai toate competițiile la televizor sau, mai nou, pe internet. Putem urmări o sumedenie de statistici despre jucătorii și atleții noștri preferați. Regulile sporturilor pot fi acum respectate aproape la perfecție, corectând greșelile umane, cu ajutorul unor tehnologii ca Hawk-Eye, din tenis sau VAR, implementată recent în fotbal.

Astfel, motivul pentru care am ales să abordez o temă în această zonă pentru lucrarea mea de licență este unul destul de simplu: se află la intersecția a două dintre domeniile mele de interes: programarea, care este în prezent în centrul activităților mele academice și profesionale și fotbalul, care este o pasiune pe care o am de mic și de care sunt atașat și în prezent. Prin dezvoltarea unei aplicații de scouting pentru identificarea și evaluarea jucătorilor de fotbal, îmi doresc să pot pune în lumină o modalitate de a face selectarea celor mai buni sportivi mai facilă și mai eficientă.

În primul rând, capacitatea de a determina ce jucători sunt cei mai potriviți pentru a fi cumpărați într-o echipă de fotbal este, dacă nu cel mai important, cel puțin unul din cele mai relevante aspecte ale lumii “sportului rege”. Până la urmă, deși necesită multă disciplină și perseverență, fotbalul are în prim-plan talentul unui jucător, care este cultivat prin numeroase antrenamente și meciuri. De aceea, soluția pe care doresc să o prezint ajută ambele părți implicate în acest proces complex și dinamic.

Pe de o parte, aplicația va pune la dispoziția scouterilor anumite date ce îi vor ajuta să identifice și să evalueze mult mai facil și rapid abilitățile și performanțele unui jucător, iar avantajul este că o va putea face la distanță. Deci, standardul din ultimele decenii, acela ca un scouter să meargă personal la meciurile unei echipe, acțiune consumatoare de timp și resurse, va putea fi făcută într-un mod mai eficient, la care se adaugă alte statistici menite să completeze imaginea de ansamblu a sportivului.

Pe de altă parte, pentru fotbaliști în sine acest tip de aplicație reprezintă o oportunitate pentru ei de a se face remarcați în această lume competitivă, în care informațiile circulă rapid și deciziile sunt greu de prevăzut. Având în vedere complexitatea acestui proces de recrutare, este imposibil ca de-a lungul timpului anumite talente foarte promițătoare să nu fii trecut sub radarul scouterilor. Mai mulți sportivi de perspectivă înseamnă, până la urmă, un câștig pentru toată lumea, de la cluburi, la spectatori și, bineînțeles, pentru ei înșiși.

În al doilea rând, această aplicație ar putea fi folosită și de alte categorii de utilizatori care ar avea nevoie să observe jucători din alte echipe, dacă există o dorință de a aduce noi jucători sau, de ce nu, de a-i analiza pe cei din echipele lor. Exemple de categorii de beneficiari ar putea fi antrenorii (principali și secunzi), analiștii tehnici, arbitrii, jurnaliștii și, de ce nu, chiar jucătorii în cauză. În plus, chiar și pasionații de fotbal, ce nu sunt angrenați profesional în acest fenomen, dar vor să aibă o privire mai analitică asupra fotbalului, pot alege să exploreze această perspectivă.

Următoarele capitole parcurg în mod progresiv etapele necesare pentru proiectarea, implementarea și documentarea acestei aplicații. Scopul este de a facilita înțelegerea necesității acesteia, modul în care acesta a fost conceput, detalii despre implementarea modulelor sale și funcționalitățile oferite utilizatorilor finali.

Primul capitol se concentrează pe evaluarea cunoștințelor existente în domeniul studiului, oferind o sinteză a literaturii de specialitate relevante pentru subiect. Prin analizarea câtorva studii semnificative, se pun în evidență contribuțiile cheie și perspectivele relevante ale altor autori în domeniul de interes. Scopul este de a demonstra o înțelegere solidă a contextului în care produsul software ar putea apărea, furnizând astfel un suport teoretic robust pentru analiza și interpretarea propriilor rezultate.

Al doilea capitol cuprinde prezentarea metodologiei cercetării, adică modul în care voi concepe și proiecta aplicația și care va fi împărțită în acest fel:

Prima parte oferă o descriere a funcționalităților de bază, prin specificarea cerințelor, cu ajutorul diagramei cazurilor de utilizare, ce a evidențiat obiectivele principale și a conturat o viziune abstractă a întregului sistem.

A doua parte tratează analiza sistemului informatic prin utilizarea unor instrumente specifice, ce au ca scop evidențierea structurii și a felului în care se comportă aplicația, în diferite perspective și scenarii.

Ultima parte a acestui capitol tratează proiectarea sistemului și servește drept legătură între modelele generale și echivalentele lor tehnice, cu accent pe validarea informațiilor obținute în faza de analiză și observarea oricăror inconsecvențe în timpul traducerii acestora în termeni tehnologici.

Lucrarea se încheie cu formularea unor concluzii care sintetizează rezultatele și apreciază calitatea și utilitatea sistemului, precum și cu identificarea potențialelor direcții de dezvoltare viitoare a sistemului.

## **Capitolul 1 - Stadiul cunoașterii în domeniu**

Așa cum a fost menționat și în introducere, acest capitol se concentrează pe explorarea, sintetizarea și analiza critică a diferitelor articole din zonele care cuprind tematica acestei lucrări, pentru a ajuta la formarea unei opinii cât se poate de bine documentate și a unei viziuni de ansamblu asupra contextului actual al problematicii puse în prim-plan.

Astfel, am decis să selectez în cadrul acestui demers cercetări din mai mult perspective ale fenomenului de căutare și analizare al jucătorilor de fotbal, cum ar fi: prezentarea procesului de scouting, impactul tehnologiei în lumea sportului și respectiv modalități de implementare a unor soluții din domeniul tehnologiei informației în procesul de identificare și evaluare al jucătorilor de fotbal.

### **Premise**

Un context foarte bun în această direcție îl oferă articolul *Data Driven football scouting assistance with simulated player performance extrapolation* (Shantanu Ghar, Sayali Patil, Venkhatesh Arunachalam, 2021). În prezent, dezvoltarea tehnologiei și infrastructura internetului tot mai cuprinzătoare a creat posibilitatea ca fotbalul să fie văzut de milioane de oameni la un click distanță. Această explozie a numărului de spectatori a rezultat în oportunități de business mai mari și, bineînțeles, în sume de bani exorbitante care au fost investite în această direcție. Spre exemplu,

Premier League, cel mai puternic campionat englez, este cea mai bogată ligă din lume și faptul că sumele de bani folosite pentru achiziționarea jucătorilor sunt foarte mari este de la sine înțeles. Toate acestea fac ca găsirea jucătorului ideal pentru o poziție anume să fie critică, întrucât neadaptarea acestuia la echipă va duce la pierderi materiale semnificative. Prin urmare, importanța scouting-ului nu poate fi ignorată, iar aceasta necesită mult research și analiză în detaliu a sportivilor.

Un scouter trebuie să analizeze un fotbalist prin urmărirea acțiunilor sale din joc, calitățile fizice și să facă o presupunere legată de cum se va descurca în echipă. Fiecare club are o formație, un stil de joc și un profil specific de jucător care este dorit în funcție de factorii menționați anterior. Totuși, persoanele delegate cu astfel de sarcini nu au decât posibilitatea urmăririi în câteva meciuri în persoană a unui jucător înainte să realizeze un raport referitor la performanța acestuia.

Fotbalul este un sport complex, performanța unui jucător depinde de capacitățile fizice, tehnice, tactice, dar și de componenta psihologică și mentală. Evaluarea unui jucător bazată pe impresii subiective sau diferite statistici pare aproape imposibil de realizat. Obiectivul echipelor de fotbal și al jucătorilor este, totuși, unul cât se poate de simplu, acela de a câștiga (*A football player rating system*, Stephan Wolf, Maximilian Schmitta and Bjorn Schuller, 2020).

### **Cum găsesc scouterii viitoarele talente**

Scouterii cluburilor de fotbal sunt de obicei primii care identifică jucătorii talentați. Totuși, există o lipsă de research în ceea ce privește modul în care aceștia apreciază și prezic performanța fotbalistică per total. De aceea, în *How soccer scouts identify talented players* (Tom L. G. Bergkamp, Wouter G. P. Frencken, A. Susan M. Niessen, Rob R. Meijer & Ruud. J. R. den Hartigh, 2022) s-a încercat realizarea unui studiu care să examineze acest proces. Pentru a realiza prognoze relevante, scouterii trebuie să pună în balanță diferite aspecte, cum ar fi: vârsta la care jucătorilor li se poate aprecia o viitoare carieră, ce calități sunt relevante pentru a realiza o astfel de prognoză și să facă o aproximare cât mai corectă bazată pe predictorii selecțiați.

Astfel, în urma acestui studiu, următoarele concluzii au fost stabilite: prima este că scouterii sunt conștienți de faptul că indicatorii timpurii ai unei performanțe viitoare pot lipsi, însă în ciuda acestui fapt aceștia recomandă selecția posibilelor talente cât mai devreme. De asemenea, în procesul de identificare a potențialilor jucători, scouterii consideră mai ușor de observat o anumită categorie de calități, cum ar fi cele tehnice. Totuși, aceștia descriu că își formează mai degrabă o privire de ansamblu când vine vorba de un sportiv decât să își formeze o opinie bazată pe anumiți predictorii. Nu în ultimul rând, scouterii au susținut că, deși evaluează jucătorii într-o manieră structurată, predicția finală este o integrare intuitivă a mai multor calități, ceea ce e considerat suboptim în literatura de specialitate. Așadar, scouterii sunt încurajați să se bazeze mai mult pe criterii predefinite pentru a oferi o predicție cât mai exactă.

### **O nouă cale de a identifica jucătorii**

O dată cu publicarea studiului *War for Talent* (Elizabeth G. Chambers, Mark Foulon, Helen Handfield-Jones, Steven M. Hankin, Edward G. Michaels III, 1998), subiectul managementului

talentor a primit o atenție deosebită din partea academicienilor și a celor din domeniul recrutării. Pentru toate organizațiile, punerea în valoare a anumitor angajați este un element esențial în atingerea obiectivelor propuse. Astfel, în teorie, toate rolurile ar trebui să fie ocupate de către cei mai buni din domeniu. Cu atât mai mult în sport, identificarea jucătorilor cu potențial a stat la baza cluburilor și federațiilor de fotbal, pentru că aceștia sunt poate cele mai de valoare “active” ale organizațiilor. Prin prisma acestui fapt, procesul de scouting a inspirat și procesele de recrutare clasice, pentru că cele mai de succes echipe sunt cele care identifică mai rapid sportivii mai buni decât competiția.

*Social Talent Scouting: A New Opportunity for the Identification of Football Players?* (Elena Radicchi, Michele Mozzachiodi, 2016) oferă o perspectivă mai detaliată asupra recrutării în zona sportivă. Mai multe organizații sportive se bazează aproape exclusiv pe expertiza umană pentru procesul de scouting. În activități precum fotbalul, unele țări, cum ar fi Italia, se crede în continuare acel domeniu expertii (antrenori, managerii, analiștii) pot converti în mod eficient datele colectate în cunoștințe utilizabile.

În schimb în alte, contexte, ca cel european sau mai degrabă cel american, industria sportului are o tradiție mai lungă de a colecta date statistice, în special în baseball și baschet. În ultimii ani, noile tehnologii (software pentru analiza jucătorilor, bazele de date multimedia online etc.) au oferit avantaje suplimentare, cum ar fi prezicerea unor anumite rezultate ale meciurilor și/sau prognozarea modului în care un atlet ar putea performa în anumite condiții. Explorarea website-urilor și a platformelor online pentru scoutingul talentelor fotbalistice și interviurilor cu profesioniștii din echipele și federațiile italiene indică faptul că în general există două direcții în intersecția online-ului cu acest sport: suport pentru activitatea “pe teren” și descoperirea de noi talente.

Scouterii joacă un rol intermediar între fotbaliști și cluburi și beneficiază de diferite surse de informare (agenți FIFA, agenți ai jucătorilor etc.). Internetul poate fi un instrument valoros, pentru scouteri pentru selectarea de informații despre jucători, observarea și găsirea unor potențiali candidați. De asemenea, ar putea reduce costurile de transport ale scouterilor și să ajute la creșterea capacității cantitative și geografice de a putea identifica jucători.

Așadar, implementarea unor soluții IT în domeniul sportului pot să aducă un plus valoare acestuia. Totuși, cel mai probabil, scouting-ul “pe teren” nu va dispărea complet, ci va fi complementar cu această nouă modalitate de a recruta talente. În plus, anumite cluburi sau culturi fotbalistice sunt mai reticente la schimbările de acest gen, ceea ce duce la o îmbunătățire mai lentă în această direcție.

### **Impactul Tehnologiei Informației în procesul de scouting din sport**

Participarea în sport este de obicei asociată cu intenția de a îmbunătățire. Una din cele mai importante variabile care influențează învățarea și succesul este feedback-ul. Dar, pentru a avea un feedback de calitate, antrenorii au nevoie să observe și să evalueze performanța. Totuși, mai multe studii au arătat că acest gen de observații pot fi eronate, din cauza capacităților limitate ale oamenilor de a memora toate detaliile unui meci. Mai mult decât atât, nu există niște criterii clare pentru evaluarea și analiza jucătorilor. Dezvoltarea tehnologică a îmbunătățit sistemul de colectare a datelor pe parcursul competițiilor și le-a făcut mai ușor de analizat. Emergența elementului video, precum și

a tehnologiei digitale a reliefat 2 efecte: mai multe acțiuni pe care datele pot fi culese, precum și aspecte mai specifice ce pot fi analizate.

Apariția digitalizării și a computerelor de buget, dublată de evoluția stocării și capacității de editare a datelor video, au mărit viteza cu care datele legate de sporturi sunt colectate. Aceste sisteme permit un anumit grad de interactivitate astfel încât analistul să poată controla unii parametri înainte de a realiza un raport.

Așadar, impactul tehnologiei în această arie a influențat considerabil sportul, în special pe cele în care se încearcă dominarea adversarului prin câștigarea spațiilor în teren (*The Impact of Information Technologies on the Scouting*, Marković, Srđan, Ivan Ćuk, and Aleksandar Živković, 2020).

### **Aplicații dezvoltate în această direcție**

În procesul meu documentare și cercetare a domeniului, am identificat câteva aplicații relevante în zona de scouting a jucătorilor de fotbal, ce au diferite funcționalități și au la bază tehnologii diferite.

Probabil cea mai importantă aplicație în contextul lucrării de față este **Wyscout**. Această aplicație folosește modalități de căutare bazate pe diferite criterii, oferirea de diferite date și videoclipuri despre jucători, generarea de statistici și rapoarte despre aceștia și modalități de analiză video.

**Hudl** este o aplicație care se concentrează pe analiza video a meciurilor sportive. Aceasta oferă analiștilor posibilitatea de a eticheta sau marca jucătorii, de a desena pe video și de a reda cu încetinitor momentele din timpul meciului.

O altă aplicație din această zonă este **InStat Scout**. Este o platformă web pentru a analiza echipe, jucători, arbitri din sporturi ca fotbalul, tenisul sau baschetul. Aceasta oferă acces la o gamă largă de statistici. Partea unică a acestei aplicații este că toate rapoartele sunt legate de clipuri video care reflectă caracteristicile generate despre jucător.

O altă unealtă disponibilă pentru a putea fi folosită este **The Scouting App**. Aceasta permite utilizatorilor să introducă date despre jucătorii pe care doresc să îi observe, să facă analize comparative între ei, să aibă acces la un dashboard cu diferite statistici bazate pe datele de intrare.

**Opta**, pe de altă parte, este cunoscută în industrie ca fiind una dintre cele mai dezvoltate baze de date despre jucători din lume. Aceasta simplifică accesul la analize statistice și unelte de analiză ce acoperă totul de la poziționare la performanțe individuale din timpul meciului.

Astfel, am putut remarca faptul că, în prezent, există mai multe aplicații care pot fi folosite pentru a putea ajuta echipele de fotbal în a recruta diferite talente. Principalele funcționalități pe care acestea le au se bazează pe posibilitatea de a căuta diferiți jucători, de a analiza videoclipuri video și de a genera diferite statistici și rapoarte ce se bazează pe datele generate despre sportivii analizați.

Aplicația pe care doresc să o proiectez și să o implementez va fi una web și va folosi **React** pe partea de front end și **Node.JS** pe partea de back-end cu o bază de date relațională. Scopul acesteia va fi de a ajuta scouterii să caute diferiți jucători și de a reuși să aibă o privire de ansamblu asupra acestora, în vederea găsirii celor ce pot arăta un potențial potrivit cu așteptările acestora.



De aceea, funcționalitățile pe care doresc ca aplicația mea să le aibe sunt: în primul rând, autentificarea și înregistrarea pe platformă, posibilitatea de a căuta jucători după diferite criterii (nume, echipă, etc.), posibilitatea de a vedea diferite date și statistici relevante despre acesta. Dacă un scout găsește un jucător potrivit, îl poate adăuga într-o listă de jucători preferați. De asemenea, voi încerca să implementez și funcționalitatea de a vedea clipuri video cu jucători sau echipele din care aceștia fac parte sau, dacă nu pot găsi un API care să faciliteze aceasta posibilitate, să încerc să le permit utilizatorilor să poată încărca un videoclip de pe dispozitivul propriu și să poată folosi unelte de desenare pe acesta.

## Capitolul 2 - Etapa de analiză

Produsul software ce urmează să fie implementat are ca scop oferirea unei modalități de a realiza identificarea, observarea și evaluarea jucătorilor de fotbal într-un mod mai rapid, mai ușor, de la distanță, acest lucru fiind făcut într-un mod centralizat. Funcționalități ale altor aplicații care au diferite scopuri vor fi aduse împreună într-o nouă modalitate, mai prietenoasă și mai intuitivă.

În urma etapei de cunoaștere în domeniu, direcția pe care îmi doresc a o avea aplicația mea este de a asigura o experiență fluentă unui scout ce dorește să identifice noi talente fotbalistice, sau a altor persoane care doresc să aibe o experiență mai analitică a sportului, de la tacticieni, la arbitri sau chiar la sportivi în sine.

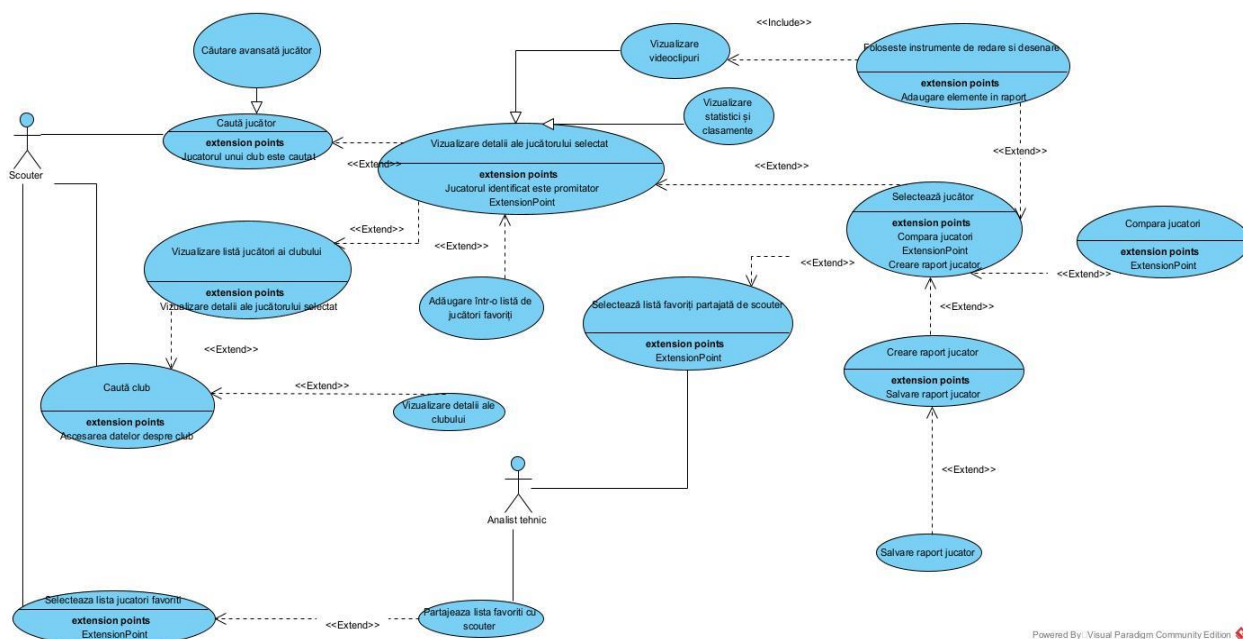
Acest capitol are în prim-plan cerințele funcționale pe care sistemul informatic trebuie să le atingă. Diagramele care vor fi folosite în lucrarea de față respectă regulile și convențiile limbajului de modelare universal (UML) și sunt create cu ajutorul programului *Visual Paradigm Community Edition 17.1*. Tipurile de diagrame prezentate în această parte a lucrării sunt:

- Diagrama cazurilor de utilizare;
- Diagrama de activitate;
- Diagrama de stare;
- Diagrama de interacțiune.

### 2.1. Diagramele cazurilor de utilizare

Această diagramă oferă o modalitate de a reliefa mult mai ușor și eficient funcționalitățile pe care le dorim ca aplicația să le poată oferi într-un stadiu final. În vederea realizării acestei modelări au fost folosite două categorii de diagrame: cea generală a cazurilor de utilizare și descrierea particulară a 2 cazuri de utilizare. (Figura 2.1)

### 2.1.1 Diagrama generală a cazurilor de utilizare



*Figura 2.1 Diagrama generală a cazurilor de activitate*

Această diagramă pune în prim-plan atât actorii care vor putea folosi sistemul informatic, și anume *Scouterul* și *Analistul tehnic*, cât și funcționalitățile de bază pe care produsul le are.

Utilizatorii de tip *Scouter* pot căuta jucători pe care vor să îi analizeze, să vizualizeze diferite date generale despre aceștia, dar și să îi poată adăuga într-o listă favoriți, dacă aceștia prezintă interes.

Pe de altă parte, utilizatorii de tip *Analist tehnic* au posibilitatea de evalua sportivii dintr-o perspectivă mai tactică, prin analiza grafică a unor clipuri video și compararea a 2 jucători în paralel.

### 2.1.2 Caz de utilizare - *Caută jucător*

Element	Descriere element
Cod	Search-Player
Stare	Schiță
Scop	Identificarea jucătorilor de interes din aplicație și în funcție de anumite criterii
Nume	Căutare jucător
Actor principal	Scouter
Descriere	Scouterul caută potențiali jucători în funcție de nume, competițiile în care evoluează sau alte metode de căutare

	avansată
Precondiții	Scouterul este înregistrat și autentificat pe platformă
Postcondiții	Jucătorii găsiți pot fi selectați pentru o eventuală observare
Declanșator	Scouterul alege opțiune de identificare și adăugare jucători
Posibile erori	Jucătorul căutat în funcție de criteriile cerute nu este găsit
Starea sistemului în caz de eroare	Se va afișa un mesaj de eroare “Niciun jucător care să îndeplinească aceste criterii nu a fost găsit”
Flux de bază	<ol style="list-style-type: none"> <li>1. Scouterul accesează platforma și navighează la secțiunea "Caută Jucători".</li> <li>2. Utilizând funcționalitățile de căutare, scouterul filtrează jucătorii în funcție de criterii precum nume, poziția, performanțele, club și altele.</li> <li>3. Scouterul decide ce profil de jucător dorește să examineze.</li> </ol>
Fluxuri alternative	-
Relații	Vizualizarea detaliilor despre jucător

### 2.1.3 Caz de utilizare - Crearea raportului

Element	Descriere element
Cod	Generate-report
Stare	Schiță
Scop	Realizarea unui raport tehnic detaliat în ceea ce ține de performanțele jucătorului selectat
Nume	Creare raport jucător
Actor principal	Analist tehnic
Descriere	Analistul va scrie detaliat observațiile legate de jucătorul de

	fotbal evaluat
Precondiții	Analistul de performanță este autentificat în platforma de scouting și are acces la datele jucătorilor. Analistul a analizat detaliat un jucător folosind instrumentele disponibile pe platformă.
Postcondiții	Clubul de fotbal dispune de informații detaliate și analize comparative care îi permit să ia decizii informate în ceea ce privește recrutarea și achiziționarea jucătorilor sau îmbunătățirea performanței actuale a echipei.
Declanșator	Alegerea opțiunii de evaluare a jucătorilor
Posibile erori	Raportul ce trebuie generat este gol
Starea sistemului în caz de eroare	Un raport care este gol nu poate fi generat
Flux de bază	<ol style="list-style-type: none"> <li>1. Analistul de performanță accesează platforma și selectează secțiunea "Evaluare și Comparare Jucători".</li> <li>2. Utilizând instrumentele de analiză disponibile, analistul selectează și evaluează jucătorii pe care aceasta îi dorește.</li> <li>3. Analistul redactează raportul, putând să adauge secvențe de videoclipuri și grafice sau statistici realizare cu instrumentele disponibile.</li> <li>4. Analistul generează rapoarte și recomandări pe baza evaluărilor sale, asistând astfel clubul în procesul de luare a deciziilor privind transferurile și achizițiile de jucători sau de îmbunătățire a performanței echipei.</li> </ol>
Fluxuri alternative	-
Relații	-

## 2.2 Diagramele de activitate

Diagramele de activitate sunt folosite pentru a modela procesele procedurale ale unui sistem, oferind o reprezentare vizuală a secvențelor de acțiuni necesare pentru a atinge un rezultat specific.

Ele ilustrează fluxul de lucru de la început până la sfârșit, incluzând căile de decizie care pot apărea pe parcursul unei activități. Mai jos am atașat trei diagrame de activitate.

### 2.2.1 Diagrama de activitate - *Caută jucător*

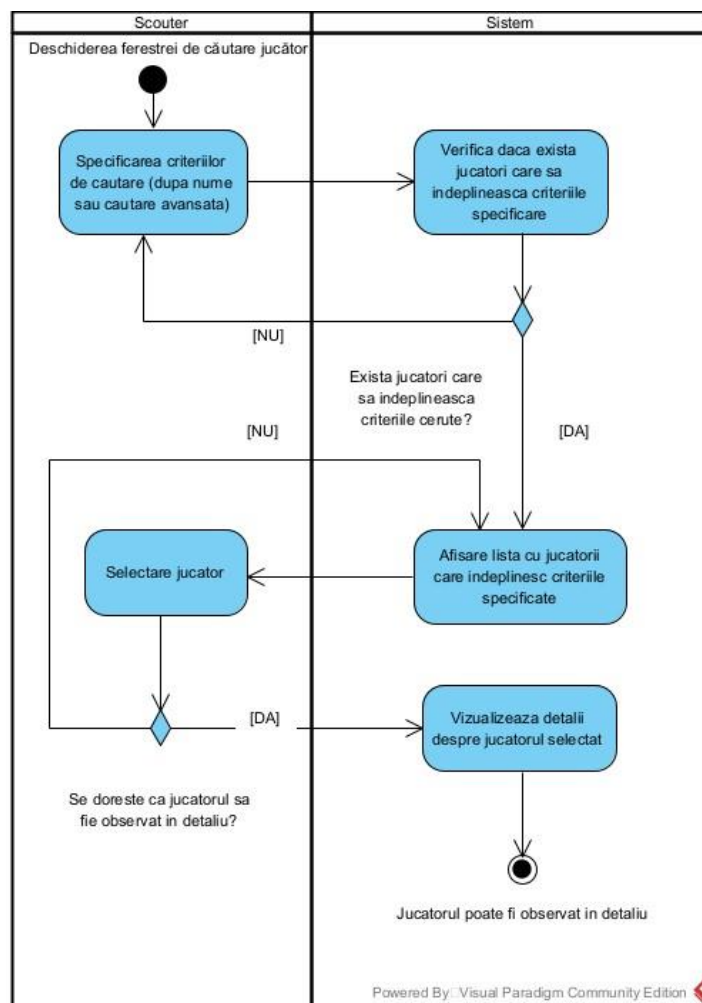


Figura 2.2 Diagrama de activitate “Caută jucător”

Prima diagramă surprinde una dintre funcționalitățile de bază pe care aplicația mea le are, și anume posibilitatea ca un utilizator să caute un jucător. Primul pas este acesta să selecteze opțiunea de a căuta jucători, urmând ca acesta să introducă numele, sau alt criteriu de căutare pe care acesta îl dorește. Dacă există sportivi care să corespundă criteriilor selectate, acesta poate să aleagă să îi vizualizeze profilul. În schimb, dacă niciun jucător nu este găsit conform căutării sau dacă cei găsiți nu sunt considerați potriviți pentru a fi observați mai detaliat, căutarea poate fi reluată.

## 2.2.2 Diagrama de activitate - *Vizualizare detalii jucător selectat*

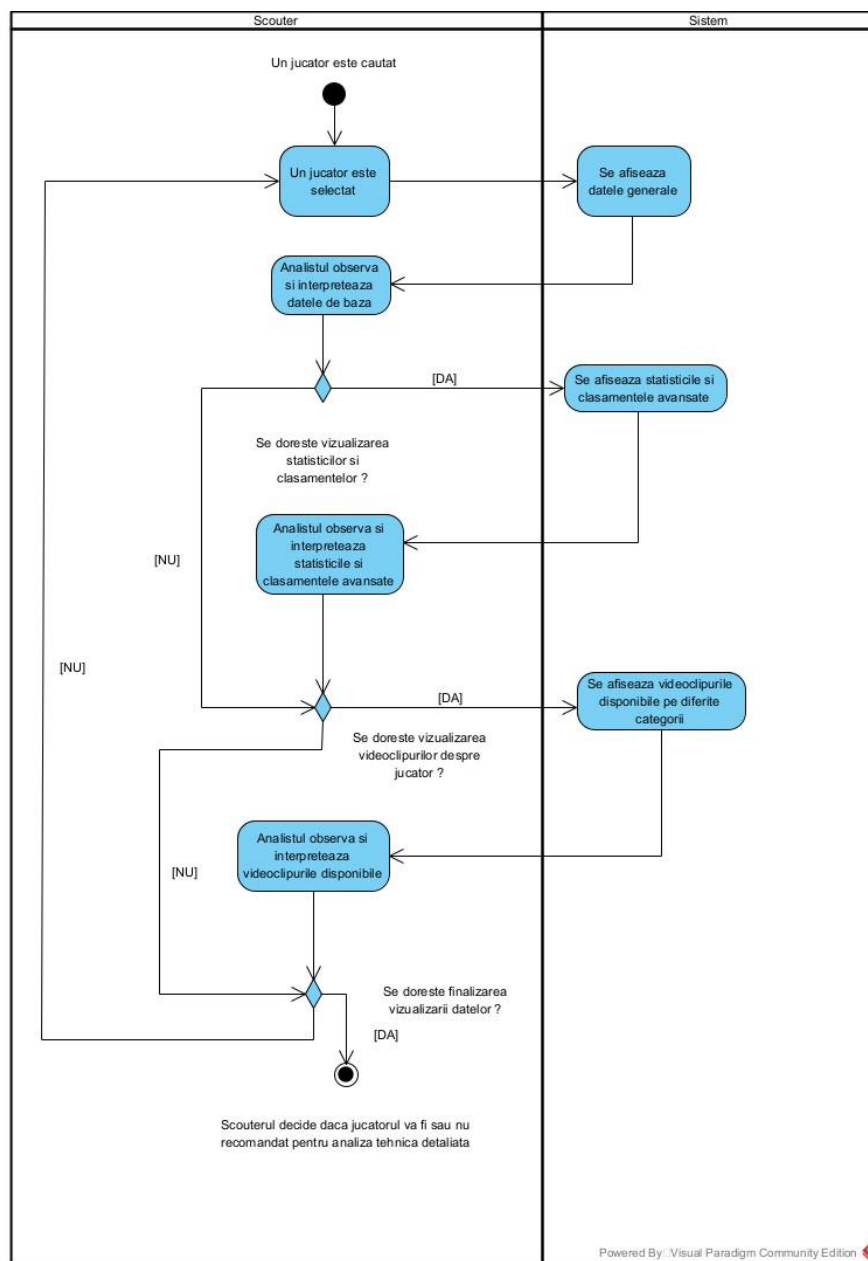


Figura. 2.3 Diagrama de activitate “Vizualizare detalii jucător selectat”

Această diagramă pune în evidență funcționalitatea unui jucător de a fi selectat, de a se putea vizualiza câteva date de bază despre acesta, iar în urma selectării opțiunii aferente, se va afișa o prezentare mai detaliată a acestuia cu câteva statistici mai specifice și posibilitatea vizionării de clipuri video în care sportivul să apară. Pe tot parcursul procesului, scouterul se poate întoarce la pașii anteriori de observare sau poate să finalizeze procesul.

### 2.2.3 Diagrama de activitate - *Folosește instrumente de redare și desenare*

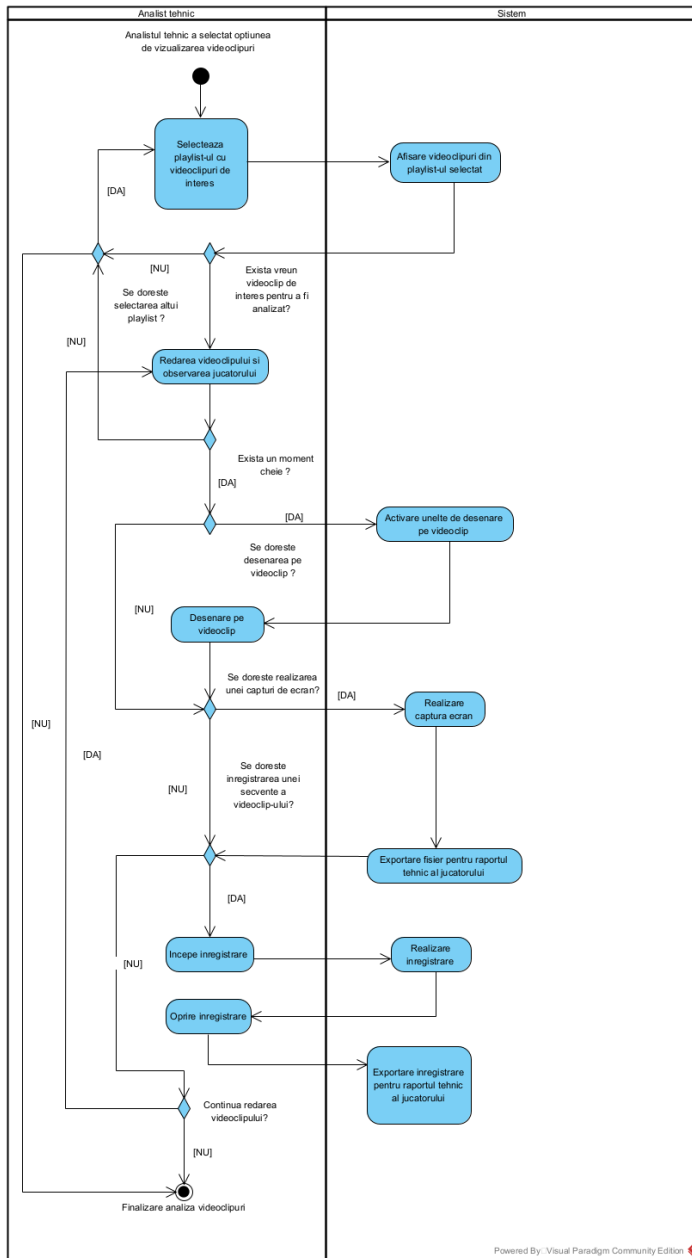


Figura. 2.4 Diagrama de activitate “Folosește instrumente de redare și desenare”

Nu în ultimul rând, o altă funcție a acestui produs software ce poate fi exemplificată în acest tip de diagramă va fi opțiunea de a putea desena și reda clipuri video cu un anumit jucător. Analistul va putea selecta dintr-un playlist de videoclipuri pe cel pe care dorește să îl vizualizeze, iar dacă acesta consideră, la un moment dat, că există un “moment cheie”, acesta poate să fie înregistrat sau se pot folosi instrumente de desenare sau realizarea de capturi de ecran pe anumite cadre de pe parcursul lui.

## 2.3 Diagramele de stare

Diagramele de stare sunt instrumente vizuale utilizate pentru a descrie comportamentul unei entități într-un anumit moment de timp. Ele ilustrează modul în care un sistem se schimbă și reacționează la diferite evenimente externe sau interne.

### 2.3.1 Diagrama de stare - *raport tehnic*

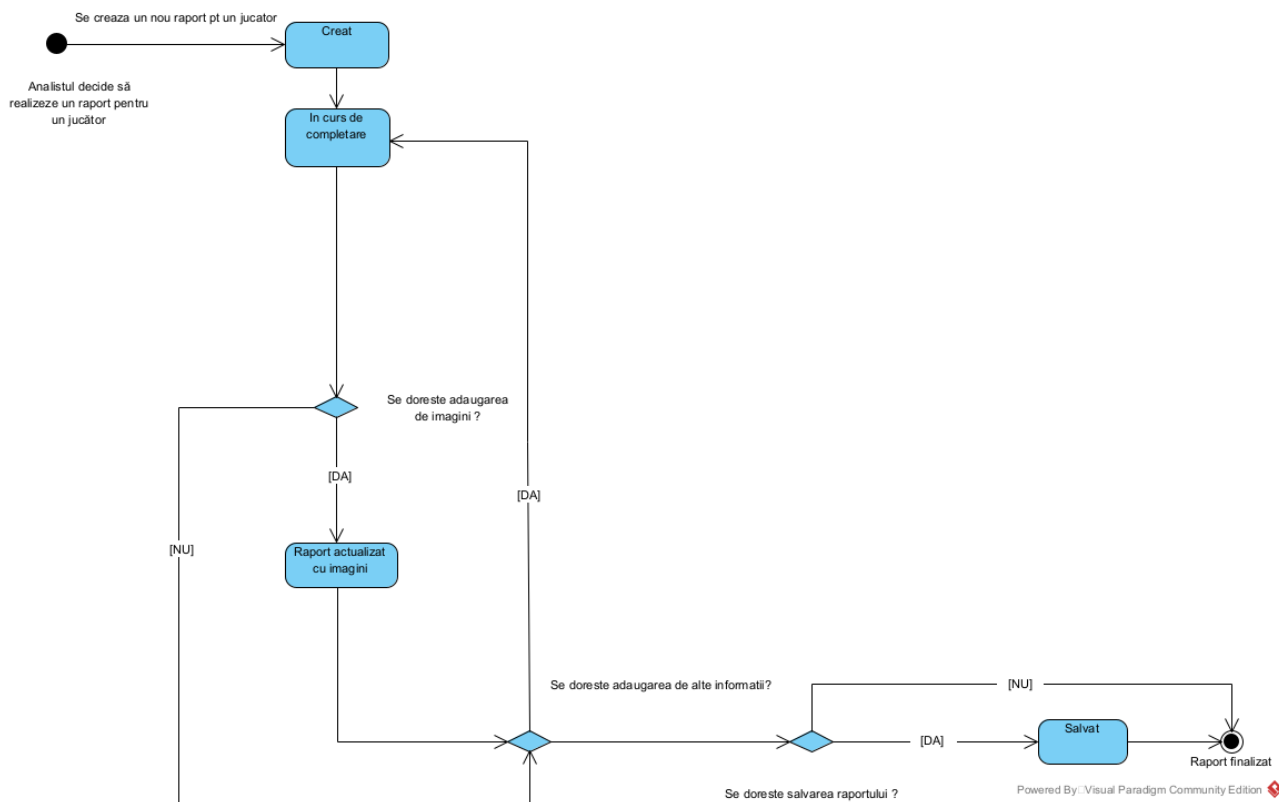


Figura. 2.5 Diagrama de stare - “Raport tehnic”

Obiectul pe care am ales să îl descriu în această diagramă de stare este raportul tehnic și are 4 stări. Acesta trece din starea de *creat*, atunci când utilizatorul decide să evalueze un jucător, la starea de *în curs de completare*, atunci când utilizatorul poate să adauge informații text în care descrie abilitățile și modul de joc al unui jucător. Raportul poate fi *actualizat* cu diferite *imagini* din videoclipuri cu acesta și, opțional, poate fi *salvat*. Ulterior, documentul ajunge în starea de *finalizat*.



### 2.3.1 Diagrama de stare - *jucător de fotbal*

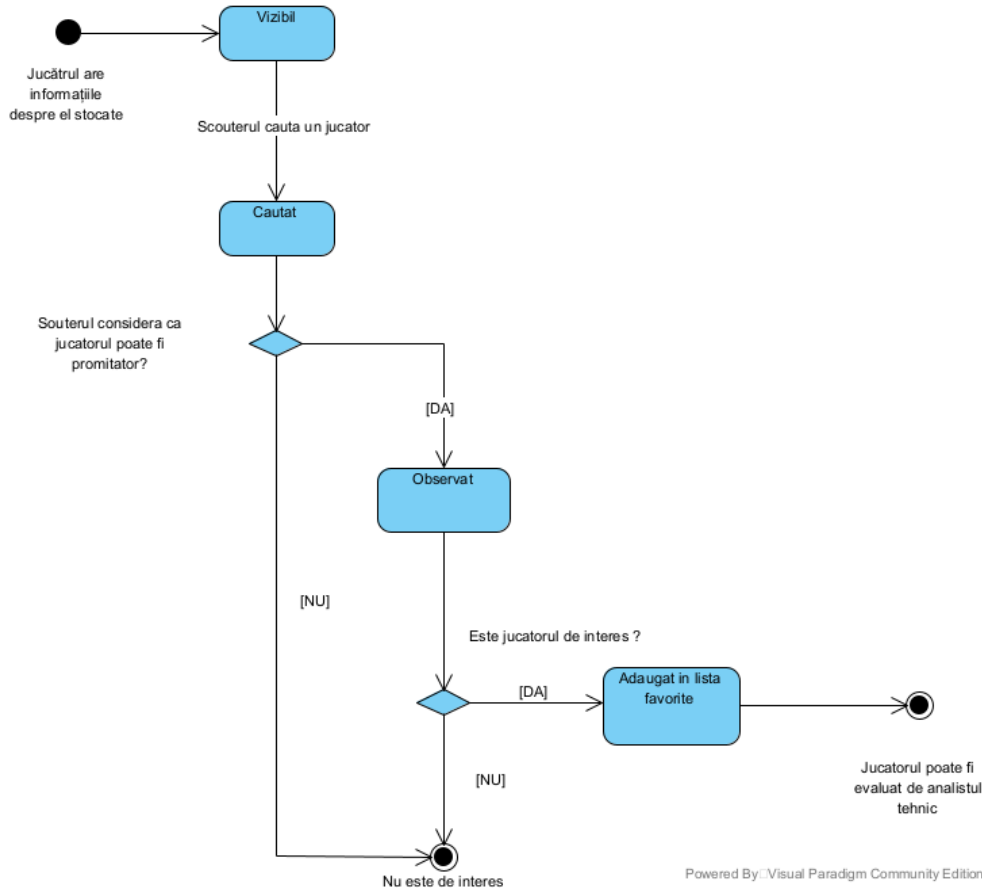


Figura. 2.6 Diagrama de stare “Jucător de fotbal”

Cealaltă diagramă de stare surprinde entitatea *jucător de fotbal*, care trece prin următoarele stări: inițial, acesta este *vizibil* pentru a fi căutat prin intermediul platformei de scouting, urmând ca apoi să treacă în stadiul de *observat*, dacă un utilizator din poziția de scouter consideră că acesta prezintă interes. În final, dacă fotbalistul este promițător, va fi *adăugat* într-o listă de favorite care poate fi partajată cu un evaluator tehnic.

## 2.4 Diagramele de interacțiune

Diagramele de interacțiune sunt utilizate pentru a descrie modul în care obiectele și componentele unui sistem colaborează pentru a realiza funcționalități specifice. Aceste diagrame evidențiază schimbul de mesaje între obiecte într-un anumit context temporal și spațial, facilitând înțelegerea detaliată a comportamentului dinamic al sistemului. Prin reprezentarea vizuală a secvențelor de interacțiuni și a mesajelor transmise, diagramele de interacțiune ajută la clarificarea responsabilităților fiecărui obiect și la identificarea punctelor critice de comunicare și sincronizare în cadrul sistemului.

### 2.4.1 Diagrama de interacțiune - *Identificare jucător de fotbal*

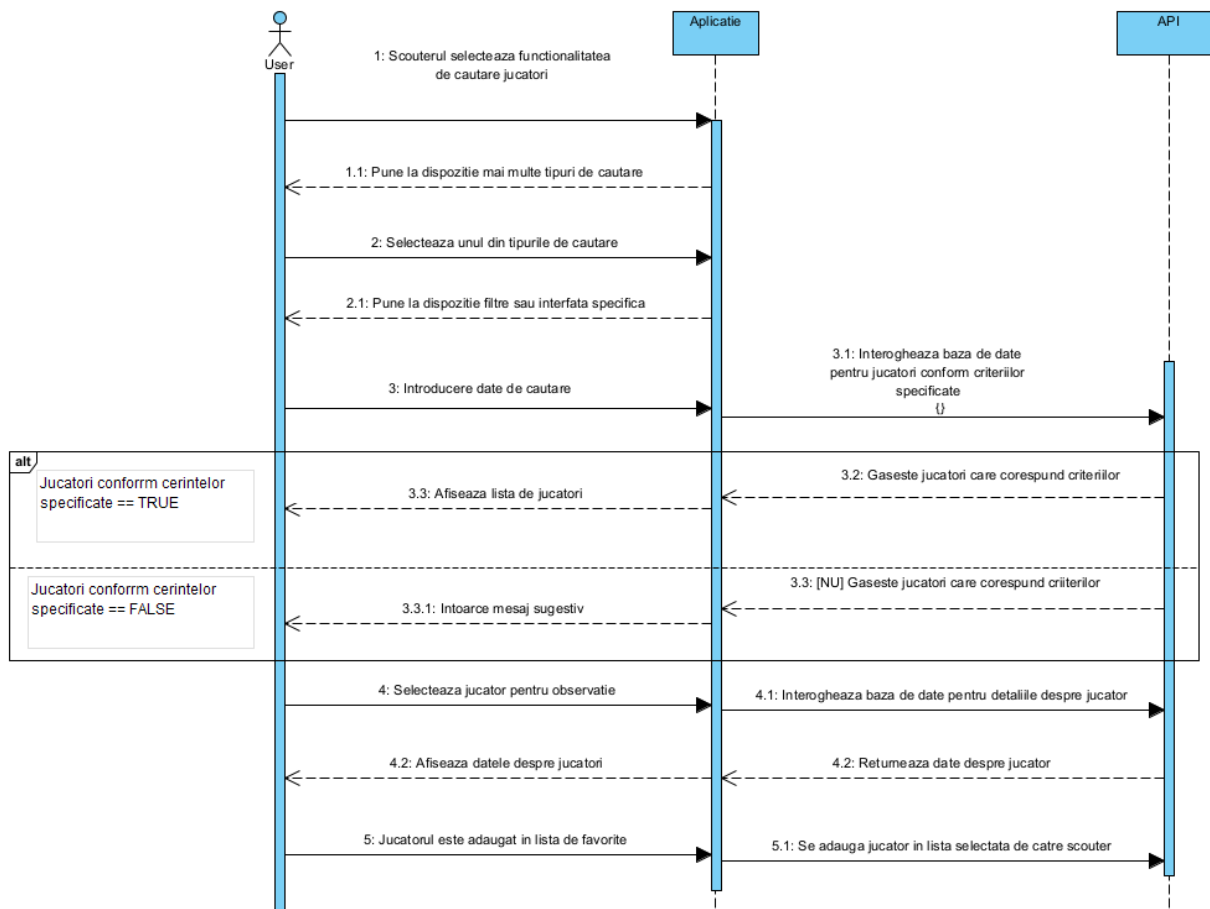


Figura. 2.7 Diagrama de interacțiune “Identificare jucător de fotbal”

Diagrama din Figura 2.7 prezintă modalitatea prin care un user poate identifica un jucător care este relevant criteriilor selectate. Mai întâi, se va selecta opțiunea de *Căutare jucători*, iar sistemul va permite căutarea de jucători după anumite preferințe. După alegerea tipului de search dorit și introducerea datelor de intrare, vor exista două variante alternative: prima, în care o listă de unul sau mai mulți jucători va fi afișată conform regulilor de căutare și a doua, în care niciun jucător nu este găsit, iar un mesaj aferent unei căutări fără rezultat va fi afișat. În continuarea primului scenariu, un fotbalist poate fi selectat pentru a fi observat și, dacă este o potențială adiție la echipă, va fi adăugat într-o listă de preferințe.

## 2.4.2 Diagrama de interacțiune - *Accesare informații despre un club de fotbal*



Figura. 2.8 Diagrama de interacțiune “Identificare jucător de fotbal”

Un scenariu mai simplu decât cel anterior este pus în evidență în această diagramă de interacțiune. Un utilizator poate să aleagă dintre competițiile puse la dispoziție de către sistem. Acesta va returna toate campionatele disponibile în momentul de timp selectat. În urma accesării unei competiții, vor fi prezentate toate cluburile ce participă în acea ediție. Nu în ultimul rând, prin selectarea unui anume echipe, se vor afișa informațiile aferente acesteia (valoare, nr, jucători etc.).

## 2.4.3 Diagrama de interacțiune - *Evaluarea unui jucător de fotbal*

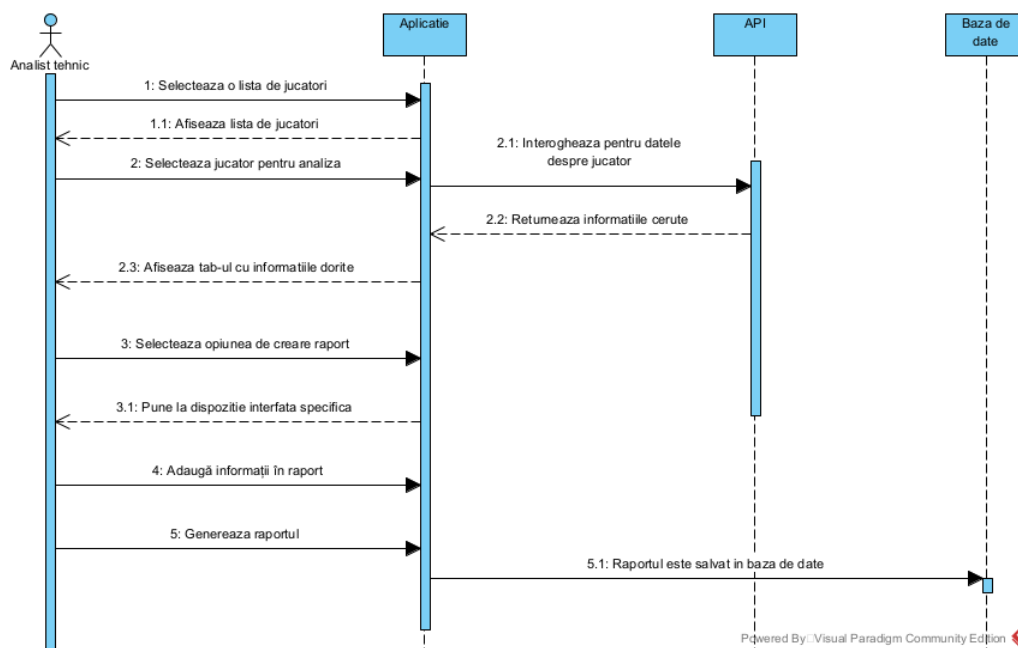


Figura. 2.9 Diagrama de interacțiune “Evaluarea unui jucător de fotbal”

Ultima diagramă din acest subcapitol, adică *Figura. 2.9*, are în prim-plan interacțiunea utilizatorului cu diferite părți ale sistemului cu scopul de a realiza o observație despre un anumit jucător. Analistul tehnic poate selecta un jucător adăugat într-una din listele de favorite la care are acces, sistemul îi va returna informațiile relevante despre fotbalistul respectiv, iar “munca de cercetare” poate începe. Când acesta dorește, poate să realizeze un raport de observare, în care să adauge evaluarea jucătorului din diferite perspective și, în cele din urmă, poate salva documentul în baza de date.

## **Capitolul 3 - Etapa de proiectare**

Proiectarea este o etapă importantă în cadrul dezvoltării unui sistem informatic, definind arhitectura, componentele și interacțiunile dintre acestea pentru a asigura îndeplinirea cerințelor funcționale și nefuncționale specificate în etapa de analiză, vor fi detaliate aspectele esențiale care conduc la realizarea unui sistem robust și eficient.

### **3.1 Diagrama de clase detaliată**

Diagrama de clase detaliată pentru acest produs software reflectă structura și relațiile dintre principalele entități implicate în procesul de identificare și evaluare a jucătorilor de fotbal. Astfel, se evidențiază relațiile de asociere, agregare și compoziție între aceste clase, precum și metodele esențiale necesare pentru funcționalitatea aplicației. De exemplu, un Scouter poate vizualiza și gestiona liste de jucători (Listă Jucători), iar Analistul Tehnic poate crea rapoarte tehnice (Raport Tehnic) pentru Jucători. Fiecare Jucător are asocieri cu Cluburi și statistici proprii (Statistică Jucător). Aceste relații și interacțiuni sunt esențiale pentru procesul de evaluare detaliată și precisă a jucătorilor, asigurând astfel o gestionare eficientă a datelor și o evaluare corectă în cadrul aplicației.

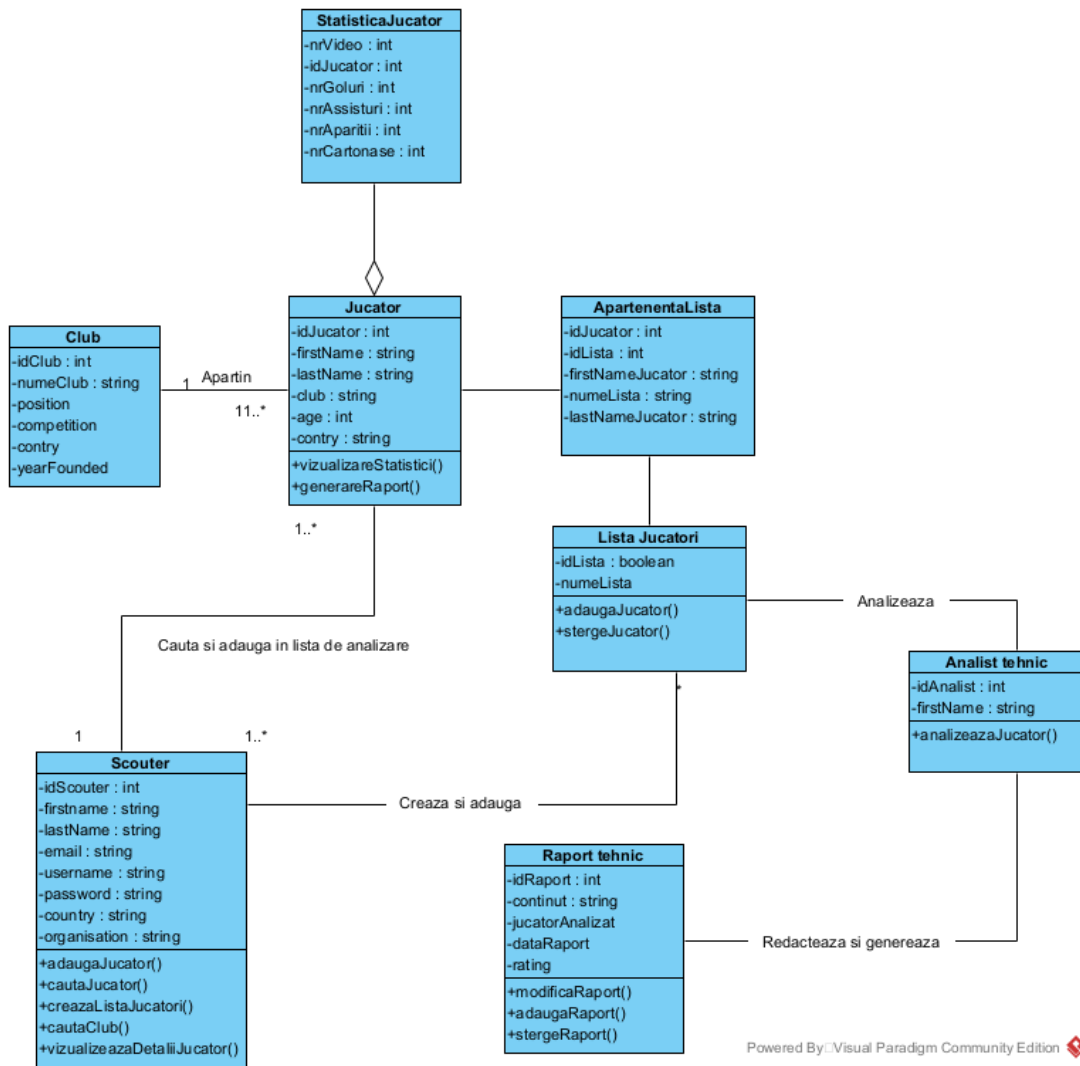
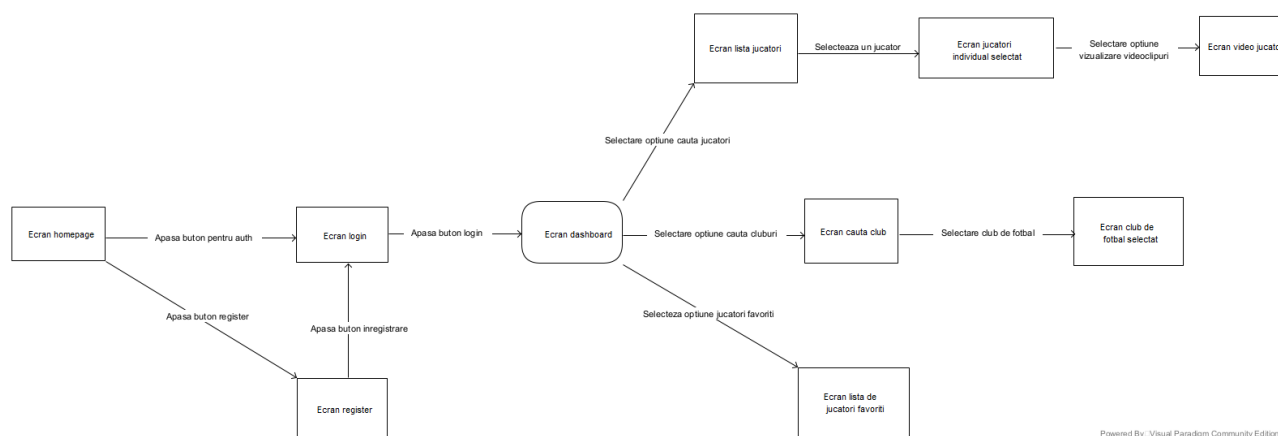


Figura. 3.1 Diagrama de clase detaliată

### 3.3 Proiectarea interfețelor grafice

Această parte din etapa de proiectare are ca scop implementarea unui prototip al produsului software (prototip de interfață). În acest fel, clientul poate beneficia de o exemplificare intuitivă a modului în care aplicația arată și se comportă și facilitează primirea de feedback din partea acestuia.

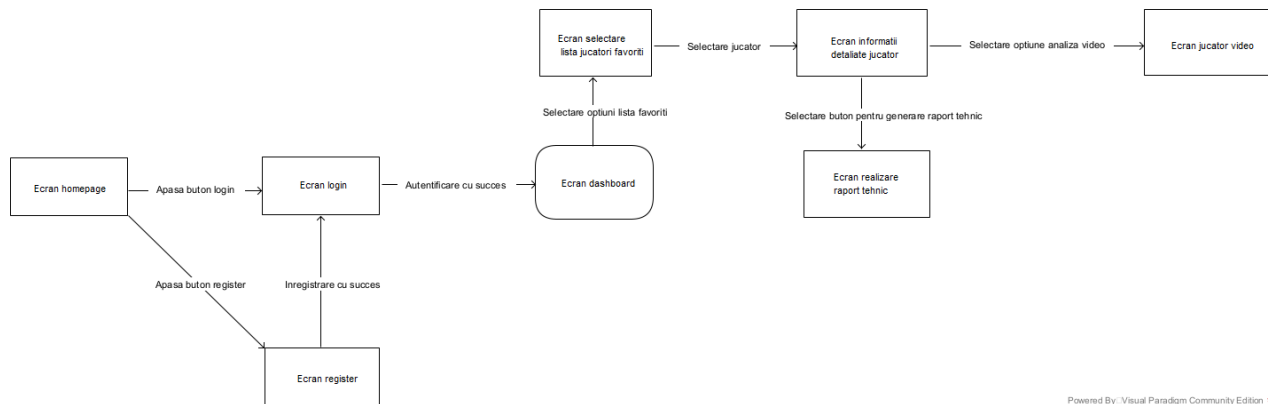
Pe de o parte, acest lucru se realizează prin hărțile de structură ale ecranului, în care este evidențiat fluxul aplicației, care este bazat pe cazurile de utilizare definite anterior. Aceste diagrame ajută la identificarea posibilelor blocaje sau confuzii în fluxul aplicației, permițând optimizarea și rafinarea designului interfeței.



*Figura 3.2 Schemă interfață grafică - Scouter*

Un utilizator din categoria *scouter*, în momentul deschiderii aplicației, poate selecta din pagina principală opțiunea de logare sau, în cazul în care nu are creat un cont, de înregistrare. În urma unei autentificări cu succes, va vizualiza un ecran cu acces la mai multe opțiuni, care coincid cu funcționalitățile principale ale aplicației. Prima din ele este aceea de a căuta jucători de fotbal, al cărei ecran poate fi urmat de cel al sportivului aferent, cu detalii referitoare la acesta și de un ecran cu opțiunea de vizualizare de videoclipuri. A doua opțiune este aceea de a căuta un club de fotbal, cu scopul de a observa jucători ai unei anumite echipe. Astfel, un ecran de căutare al cluburilor sportive va fi disponibil, urmat opțional de un ecran detaliat despre un club de fotbal. Ultimul “traseu” al scouterului este acela de a selecta și vizualiza listele de jucători marcați ca favoriți.

Următoarea figură evidențiază o altă perspectivă, și anume interfețele pe care *analistul tehnic* le va putea vizualiza în timpul interacțiunii cu aplicația. La fel ca și utilizatorul de tip *scouter*, analistul tehnic se poate autentifica și înregistra. Flow-ul aplicației este puțin simplificat după accesarea dashboard-ului, pentru că se urmărește funcționalitatea de evaluare a jucătorilor de fotbal. Această categorie de utilizatori poate accesa listele de jucători favoriți partajate de către scouter, poate selecta jucători de fotbal și poate vizualiza atât datele despre aceștia, cât și să analizeze clipuri video aferente. De asemenea, se poate accesa opțiunea de generare raport în urma observării informațiilor necesare și ulterior salva.



*Figura 3.3 Schemă interfață grafică - Analist tehnic*

După prezentarea hărților de structură a ecranului, vor fi ilustrate câteva dintre interfețele grafice principale ale aplicației. Aceste capturi de ecran vor exemplifica un punct de plecare pentru designul vizual detaliat, aranjamentul strategic al elementelor și modul de interacțiune optimizat cu utilizatorul. Astfel, se va evidenția cum designul interfeței contribuie la asigurarea unei experiențe de utilizare eficiente și plăcute.

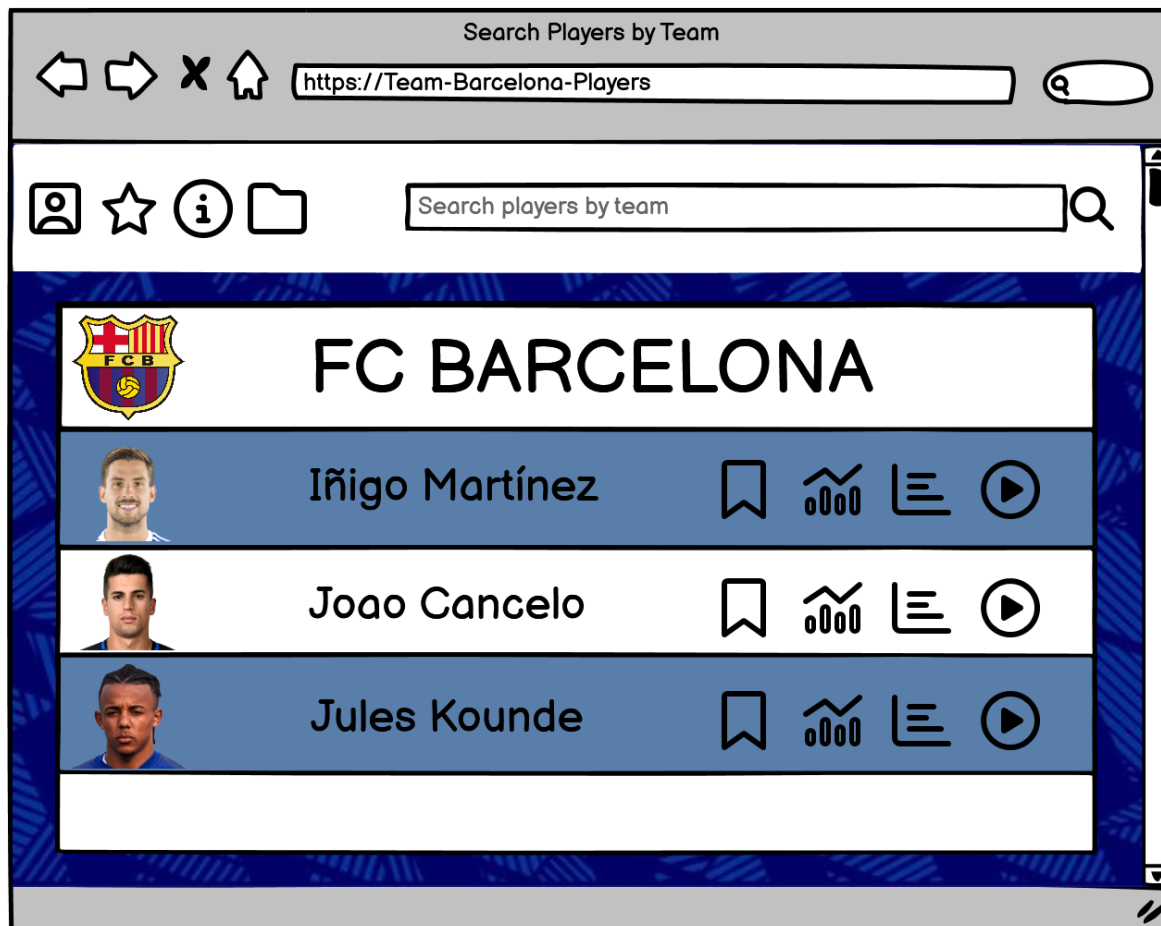
Proiectarea de interfețe a fost realizată folosind platforma *Balsamiq*. Pornind de la cerințele de funcționalitate care au fost subliniate în capitolul precedent, au fost create următoarele interfețe:

- Interfața pentru autentificarea în aplicație;
- Interfața pentru afișarea listelor de jucător;
- Interfața pentru redarea unui videoclip.



*Figura 3.4 Interfață utilizator destinată autentificării în aplicație*

Prima interfață este una destul de generică și comună pentru aplicațiile din zilele noastre, totuși una necesară. Utilizatorul are două câmpuri, de nume de utilizator și de parolă și opțiunile de a încerca să se logheze și de a alege să se autentifice pe platformă, în cazul în care nu are deja un cont și dorește crearea unui.

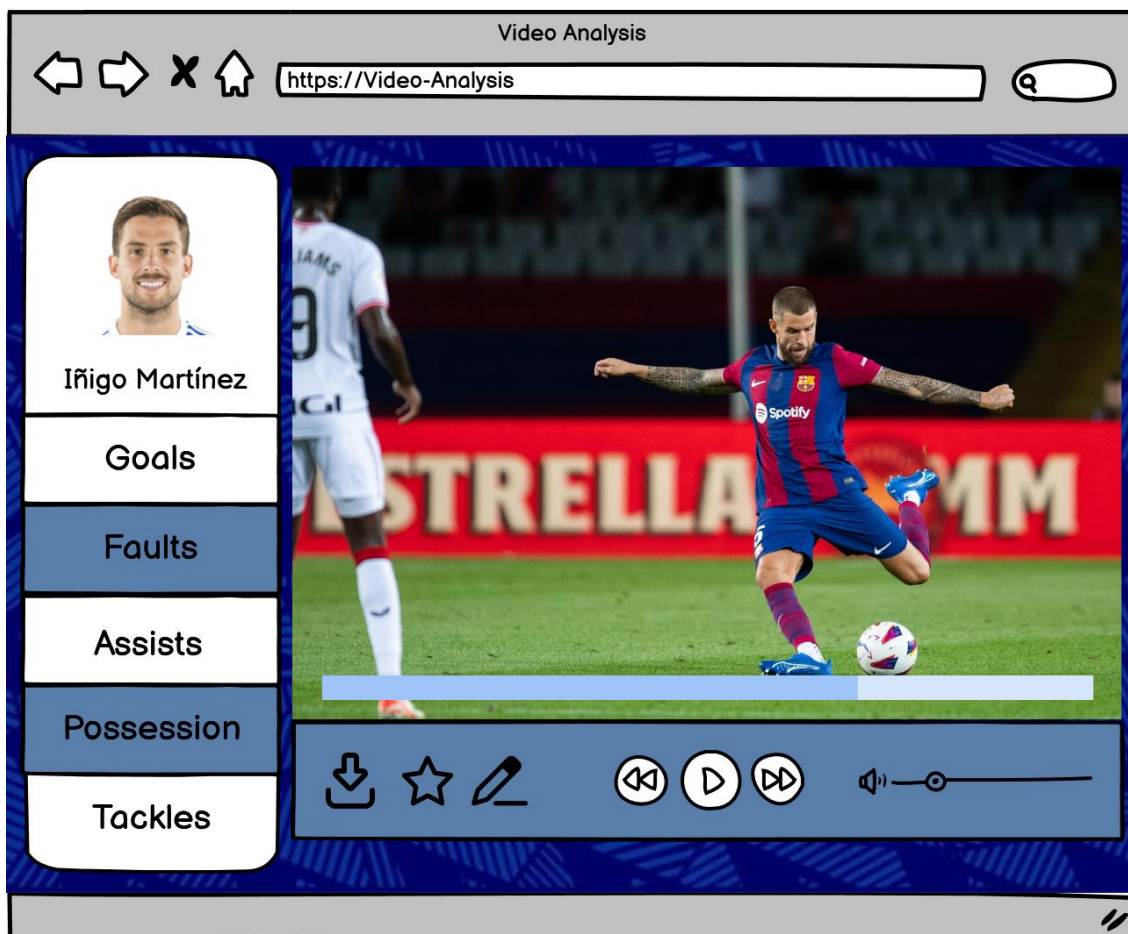


*Figura 3.5 Interfață utilizator destinată afișării unei liste de jucători în aplicație*

Figura de mai sus oferă un prototip pentru felul în care interfața utilizator pentru o listă de jucători, rezultată în urma efectuării unei căutări, ar putea arăta. Se oferă câteva informații ce au scopul de a face recunoașterea unui jucător mai ușoară, cum ar fi numele și, dacă este disponibilă, o poză cu acesta. De asemenea, există câteva comenzi rapide, cum ar fi salvarea unui jucător într-o listă de jucători favoriți, accesarea informațiilor amănunțite sau accesarea clipurilor video cu acesta.

Pe de altă parte, la nivelul următoarei reprezentări este pusă în evidență macheta ideală a unei interfețe pentru analiza video. În prim-plan se află, bineînțeles, videoclipul ce se dorește a fi analizat și comenzi specifice acestei categorii de multimedia (pauză/redare, derulare rapidă înainte/înapoi etc.), numele jucătorului care este observat în acea instanță de analiză, și opțiunea de folosire a instrumentelor de redare și desenare.





*Figura 3.6 Interfață utilizator destinată redării unui clip video*

### 3.4 Diagrama de desfășurare

Diagrama de desfășurare este o reprezentare vizuală a arhitecturii fizice a unui sistem software. Ea descrie modul în care diferite componente ale sistemului sunt distribuite pe noduri fizice sau virtuale, cum comunică aceste componente între ele și cum este gestionată infrastructura fizică a sistemului. Prin intermediul unei diagrame de desfășurare, se poate înțelege cu ușurință cum sunt conectate diversele părți ale sistemului și cum este implementată infrastructura necesară pentru funcționarea acestuia.

Diagrama de desfășurare a platformei de scouting ilustrează modul în care diferitele componente ale aplicației sunt distribuite și interconectate. La nivel de noduri, diagrama include un client (browser-ul utilizatorului), un server web pentru backend cu un server de baze de date MySQL și serviciul extern, API-ul cu care vor fi aduse informații despre jucătorii de fotbal.

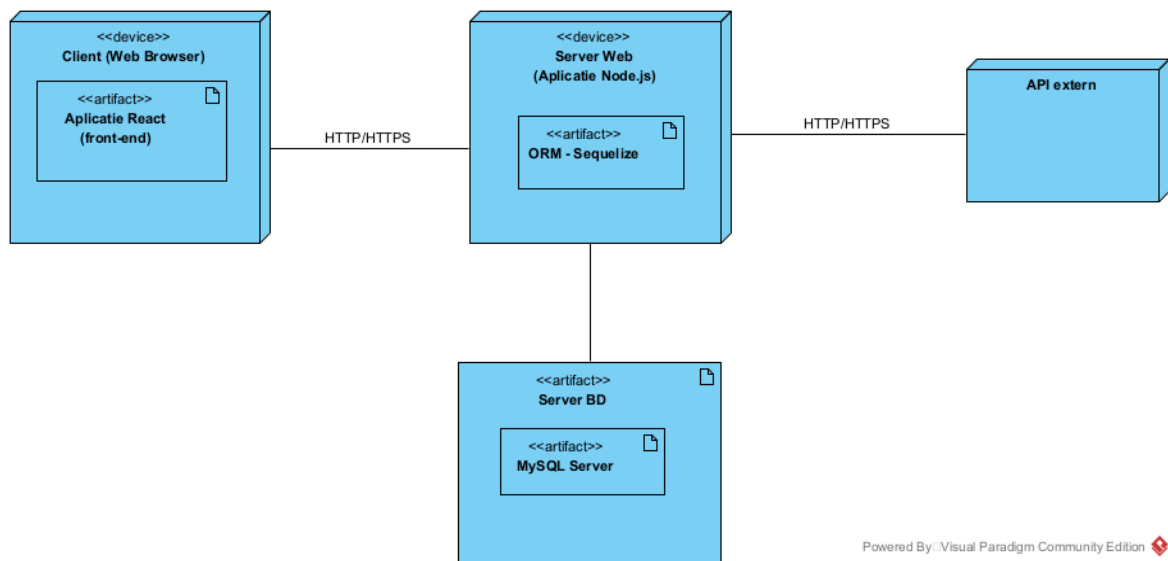


Figura 3.7 Diagrama de desfășurare

### 3.5 Diagrama de componente

În completarea diagramei de desfășurare vine diagrama de componente, al cărei scop este să pună în evidență și dependențele ce sunt prezente în cadrul sistemului unui produs software, dar și de a construi modelul arhitecturii de ansamblu și componentele locale din cadrul acesteia. Astfel, pentru a pune în prim-plan această perspectivă, a fost realizată următoarea figură:

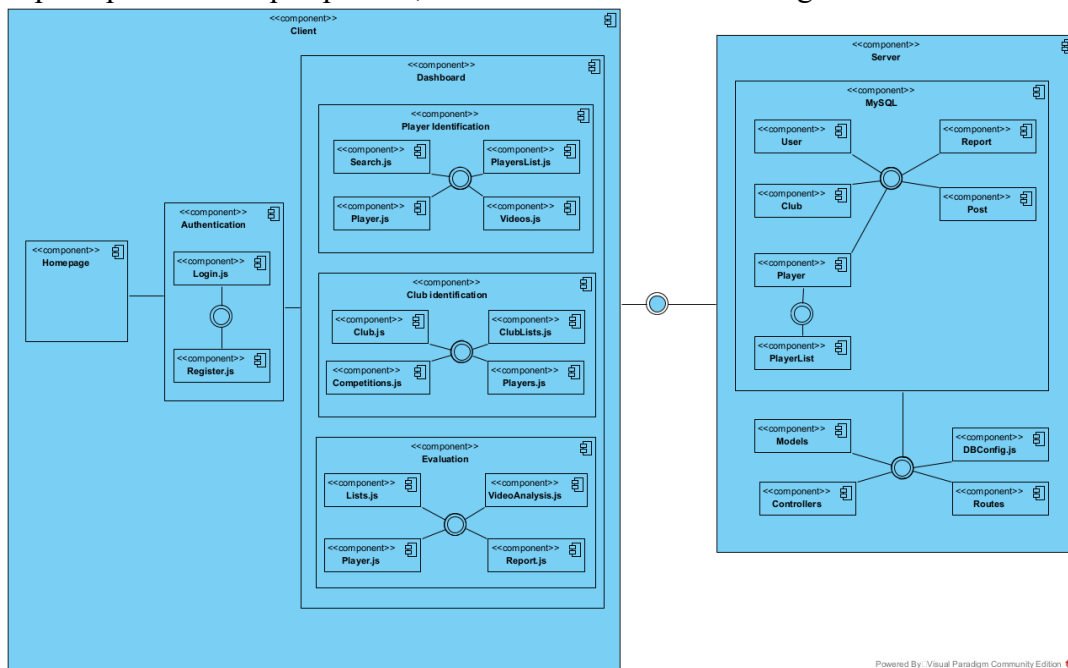


Figura 3.8 Diagrama de desfășurare

Astfel, în cadrul diagramei de mai sus se poate remarca arhitectura folosită în proiectarea sistemului informatic și anume cea de client-server. Clientul, aici o aplicație de front-end ce este rulată în cadrul unui browser reprezintă partea vizuală, cu care utilizatorul interacționează în mod direct și pune la dispoziție diferite interfețe în funcție de funcționalitățile ce se vor a fi folosite. Pe de altă parte, serverul reprezintă ceea ce se întâmplă în spate, iar utilizatorul nu poate vedea în mod direct, adică back-end-ul, care realizează diferite acțiuni specifice logicii de business, printre care prelucrarea și stocarea informațiilor în baza de date.

## Capitolul 4 - Etapa de implementare

Etapa de implementare marchează tranziția de la teorie la practică, unde designul conceptual al aplicației se transformă într-un produs funcțional. În această fază, se lucrează pentru a scrie codul efectiv care va compune aplicația, utilizând specificațiile tehnice și modelele elaborate în etapa de proiectare.

### 4.1 Prezentarea tehnologiilor utilizate

Aplicația *TalentSpotter* este realizată cu ajutorul unui set diversificat de tehnologii specifice platformei web, care permit accesarea acesteia prin intermediul unui browser. În cadrul dezvoltării acestei aplicații, am utilizat o combinație de instrumente și tehnologii moderne, fiecare contribuind la diferite aspecte ale funcționalității și performanței aplicației. Acestea includ atât limbaje de programare și de markup, cât și pachete, biblioteci și medii de dezvoltare, toate fiind alese pentru a crea o experiență de utilizare optimă și eficientă.

Pentru început, am ales să folosesc ca instrument de scriere și modificare de cod sursă probabil cea mai folosită aplicație de acest tip și anume editorul de text **Visual Studio Code**.

Visual Studio Code (cunoscut și ca VS Code) este un editor de text ce a fost dezvoltat de către Microsoft pentru cele mai cunoscute sisteme de operare. Cele mai relevante funcționalități ale acestuia sunt posibilitatea de realizare a debug-ului, versionarea codului prin Git, sublinierea cuvintelor cheie din sintaxa limbajelor folosite, posibilitatea de autocompletare a codului redactat și capacitatea de refactorizare a acestuia. (Microsoft, 2024). Totuși, probabil cel mai cunoscut avantaj pe care VS Code îl are este integrarea extensiilor, care permite programatorilor să adauge funcționalități suplimentare pentru a susține diferite limbaje de programare și fluxuri de lucru. Datorită interfeței utilizator intuitivă și a gamei largi de funcționalități disponibile, Visual Studio Code a devenit rapid un instrument preferat pentru dezvoltatorii de software din întreaga lume.

Având în vedere specificul sistemului informatic care va fi implementat, și anume o aplicație web, tehnologiile folosite de către mine pe partea de front-end sunt limbajele de markup HTML și CSS și limbajul de programare Javascript. De asemenea, pentru a dezvoltare mai facilă și în care aceste tehnologii de bază pot fi integrate împreună, voi folosi și biblioteca Javascript de front-end React.

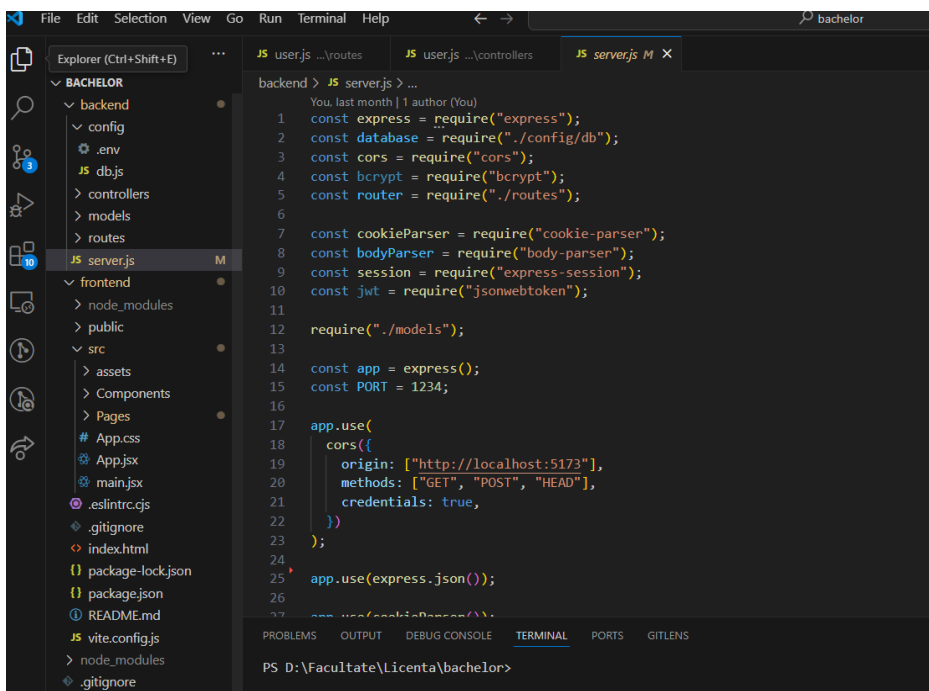


Figura 4.1 Prelucrare de cod personală folosind VS Code

**HTML** (prescurtare de la Hypertext Markup Language) reprezintă limbajul de markup utilizat pentru a oferi o structură coerentă paginilor web. HTML utilizează diferite tipuri de elemente și etichete cu scopul de a organiza conținutul web (text, imagini, audio etc.). Câteva exemple de elemente sunt: titluri, paragrafe, liste, linkuri și multe altele. HTML formează baza fiecărei pagini web și este esențial pentru orice dezvoltator web. De-a lungul anilor, HTML a evoluat pentru a include funcționalități avansate prin intermediul HTML5, care oferă suport pentru video, audio și elemente interactive fără a necesita plugin-uri suplimentare. (Mozilla, 2024).

**CSS** (prescurtare de la Cascading Style Sheets) se referă la limbajul de markup implementat cu rolul de a îmbunătăți aspectul unei pagini web (de obicei, cu o structură HTML și, mai rar, XML). Acesta controlează aspectul și stilul paginilor web, incluzând aspecte precum culorile, fonturile, spațierea și poziționarea elementelor pe pagină. Prin separarea conținutului de stil, CSS permite dezvoltatorilor să mențină și să modifice designul unui site web mult mai eficient. CSS3, cea mai recentă versiune, introduce caracteristici noi precum tranziții, transformări, animații și layout-uri flexibile, care îmbunătățesc semnificativ posibilitățile de design web (Mozilla, 2024).

**JavaScript** face parte din categoria limbajelor de programare dinamice și este folosit, în primul rând, pentru a crea și controla conținutul dinamic pe paginile web. Acest lucru include actualizări în timp real, interacțiuni cu utilizatorii, animații și multe altele. Deși creat inițial pentru a fi folosit în componenta de tip client, limbajul a evoluat spre a fi utilizat și în zona de backend, cu ajutorul platformelor precum Node.js. Cu un ecosistem vast de biblioteci și cadre (frameworks), cele mai cunoscute fiind Angular, React, și Vue.js, în mod cert JavaScript este un limbaj esențial pentru dezvoltarea web modernă (Mozilla, 2024).

**React**, una din cele mai cunoscute biblioteci JavaScript, este utilizată pentru realizarea diferitelor interfețe sau a componentelor care le alcătuiesc pe acestea. Dezvoltat de către Facebook, acesta oferă programatorilor posibilitatea să construiască aplicații de tip web rapide și interactive prin utilizarea unui model bazat pe starea și ciclul de viață al componentelor. Unul dintre avantajele majore ale React este Virtual DOM, care optimizează actualizările și randările UI pentru a îmbunătăți performanța aplicațiilor web. React este folosit pe scară largă pentru a crea aplicații de o singură pagină (Meta Open Source, 2024).

De asemenea, am ales să utilizez Material UI și React-Icons pentru reuși să optimizez dezvoltarea interfețelor utilizator și a avea o aplicație software cu un design cât mai estetic și intuitiv.

**MUI** sau Material-UI este una din bibliotecile React pentru componente ce implementează principiile Material Design dezvoltate de Google. Utilizarea MUI în aplicațiile React oferă avantajul programatorilor de a crea conținut atrăgător vizual pentru partea de front-end, respectând în același timp cele mai bune practici de design UI/UX. MUI include o gamă largă de componente pre-stilizate, cum ar fi butoane, formulare, carduri și meniuri, care pot fi personalizate pentru a se potrivi stilului și necesităților aplicației folosite. (Material UI SAS, 2024).

**React Icons** este o bibliotecă care permite integrarea ușoară a unei mari varietăți de iconițe în aplicațiile React din multe biblioteci populare precum Font Awesome, Material Design Icons, Feather Icons și multe altele. Utilizând React Icons, programatorii pot adăuga rapid acest tip de elemente de design în aplicațiile lor, îmbunătățind astfel aspectul și funcționalitatea UI. React Icons oferă o modalitate simplă de a accesa o vastă colecție de icoane și de a le utiliza în mod consistent pe tot parcursul aplicației.

**CORS** (Cross-Origin Resource Sharing) reprezintă un mecanism tip HTTP-header ce oferă sau restricționează accesul asupra resurselor ce se află pe un server web și sunt solicitate de pe un alt domeniu decât cel de pe care serverul rulează. Acest lucru este esențial pentru securitatea web, deoarece previne atacurile de tip cross-site request forgery (Mozilla, 2024).

Nu în ultimul rând, **Axios**, o bibliotecă JavaScript foarte populară, a fost folosit pentru a facilita comunicarea clientului cu serverul, prin realizarea de request-uri HTTP din browser. Axios este foarte popular datorită ușurinței de utilizare și a capacității sale de a face cereri asincrone, care sunt esențiale pentru interacțiunile cu serverul într-o aplicație web modernă. Axios suportă o gamă largă de metode HTTP, printre care GET, POST, PUT și DELETE, și permite gestionarea simplificată a răspunsurilor și a erorilor.

Pentru partea de back-end a aplicației mele, am ales să utilizez Node.js împreună cu o bază de date relațională MySQL. Utilizând aceste tehnologii, back-end-ul aplicației poate gestiona autentificarea utilizatorilor, stocarea și recuperarea datelor, și interacțiunile cu baza de date într-un mod eficient și securizat. Această combinație asigură o bază solidă pentru dezvoltarea diferitelor funcționalități implementate.

**Node.js** a fost construit pe motorul V8 al Chrome și este un mediu de execuție ce oferă posibilitatea de a utiliza codul JavaScript și pe back-end. Modelul său non-blocant, ce are la bază conceptul de eveniment, îl face ideal pentru aplicațiile scalabile și rapide (Node.JS, 2024). Acesta este

folosit pentru a crea diferite categorii de servere, aplicații sau API-uri (interfețe de programare a aplicațiilor). Ecosistemul său, cu pachetul **npm** sau **Node Package Manager**, pune la dispoziție dezvoltatorilor o multitudine de module și biblioteci care pot fi implementate astfel încât să fie adăugate funcționalități diverse aplicațiilor Node.js (OpenJSFoundation, 2024).

Alături de Node.js am folosit **Express**, care oferă un set bogat de funcționalități pentru construirea componentei server sau a API-urilor. Express simplifică gestionarea rutelor și a comunicării prin intermediul protocolului HTTP, precum și integrarea cu diverse middleware-uri pentru funcționalități suplimentare, din care se pot aminti autentificarea și autorizarea, managementul sesiunilor, procesarea fișierelor și multe altele (OpenJSFoundation, 2024).

**MySQL** este folosit pentru a prelucra și folosi bazele de date de tip relațional, cunoscut pentru performanța, fiabilitatea și ușurința în utilizare. Este utilizat pe scară largă pentru gestionarea datelor structurate și oferă funcționalități esențiale pentru stocarea, interogarea și gestionarea datelor (Oracle, 2024). Pe de altă parte, **Sequelize** este un ORM (Object-Relational Mapping) pentru Node.js, care facilitează interacțiunea cu baze de date relaționale. Prin utilizarea Sequelize, dezvoltatorii pot defini modele de date folosind JavaScript și pot efectua operațiuni CRUD (Create, Read, Update, Delete) fără a scrie manual interogări SQL.

Tehnologiile de login și autorizare pe care am decis să le utilizez includ câteva instrumente și biblioteci care facilitează autentificarea utilizatorilor. Sistemul de autentificare aderă la conceptele specifice **JWT**, și au fost implementate folosind `javascriptwebtoken`. Odată ce autentificarea se realizează fără erori, serverul oferă un **token** de tip JWT ce include diferite detalii legate de utilizator. Token-ul este direcționat clientului și stocat, de obicei, într-un cookie (pe care am ales să îl folosesc și eu) sau în local storage. Pentru fiecare cerere ulterioară către server, clientul trimite tokenul JWT în antetul cererii. Serverul verifică autenticitatea tokenului și, dacă este valid, permite accesul la resursa solicitată.

**Cookies** sunt mici fragmente de date stocate pe client care păstrează informațiile de sesiune. Acestea sunt folosite pentru a menține utilizatorii autentificați pe parcursul vizitelor pe un site web. Biblioteca pe care am folosit-o pentru hashing-ul parolelor este **bcrypt**, care asigură faptul că parolele stocate sunt securizate și greu de descifrat. Bcrypt aplică un algoritm de hashing care include un salt pentru a oferi protecție împotriva atacurilor de tip rainbow table. **Body-parser** este un middleware pentru Node.js care analizează datele din corpul solicitării HTTP și le face disponibile în obiectul `req.body`.

Pentru a reuși să pot aduce în partea de client diferite date despre jucători și despre cluburile de fotbal am ales să folosesc un serviciu extern, și anume **TheSportsDB**, care să îmi permită să pot realiza asta într-un mod facil și cu fotbaliști din cât mai multe competiții. **TheSportsDB** este un API care furnizează informații detaliate despre sporturi, inclusiv fotbal. Acesta este utilizat pe scară largă pentru dezvoltarea de aplicații sportive, site-uri de știri sportive și dashboard-uri de analiză sportivă. Dezvoltatorii pot accesa datele prin endpoint-uri RESTful și pot utiliza aceste informații pentru a îmbunătăți experiența utilizatorilor în aplicațiile lor. În completarea acestuia am folosit un alt serviciu extern și anume **transfermarkt-api**, cu scopul de a aduce statistici cât mai relevante despre jucători și alte date complementare care au fost necesare în aplicația mea.

## 4.2 Implementarea părții de front-end

Partea cu care am dorit să încep dezvoltarea aplicației a fost cea de front-end, deoarece am dorit să pot asigura în primul rând implementarea unei experiențe de utilizator cât mai plăcute înainte de a reuși să implementez și logica de back-end. Un alt argument pentru care am dorit să încep cu partea de client este pentru că o parte din aplicația mea necesită aducerea unor date folosind un serviciu extern, motiv pentru care mi s-a părut mai intuitiv să am o parte cât mai mare din front-end finalizată, întrucât modificarea acesteia ar fi mai facilă, în cazul în care apar schimbări în componenta de server.

Folosind biblioteca de Javascript **React**, paginile au fost realizate sub formă de componente, care la rândul lor au putut avea în cadrul acestora mai multe componente mai mici (cum ar fi de exemplu un navbar) care ar putea fi refolosite.

Astfel, figura de mai jos redă într-un mod succint organizarea aplicației de front-end, care conține: fișierul de module node, fișierele de configurații, dar și fișierul root HTML, fișierul de *assets* unde se află diferite elemente media folosite (în special imagini de tip png), fișierul *components* unde există diferite elemente de interfață mai mici denumite componente și, nu în ultimul rând, fișierul *pages*, unde se află toate componentele ce reprezintă pagini ale aplicației, împreună cu fișierul *App.js*, al cărui rol este de a realiza rutarea elementelor mele.

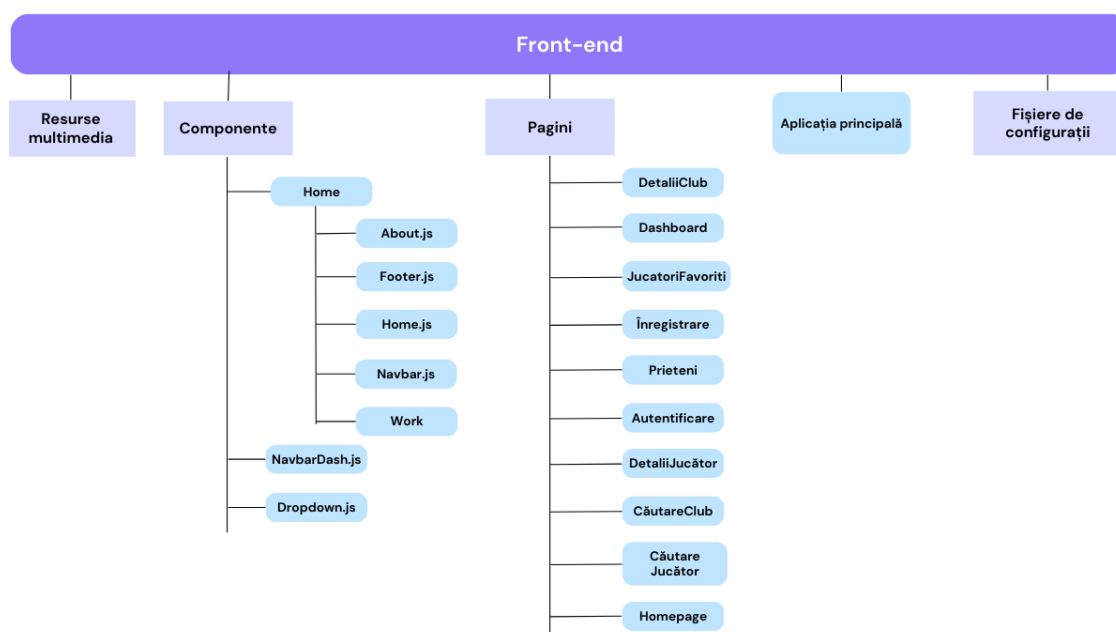


Figura 4.2 Prezentarea organizării componentei client

Pentru a gestiona transferul de date dintre diferitele pagini ale aplicației, au fost folosite diverse elemente care au scopul de a prelua informația, dar și diferite butoane sau alte elemente generatoare de evenimente.

Pe parcursul implementării aplicației, în special în pagina de prezentare, am folosit resurse multimedia cu scopul de a oferi un aspect cât mai bun aplicației. Prin intermediul elementului de tip imagine din HTML, adică `<img>`, imaginile pot fi adăugate unei pagini web și modificate prin intermediul atributelor astfel încât să respecte anumite specificații.

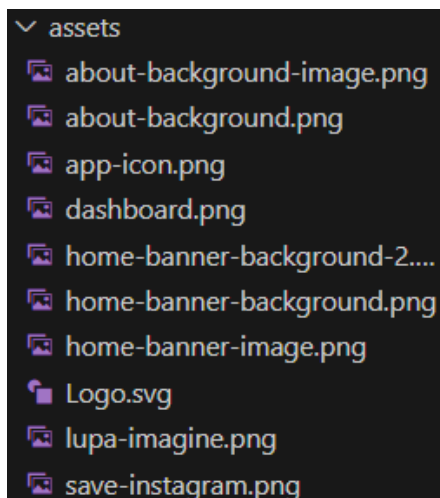


Figura 4.3 Fișierul pentru resurse multimedia

În cadrul aplicației, modalitatea care a fost folosită pentru a realiza rutarea diferitelor pagini a fost utilizarea bibliotecii react-router-dom, care ușurează navigarea între acestea. După importarea tuturor componentelor necesare și crearea unui router folosind `createBrowserRouter` și `createRoutesFromElements`, rutele au fost definite, iar prin utilizarea `RouterProvider`, obiectul de tip `route` este furnizat aplicației.

```
const router = createBrowserRouter(  
  createRoutesFromElements(  
    <Route>  
      <Route index element={<HomePg />}></Route>  
      <Route path="/login" element={<LoginForm />}></Route>  
      <Route path="/register" element={<Register />}></Route>  
      <Route path="/home" element={<Main />}></Route>  
      <Route path="/dashboard" element={<Dashboard />}></Route>  
      <Route path="/search/player" element={<SearchPlayer />}></Route>  
      <Route path="/search/club" element={<SearchClubs />}></Route>  
      <Route path="/club-details/:transfermarktId" element={<ClubDetails />} />  
      <Route  
        path="/player-details/:playerId/:transfermarktId"  
        element={<PlayerDetails />}  
      />  
      <Route path="/friends/:userId" element={<FriendComponent />} />  
      <Route path="/players-list" element={<FavoritePlayersList />} />  
    </Route>  
  )  
)
```

Figura 4.4 Routing-ul aplicației în fișierul App.js



Cu ajutorul bibliotecii Javascript Axios, au putut fi implementate în componenta de client operațiile CRUD, mai exact crearea, citirea, actualizarea și, nu în ultimul rând, ștergerea datelor ce sunt furnizate și transferate în cadrul aplicației.

Una din cele mai folosite metode HTTP din cadrul aplicației sunt metodele de tip „get()”, deoarece pentru a putea aduce utilizatorilor în interfață datele necesare realizării procesului de scouting este nevoie de apelarea rutelor care folosesc serviciile externe implementate. Un lucru de avut în vedere este faptul că cererile de acest tip au la bază folosirea de *promises*, adică de obiecte care reprezintă rezultatul și valoarea unei operații asincrone (mdn web docs, 2024). Pe acest tip de obiecte este construit și mecanismul de *syntactic sugar* denumit *async await*, ce este adesea folosit pe parcursul implementării aplicației și face codul asincron să fie mult mai ușor de citit și modificat.

În exemplul de mai jos este pusă în evidență o cerere de tip GET, care are rolul de a realiza căutarea de cluburi de fotbal după numele acestora, și de a returna rezultatul în componenta aferentă. Se poate remarca faptul că se face o cerere către serviciul TheSportsDB, unde variabila *searchTerm* este cuvântul cheie după care se realizează căutarea:

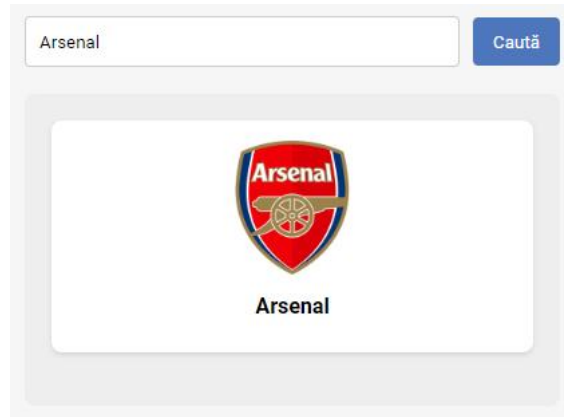
```
const handleSearch = async () => {
  try {
    const response = await axios.get(
      `https://www.thesportsdb.com/api/v1/json/3/searchteams.php?t=${searchTerm}`
    );
    setTeams(response.data.teams);
  } catch (error) {
    console.error("Error fetching teams:", error);
  }
};
```

Figura 4.5 Realizarea unei cereri de tip GET în cadrul aplicației client

Pentru a fișă toate echipele de club care au fost găsite în funcție de numele căutat, este necesară preluarea datelor rezultate din cererea HTTP și utilizarea lor pentru afișarea în interfață a cluburilor de fotbal care corespund căutării.

```
<div className="search-results">
  {teams.map((team) => (
    <div
      key={team.idTeam}
      className="team-result"
      onClick={() => handleClubClick(team.idTeam, team.strTeam)}
    >
      <div className="team-details">
        <img src={team.strBadge} alt={team.strTeam} width={100} />
        <p>{team.strTeam}</p>
      </div>
    </div>
  ))}
</div>
```

Figura 4.6 Afișarea datelor rezultate unei cereri GET în front-end



*Figura 4.7 Afișarea datelor rezultate unei cereri GET în front-end, interfața aferentă*

În cadrul aplicației, aceste tipuri de cereri nu au fost, bineînțeles, singurele utilizate. Un alt tip de funcționalitate implementată este cea de creare de obiecte. Spre exemplu, atunci când un utilizator dorește să adauge un alt utilizator în rețeaua lui de prieteni, o cerere „post()” este realizată, cu scopul de a popula în baza de date rețeaua de prietenie aferentă.

```
const handleAddFriend = async (friendId) => {
  try {
    const token = localStorage.getItem("token");
    const response = await axios.post(
      `http://localhost:1234/api/friend/add/${userId}/${friendId}`,
      {},
      {
        headers: {
          "x-access-token": token,
        },
      }
    );
    if (response.data.message === "Friend added successfully") {
      setIsFriend({
        ...isFriend,
        [friendId]: true,
      });
    }
  } catch (error) {
    console.error("Error adding friend:", error);
  }
};
```

*Figura 4.8 Realizarea unei cereri de tip GET în cadrul aplicației client*

De asemenea, la nivelul aplicației se realizează și ștergeri de date, prin metoda HTTP de tip „delete()”. În secvența de cod care evidențiază această operație CRUD, metoda apelează o rută de tip *POST*, dacă un jucător este considerat ca fiind de interes și o rută de ștergere, în cazul în care jucătorul nu mai este relevant pentru utilizator. Anterior ștergerii, în corpul cererii sunt preluate datele legate de jucător în cadrul acestei componente, identificatorul pentru user și cel pentru jucător fiind cele necesare realizării acestei operații.

```

const toggleFavorite = async () => {
  try {
    const url = isFavorite
      ? "http://localhost:1234/api/player/remove"
      : "http://localhost:1234/api/player/add";

    const method = isFavorite ? "delete" : "post";
    const data = {
      idTransfermarkt: transfermarktId,
      idTheSportsDB: playerId,
      name: player.strPlayer,
      team: player.strTeam,
      id: userId,
    };

    await axios({
      method,
      url,
      data,
    });

    setIsFavorite(!isFavorite);
  } catch (error) {
    console.error("Error toggling favorite:", error);
  }
};

```

Figura 4.9 Realizarea unei cereri de tip DELETE în cadrul aplicației client

Una dintre facilitățile pe care biblioteca React le oferă este posibilitatea de a realiza diferite componente ce pot fi reutilizate pe parcursul aplicației, avantajul fiind faptul că este necesar mai puțin cod de scris decât în cazul realizării unui front-end „clasic”, adică fără folosirea unui framework sau biblioteci Javascript.

Una dintre componentele care au fost utilizate în mai multe pagini ale aplicației este *bara de navigare*, denumită în cadrul secvențelor de cod *NavbarDash*, pentru a o diferenția de cea implementată în cadrul paginilor de prezentare, login și înregistrare. Aceasta are diferite stiluri CSS aplicate și câteva link-uri ce permit utilizatorului să navigheze spre diferite pagini ale aplicației. Cu ajutorul atributului specific React, *className*, similar cu atributul *class* din CSS, se permite manipularea stilurilor elementelor HTML. În exemplul de mai jos, este modificat felul în care conținutul barei de navigare va fi afișat, adică centrat atât vertical, cât și orizontal, folosind un container flexibil, dar și mărimea și grosimea fontului și distanța între link-urile navbar-ului.

```

return (
  <nav className="navbar-dash">
    <div className="nav-logo-container">
      <img src={Logo} alt="" />
    </div>
    <div className="navbar-links-container">
      <Link to="/dashboard">Acasă</Link>
      <Link to="/home">Despre</Link>
      <Link to="/home">Contact</Link>
      <Link to="/login">
        <button className="primary-button">Delogare</button>
      </Link>
    </div>
  </nav>
)

```

Figura 4.10 Bara de navigare a aplicației implementată în front-end

```

nav {
  display: flex;
  align-items: center;
  justify-content: space-between;
  min-height: 90px;
}
.navbar-menu-container {
  display: none;
}
.navbar-links-container a {
  margin-right: 3rem;
  text-decoration: none;
  color: black;
  font-size: 1.1rem;
  font-weight: 600;
}

```

Figura 4.11 Stiluri CSS aplicate componentei Bară de navigare

Pe de altă parte, autentificarea utilizatorului are în vedere redirecționarea acestuia în pagina principală a aplicației, dacă o confirmare în urma verificării credențialelor în baza de date este primită. Funcția *handleLogin* este declanșată atunci când butonul de login este apăsător. Pe lângă prevenirea comportamentului de bază, acela de a reîncărca pagina, prin *e.preventDefault()*, se face o cerere de tip POST către endpoint-ul de autentificare, prin care sunt trimise datele de conectare ale utilizatorului. Dacă autentificarea este reușită, utilizatorul este redirecționat către tabloul de bord, iar token-ul de logare și id-ul utilizatorului sunt salvate în *localStorage*.

```

const handleChangeLogin = (e) => {
  const { name, value } = e.target;
  setLoginData((prevData) => ({
    ...prevData,
    [name]: value,
  }));
};

axios.defaults.withCredentials = true;

const handleLogin = (e) => {
  e.preventDefault();
  axios
    .post("http://localhost:1234/api/user/login", loginData)
    .then((response) => {
      navigate("/dashboard", { replace: true });
      localStorage.setItem("token", response.data.token);
      const userId = response.data.result.id;
      localStorage.setItem("userID", userId);
    })
    .catch((error) => {
      console.error("Error logging user: ", error);
    });
};

```

Figura 4.12 Gestionarea procesului de autentificare în client

### 4.3 Implementarea părții de back-end

Aplicația de scouting a jucătorilor de fotbal are integrată și o componentă de server, care este necesară pentru a realiza operațiile asupra datelor folosite, adică stocarea, modificarea, ștergerea și retrimirerea acestora, după caz, în partea de interfață. Backend-ul are astfel rolul de a defini reprezentarea modelelor de date, tipurile de operații ce pot fi realizate pe acestea, precum și rutele pe care acestea vor fi apelate și tipul de cerere. În plus, serverul are responsabilitatea de a integra corespunzător serviciile externe folosite pentru a aduce diferite date relevante despre jucători sau cluburi în front-end.

În figura de mai jos este reprezentată structura aplicației de back-end, care în cadrul acestei lucrări a cuprins următoarele tipuri de fișiere ce respectă arhitectura MVC (Model-View-Controller): **model**, ce are rolul de a defini și gestiona relațiile dintre diferite tipuri de date; **config**, unde sunt selectate atât dialectul de manipulare a datelor, în cazul de față MySQL, cât și conexiunea efectivă la baza de date și host-ul pe care serverul va rula; **controllers** reprezintă directorul cu metodele executate pe fiecare model de date în parte, mai precis diferitele cereri de tip HTTP provenite din front-end; **routes** este directorul unde sunt specificate rutele unde operațiile pe diferitele modele de date vor fi efectuate, iar fișierul Javascript **server.js** implementează diferite mecanisme de bază pentru crearea și configurarea aplicației de back-end, cum ar fi denumirea portului, introducerea mecanismului CORS sau a librăriilor pentru autentificare și logare.

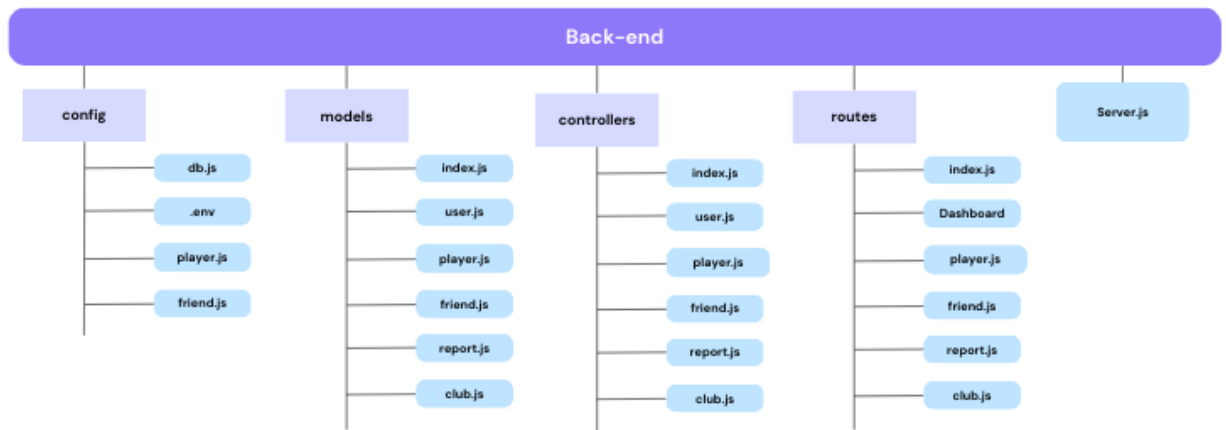


Figura 4.13 Prezentarea organizării componentei server

Deoarece este folosit un sistem de baze de date relațional, fiecare model necesar funcționării aplicației într-un mod optim reprezintă, de fapt, o tabelă de date care are diferite câmpuri ce descriu entitatea respectivă. Pentru implementarea bazei de date MySQL este utilizat ORM-ul (Object-Relational-Mapping) Sequelize.

```

const { DataTypes } = require("sequelize");
const database = require("../config/db");

module.exports = (database) => {
  const model = database.define("user", {
    firstName: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    lastName: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true,
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true,
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  });

  country: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  role: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  organisation: {
    type: DataTypes.STRING,
    allowNull: false,
  },
});

return model;
};

```

Figura 4.14 Prezentarea modelului obiectului de tip Utilizator

Pentru a oferi un exemplu de entitate care a fost modelată, a fost prezentat obiectul de tip *utilizator*, necesar pentru autentificarea și apelarea diverselor endpoint-uri din cadrul aplicației. Conform acestuia, fiecare dintre utilizatorii înregistrați are 8 câmpuri, cum ar fi nume, prenume, adresa de e-mail sau parola și au scopul de a stoca datele de identificare ale acestora. În plus, Sequelize creează automat și un câmp de tip întreg ce reprezintă numărul de identificare pentru înregistrare din tabelă. În această manieră au fost create modele de date similare pentru fiecare dintre modelele prezente în aplicația de back-end.

În ceea ce privește fișierele de *control*, acestea implementează mecanisme specifice pentru manipularea resurselor prezente în cadrul aplicației. De aceea, metodele HTTP din partea clientului vor fi primite și prelucrate de către server. La fel ca și în cazul modelelor, fiecare controller este corespondent entității din cadrul sistemului.

Prima metodă exemplificată este una de tip GET și servește preluării și returnării unei liste de jucători favoriți unui utilizator specific din baza de date. Pentru acest lucru este necesar ID-ul userului logat, care este preluat ca și parametru din URL-ul request-ului. În funcție de acesta, sunt direcționați pentru a fi afișați în interfață toți jucătorii ce au fost salvați anterior.

```
getFavoritePlayers: async (req, res) => {
  try {
    userId = req.params.userId;
    const favoritePlayers = await PlayerDb.findAll({
      where: { userId },
    });
    res.status(200).json(favoritePlayers);
  } catch (error) {
    console.error("Error removing favorite player:", error);
    res.status(500).json({ error: "Error removing favorite player" });
  }
},
```

Figura 4.15 Metodă de tip GET implementată în fișierul obiectului Jucător

În aplicație există totuși și metode de tip GET care nu aparțin unei entități, deci nu trebuie să realizeze în mod inițial modificări în baza de date. Aceste metode sunt cele care lucrează cu serviciile externe implementate și au scopul de a aduce în interfață datele de interes despre jucători și cluburile de fotbal:

```

app.get("/api/club-players/:id", async (req, res) => {
  const clubId = req.params.id;
  try {
    const playersResponse = await axios.get(
      `https://transfermarkt-api.fly.dev/clubs/${clubId}/players`
    );

    const players = playersResponse.data.players;

    const playerProfiles = await Promise.all(
      players.map(async (player) => {
        const profileResponse = await axios.get(
          `https://transfermarkt-api.fly.dev/players/${player.id}/profile`
        );

        const profile = profileResponse.data;

        return {
          transfermarktId: profile.id,
          name: profile.name,
          image: profile.imageURL,
        };
      })
    );

    res.json({ players: playerProfiles });
  } catch (error) {
    console.error("Error fetching player profiles:", error);
    res.status(500).json({ error: "Error fetching player profiles" });
  }
});

```

*Figura 4.16 Metodă de tip GET care lucrează cu un serviciu extern*

Aici, ID-ul clubului este obținut din URL-ul solicitării, care este utilizat pentru a obține lista tuturor jucătorilor unui club respectiv. Răspunsul este rezultatul operației de mapare a listei de jucători și extragerea datelor relevante ale acestora. Prin urmare, profilurile detaliate ale jucătorilor sunt returnate în format JSON.



```

addFriend: async (req, res) => {

  const user = await UserDb.findByIdPk(userId);
  const friend = await UserDb.findByIdPk(friendId);

  if (!user || !friend) {
    return res.status(404).json({ error: "User or friend not found" });
  }

  if (userId === friendId) {
    return res
      .status(400)
      .json({ error: "You cannot add yourself as a friend" });
  }

  const existingFriendship = await FriendDb.findOne({
    where: {
      userID: userId,
      friendID: friendId,
    },
  });

  if (existingFriendship) {
    return res
      .status(400)
      .json({ error: "You are already friends with this user" });
  }

  await FriendDb.create({ userID: userId, friendID: friendId });

  res.json({ message: "Friend added successfully" });
} catch (error) {
  res.status(500).send("Server error!" + error.message);
}
},

```

Figura 4.17 Metodă de tip POST în aplicația de back-end

Această metodă are rolul de a gestiona cereri de tip POST la ruta pentru adăugarea unui prieten în aplicație. ID-urile celor 2 utilizatori, cel care adaugă și prietenul adăugat sunt extrase din parametrii URL-ului. Dacă utilizatorii sunt găsiți în baza de date, iar relația de prietenie nu există, o nouă relație de prietenie va fi creată.

Pe de altă parte, metoda *removeFavoritePlayer* este folosită pentru a șterge un jucător favorit din baza de date. Este nevoie de ID-ul userului și de ID-ul jucătorului care trebuie eliminat din lista de favorite. Se folosește metoda specifică de eliminare din cadrul modelului, și anume *destroy*.

```

removeFavoritePlayer: async (req, res) => {
  try {
    const { idTransfermarkt, id } = req.body;
    console.log(id);
    const deletedPlayer = await PlayerDb.destroy({
      where: {
        idTransfermarkt,
        userId: id,
      },
    });

    if (deletedPlayer) {
      res.status(200).json({ message: "Favorite player removed" });
    } else {
      res.status(404).json({ error: "Favorite player not found" });
    }
  } catch (error) {
    console.error("Error removing favorite player:", error);
    res.status(500).json({ error: "Error removing favorite player" });
  }
},

```

Figura 4.18 Metodă de tip DELETE în aplicația de back-end

Nu în ultimul rând, *server.js*, fișierul principal al aplicației de back-end, a fost construit pentru a inițializa și configura un server Express. Sunt implementate diferite module, cum ar fi *cors*, ce facilitează prelucrarea cererilor care provin din alt domeniu decât cel al aplicației, *bodyparser*, un middleware ce permite analiza body-urilor cererilor HTTP sau *cookieParser*, un alt middleware, dar folosit pentru manipularea cookie-urilor.

```

const express = require("express");
const database = require("../config/db");
const cors = require("cors");
const router = require("../routes");
const axios = require("axios");
const cookieParser = require("cookie-parser");
const bodyParser = require("body-parser");
const session = require("express-session");

require("../models");

const app = express();
const PORT = 1234;

app.use(
  cors({
    origin: ["http://localhost:5173"],
    methods: ["GET", "POST", "HEAD", "DELETE"],
    credentials: true,
  })
);

app.use(express.json());

app.use(cookieParser());

app.use(bodyParser.urlencoded({ extended: true }));

```

Figura 4.19 Conținut al fișierului principal al aplicației de back-end

## Capitolul 5 - Prezentarea părții aplicative

**TalentSpotter** reprezintă partea aplicativă a lucrării de față: un sistem de scouting pentru jucătorii de fotbal ce are ca obiectiv eficientizarea acestui proces prin furnizarea informațiilor și partajarea acestora între utilizatori într-un mod cât mai intuitiv. De menționat este faptul că aceasta se adaptează la mărimea ecranului dispozitivului pe care este utilizată, pentru a permite utilizatorilor să o folosească într-un context cât mai diversificat.

Fluxul informațional al lucrării se bazează, în general, pe folosirea datelor primite pentru a avea o cât mai bună informare și o capacitate de decizie crescută în vederea achiziționării jucătorilor.

Datorită acestui lucru, modalitatea de prezentare pe care am ales-o este una care trece prin majoritatea funcționalităților implementate și va avea atașate capturi de ecran aferente acestora.

Prima interfață pe care un utilizator o va vizualiza în momentul accesării aplicației este o pagină de prezentare, ce oferă o introducere clară referitoare la funcționalitățile și beneficiile aplicației. De aceea, o descriere generală a aplicației și sublinierea scopului acesteia, facilitarea identificării jucătorilor de fotbal este pusă în prim-plan. De asemenea, sunt integrate și imagini atractive pentru îmbunătățirea experienței utilizatorului. În plus, există posibilitatea de a naviga atât spre opțiune de autentificare, dacă un cont a fost anterior creat sau opțiunea de creare a unui cont nou pe platformă.

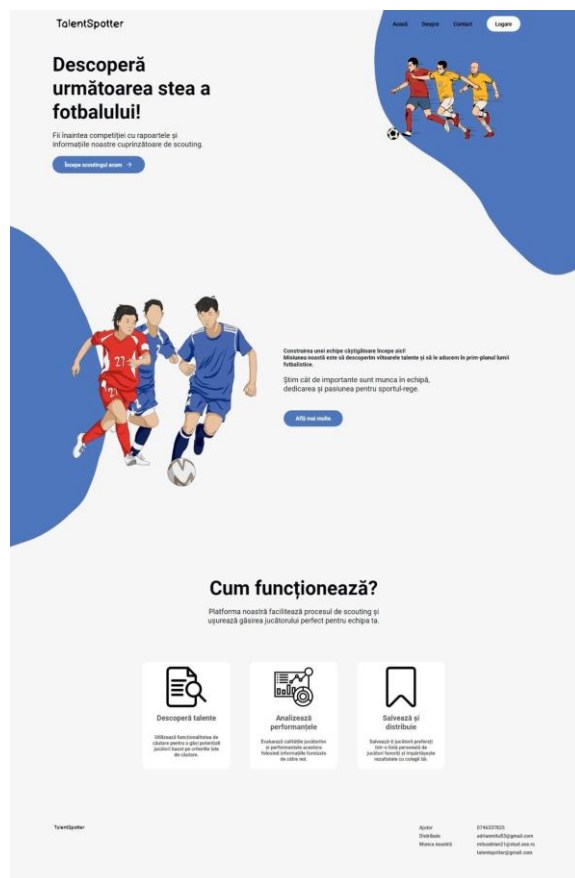
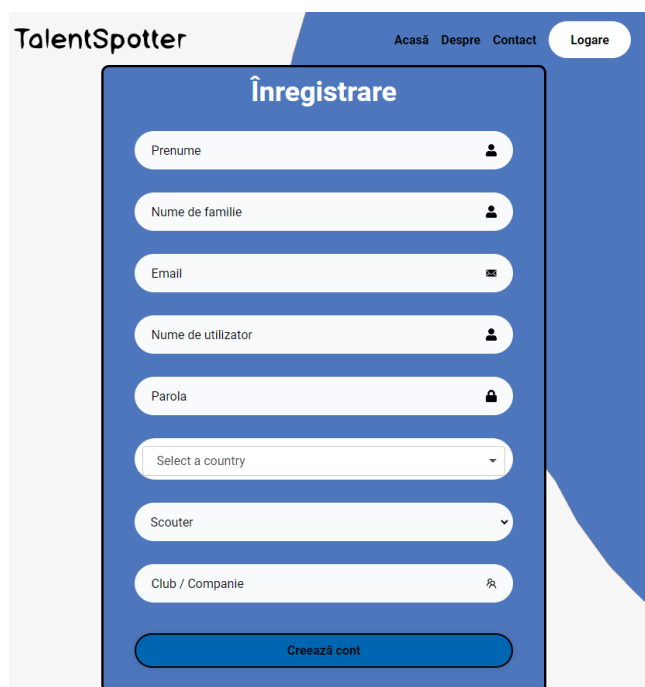


Figura 5.1 Pagina de prezentare a aplicației

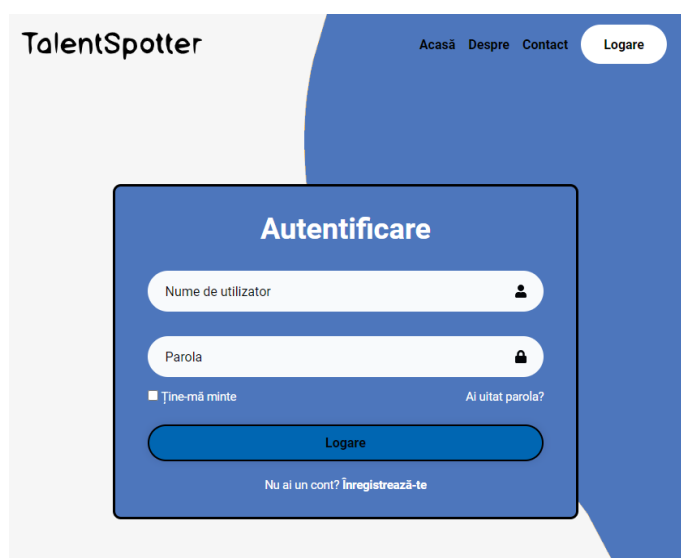
Prima funcționalitate pe care un utilizator nou o va face atunci când va începe să folosească platforma este să se înregistreze în baza de date a aplicației. De aceea, acesta va completa formularul de înregistrare:



The image shows a web page for 'TalentSpotter'. At the top, there is a navigation bar with links: 'Acasă', 'Despre', 'Contact', and a 'Logare' button. The main content area is titled 'Înregistrare' (Registration). It contains a vertical stack of input fields: 'Prenume' (First Name), 'Nume de familie' (Last Name), 'Email', 'Nume de utilizator' (Username), 'Parola' (Password), 'Select a country' (a dropdown menu), 'Scouter' (a dropdown menu), and 'Club / Companie' (Company). Each field has a small icon to its right. At the bottom of the form is a blue button labeled 'Creează cont' (Create account).

*Figura 5.2 Pagina de înregistrare utilizator a aplicației*

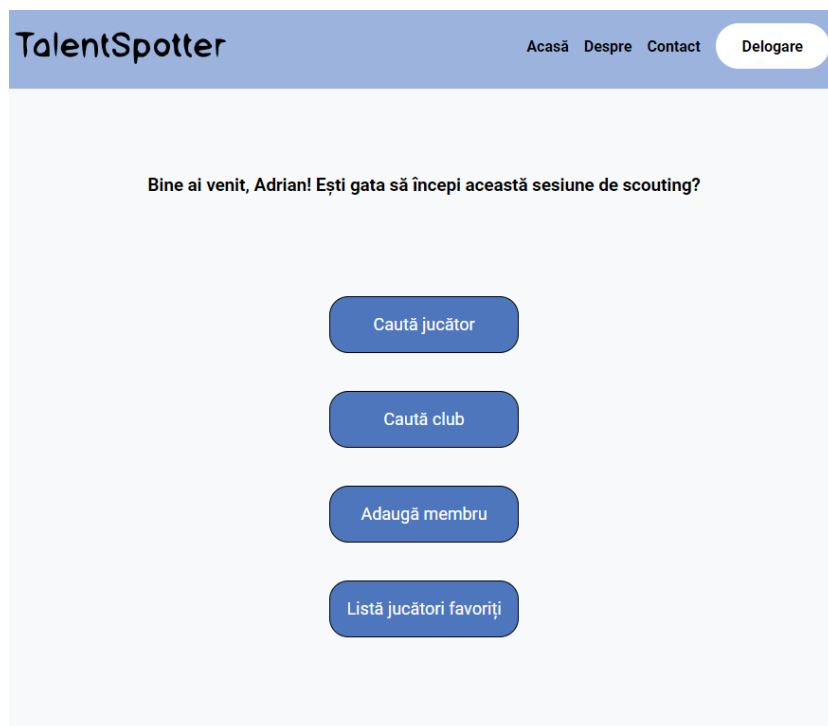
În pagina de autentificare, prin completarea unui formular ce are câmpurile de nume de utilizator și parolă, cei care folosesc aplicația pot accesa pagina principală a aplicației, prin care pot fi accesate funcționalitățile acesteia.



The image shows a web page for 'TalentSpotter'. At the top, there is a navigation bar with links: 'Acasă', 'Despre', 'Contact', and a 'Logare' button. The main content area is titled 'Autentificare' (Authentication). It contains a vertical stack of input fields: 'Nume de utilizator' (Username) and 'Parola' (Password). Below the password field are two links: 'Ține-mă minte' (Remember me) and 'Ai uitat parola?' (Forgot your password?). At the bottom of the form is a blue button labeled 'Logare' (Login). Below the button is a link: 'Nu ai un cont? Înregistrează-te' (Don't have an account? Register).

*Figura 5.2 Pagina de autentificare a utilizatorului în aplicație*

Pagina centrală a aplicației, denumită Dashboard, este construită pentru a oferi un punct central pentru opțiunile disponibile utilizatorilor. După logarea cu succes, un mesaj de bun venit personalizat, care include numele utilizatorului este afișat. Astfel, prin intermediul butoanelor, se pot accesa principalele direcții din cadrul sistemului: căutare de jucători, căutare de cluburi, adăugare de prieteni și posibilitate de vizualizare a listelor de jucători favoriți.



*Figura 5.3 Pagina de înregistrare utilizator a aplicației*

Odată selectată prima opțiune, pot fi efectuate căutări după numele jucătorilor de fotbal, iar rezultatele obținute vor fi afișate pe câte un card separat, ce are numele jucătorului, echipa actuală și o imagine a acestuia.

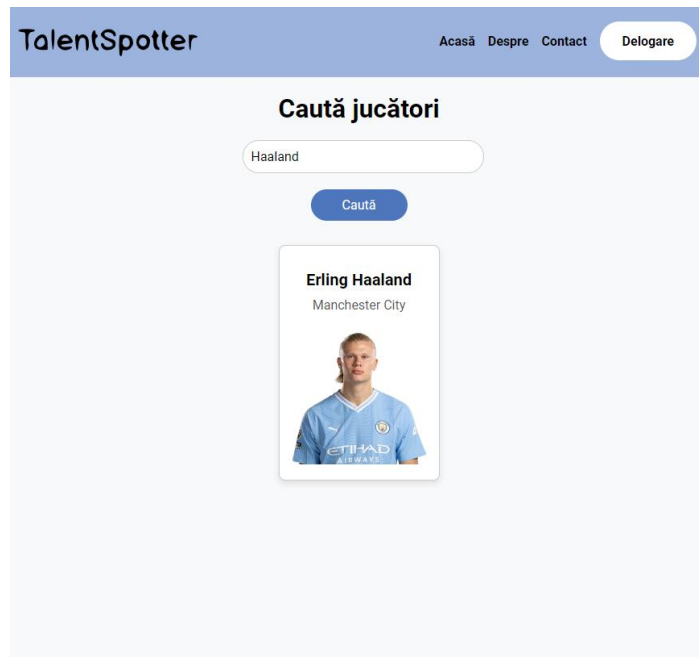


Figura 5.4 Pagina de căutare a jucătorilor

Dacă jucătorul este selectat, se va afișa o pagină detaliată referitoare la jucătorul în sine. Pe primul card, sunt prezentate câteva date generale despre acesta, cum ar fi poziția, înălțimea, greutatea, valoarea pe piața transferurilor sau naționalitatea. În plus, există câteva date suplimentare, cum ar fi o descriere a jucătorului în limba engleză, realizările și premiile acestuia, precum și o serie de statistici referitoare la performanța acestuia.

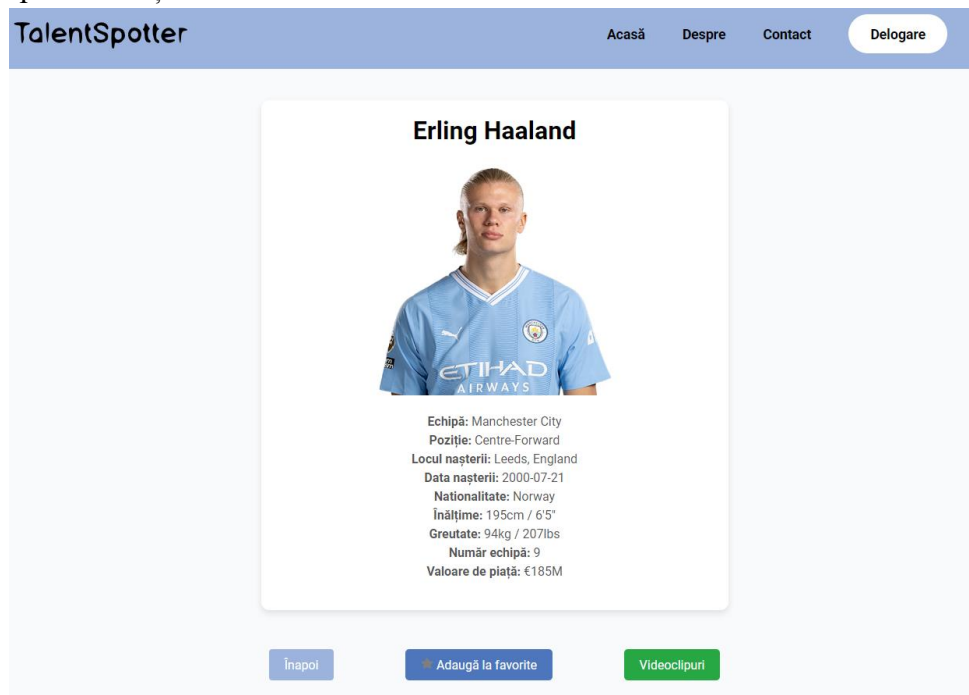
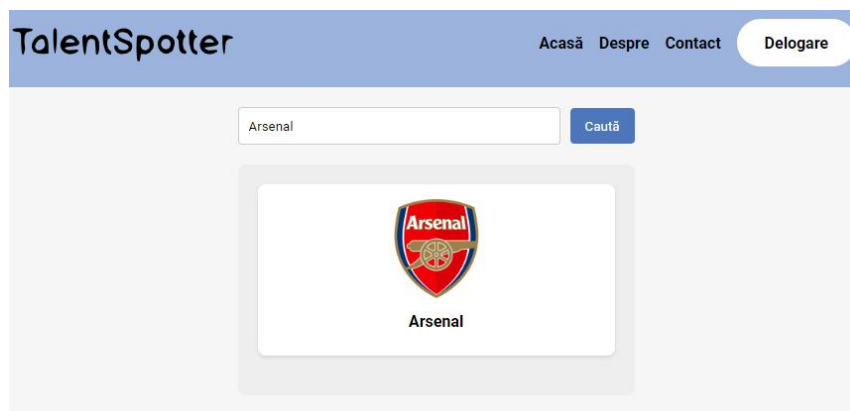


Figura 5.5 Pagina detaliată a unui jucător

Statistici						
Competiție	Sezon	Apariții	Goluri	Assist-uri	Cartonașe galbene	Minute jucate
FA Cup	23/24	3	5	-	-	254'
Premier League	23/24	31	27	5	1	2.558'
Champions League	23/24	9	6	1	-	779'
EFL Cup	23/24	-	-	-	-	-
UEFA Super Cup	23/24	1	-	-	-	90'
Community Shield	23/24	1	-	-	-	64'
Champions League	22/23	11	12	1	1	846'
FA Cup	22/23	4	3	-	-	311'
Premier League	22/23	35	36	8	5	2.777'
EFL Cup	22/23	2	1	-	-	107'
Community Shield	22/23	1	-	-	-	90'

*Figura 5.6 Statistici despre jucător preluate din pagina detaliată a acestuia*

A doua opțiune din cadrul meniului principal presupune căutarea unui club de fotbal în vederea afișării datelor relevante despre acestea. Mecanismul este destul de similar cu cel implementat în cadrul căutării de jucători de fotbal.



*Figura 5.6 Pagina de căutare a unui club*

Mai important însă pentru procesul de scouting, se oferă posibilitatea selectării tuturor sportivilor dintr-o anumită echipă, în cazul în care un scouter sau un analist tehnic este interesat să observe jucători dintr-un anumit lot.

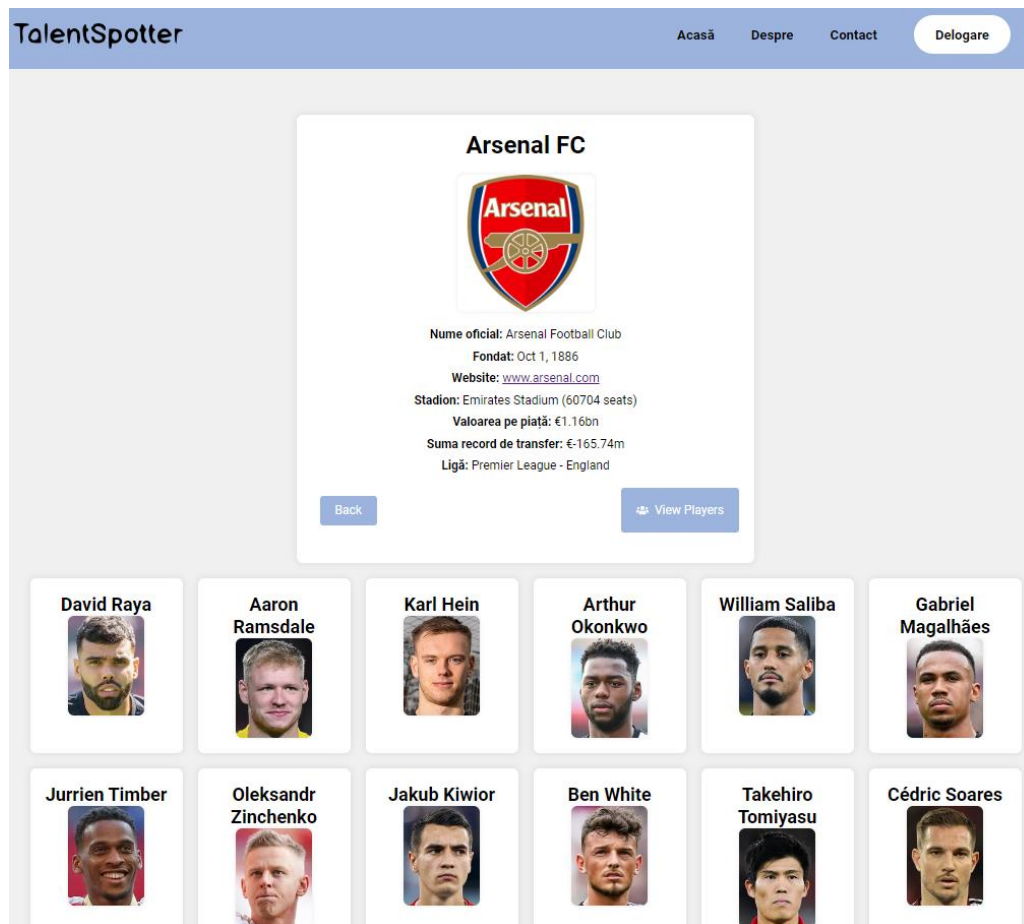


Figura 5.7 Pagina detaliată a unui club de fotbal și jucătorii acestuia

Altă posibilitate pe care un utilizator o are este să își mărească rețeaua de prieteni, cu scopul de a partaja cu aceștia jucători care sunt de interes. Pentru aceasta, se poate căuta un utilizator în baza de date, iar acesta poate fi adăugat sau șters din lista de prieteni.

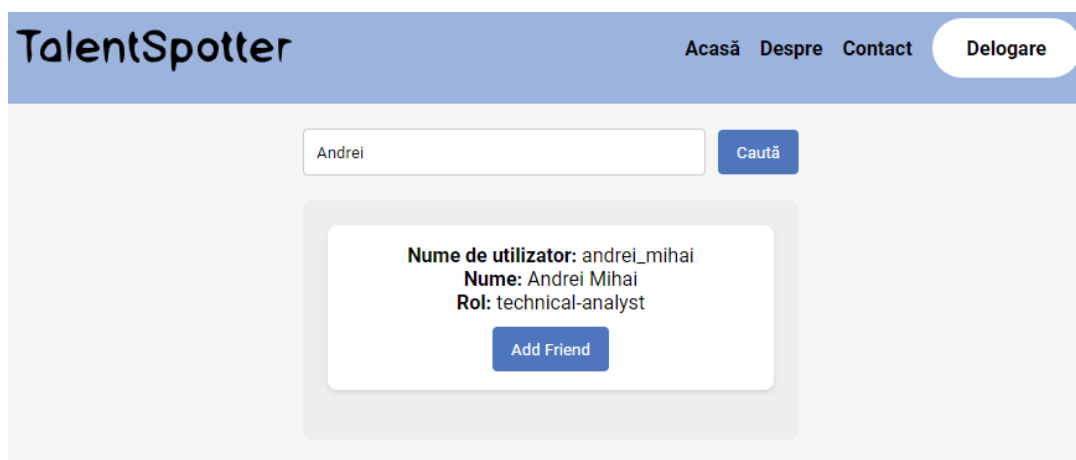
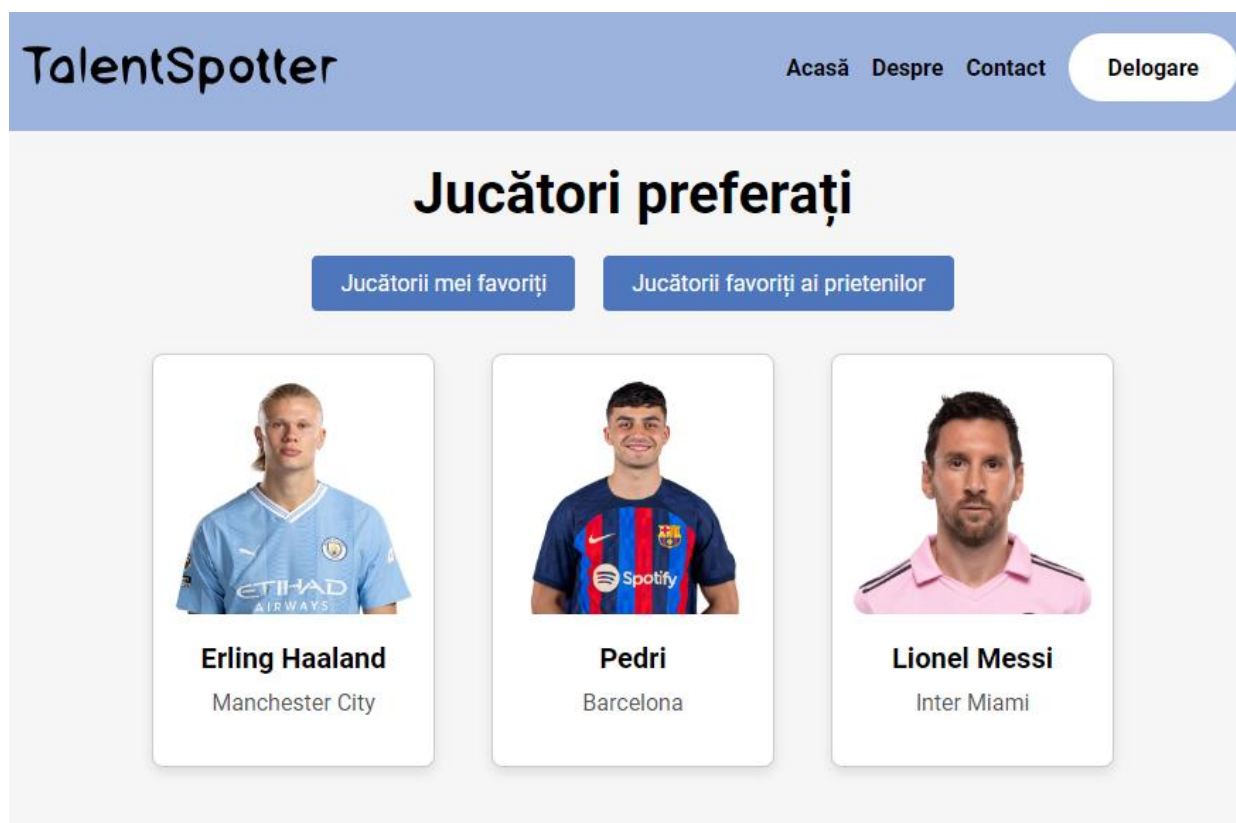


Figura 5.8 Pagina de adăugare/ștergere din lista de prieteni



Pentru a afișa listele de jucători favoriți, sunt folosite datele utilizatorului care sunt stocate pe parcursul sesiunii. În funcție de acesta, pot fi afișați jucătorii care au fost marcați ca și preferați atât de utilizatorul în sine, cât și de prietenii săi.



*Figura 5.9 Pagina de jucători favoriți*

## Concluzie

Principalul scop al acestei lucrări a fost combinarea mai multor funcționalități prezente și în cadrul altor aplicații disponibile pe piață cu unele noi, în vederea creării unei aplicații care să fie cât mai intuitivă, completă și care să poată fi un sprijin real în procesul de căutare și de evaluare a jucătorilor de fotbal. În special în contextul evoluției digitale și a transferului de informație accelerată, cred că o astfel de unealtă poate fi foarte utilă atât pentru diferiți profesioniști din zona de recrutare, precum scouteri, analiști tehnici sau antrenori și, de ce nu, a jurnaliști, arbitrii, analiști sportivi sau chiar simplii entuziaști ai fotbalului.

Prima parte a lucrării de față a avut în prim-plan realizarea unei cercetări în domeniu, pentru a analiza și documenta, pe de o parte, știința din spatele procesului de scouting și a scoate la suprafață realitățile acestei componente a lumii fotbalului. Pe de altă parte, este prezentat și impactul pe care îl are tehnologia în recrutarea viitoarelor talente și necesitatea ca modalitatea obișnuită, aceea de a viziona în persoană un meci de fotbal, să fie dublată de folosirea unor aplicații care să permită

observarea jucătorilor dintr-o perspectivă analitică, prin prisma cifrelor și a diferitelor date pe care performanțele acestora le generează sau sunt deja cunoscute din diferite surse.

În continuare, este urmărit procesul prin care aplicației i se definesc funcționalitățile și este supusă la diferite scenarii, fapt reliefat prin diferitele diagrame din etapa de analiză, toate cu scopul de a scoate în evidență diferitele cazuri limită sau posibile erori care ar putea fi prevenite în procesele ulterioare.

În etapa de proiectare se urmărește construirea scheletului unei aplicații web care respectă arhitectura client-server și se bazează pe tehnologiile specifice, folosind limbajul Javascript în întreaga aplicație, și care integrează un serviciu extern capabil să faciliteze obținerea și prelucrarea datelor referitoare la jucătorii de fotbal.

Ultimele două etape pun în evidență pe larg tehnologiile utilizate pentru dezvoltarea aplicației, dar și prezentarea flow-ului aplicației și felul în care au fost integrate diferite elemente de cod în cadrul frontendului și al backendului.

În ceea ce ține de extinderea funcționalităților, una dintre acestea ar putea fi crearea, îmbunătățirea sau adăugarea unor alte servicii de obținere a datelor despre jucători, deoarece acesta a fost un element mai dificil în implementarea funcționalității de căutare a jucătorilor de fotbal. De asemenea, integrarea sistemelor de vizualizare a datelor, implementarea unor algoritmi de inteligență artificială pentru predicția performanței sau pentru analiza tehnologiilor disponibile și dezvoltarea unor module de comunicare și colaborare pentru utilizatori sunt direcții valide pentru evoluția viitoare a platformei.

Așadar, scopul final al cercetării a fost atins, acela de a crea o aplicație care să sprijine procesul de scouting. Principalele funcționalități de care acesta dispune sunt posibilitatea de a căuta jucători, de a obține informații și date relevante despre aceștia pentru a realiza o evaluare obiectivă a acestora.

## Bibliografie

- [1] Shantanu Ghar, Sayali Patil, Venkhatesh Arunachalam - *Data Driven football scouting assistance with simulated player performance extrapolation*, 2021
- [2] Stephan Wolf, Maximilian Schmitta and Bjorn Schuller, *A football player rating system*, vol. 6, no. 4, pp. 243-257, 2020
- [3] Tom L. G. Bergkamp, Wouter G. P. Frencken, A. Susan M. Niessen, Rob R. Meijer & Ruud. J. R. den Hartigh (2022) How soccer scouts identify talented players, *European Journal of Sport Science*, 22:7, 994-1004
- [4] Elena Radicchi, Michele Mozzachiodi *Social Talent Scouting: A New Opportunity for the Identification of Football Players?*, p. 28 - 43, May 09, 2016
- [5] Marković, Srđan, Ivan Ćuk, and Aleksandar Živković. 2020. *The Impact of Information Technologies on the Scouting Process in Sports Games*. Belgrade, Serbia: Sinteza.
- [6] Microsoft (2024). Visual Studio Code Documentation Page:  
<https://code.visualstudio.com/docs>
- [7] Mozilla (2024), MDN Web Docs, *HTML: HyperText Markup Language*:  
<https://developer.mozilla.org/en-US/docs/Web/HTML>
- [8] Mozilla (2024), MDN Web Docs, *CSS: Cascading Style Sheets*: <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [9] Mozilla (2024), MDN Web Docs, *JavaScript*: <https://developer.mozilla.org/en-US/docs/Web/HTML>
- [10] Meta Open Source(2024), *React Documentation*: <https://react.dev/>
- [11] Material UI SAS (2024), MUI Core:  
<https://mui.com/material-ui/getting-started/>
- [15] Mozilla(2024), *MDN web docs, Cross-Origin Resource Sharing (CORS)*:  
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- [12] OpenJS Foundation (2024), *About Node.js*: <https://nodejs.org/en/about>

[13] OpenJS Foundation (2024), *Express.js documentation*: <https://expressjs.com/>

[14] Oracle (2024), *MySQL documentation*: <https://dev.mysql.com/doc/>

[15] Mozilla(2024), *MDN web docs, Promises*: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise)