

UNIVERSITATEA “ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

“Find Football Friends”

propusă de

Cucuteanu Lucian-Andrei

Sesiune: iulie, 2025

Coordonator științific

Asist. drd. Ioniță Alexandru

UNIVERSITATEA “ALEXANDRU-IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

“Find Football Friends”

Cucuteanu Lucian-Andrei

Sesiune: iulie, 2025

Coordonator științific

Asist. drd. Ioniță Alexandru

Avizat,
Îndrumător Lucrare de Licență
Asist. drd. Ioniță Alexandru

Data _____ Semnătura _____

DECLARAȚIE privind autenticitatea conținutului lucrării de licență

Subsemnatul Cucuteanu L. Lucian-Andrei domiciliat în Jud. Iași, Com. Tomești, Str. Trandafirului, Nr. 8, Bl. 8, Sc. A, Et. 1, Ap 7, născut la data de 27.02.2003, identificat prin CNP 5030227226807, absolvent al **Universității „Alexandru Ioan Cuza” din Iași, Facultatea de Informatică**, specializarea Informatică, promoția 2025, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul: Find Football Friends elaborată sub îndrumarea dlui Asist. drd. Ioniță Alexandru,

este autentică, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop. Declar că lucrarea de față are exact același conținut cu lucrarea în format electronic pe care profesorul îndrumător a verificat-o prin intermediul software-ului de detectare a plagiatului.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens declar pe proprie răspundere că lucrarea de față nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

Data _____

Semnătură student _____

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Find Football Friends*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent Cucuteanu L. Lucian-Andrei

Data: _____

Semnătura: _____

Cuprins

1. Introducere	3
1.1 Motivație	3
1.2 Contribuție	3
1.3 Aplicații Similare.....	4
1.3.1 KickChat.....	4
1.3.2 TheFans	4
1.4 Tehnologii folosite.....	5
1.4.1 React	5
1.4.2 Material UI	5
1.4.3 Framer Motion și @use-gesture/react:	6
1.4.4 WebSockets.....	6
1.4.5 React-Leaflet	6
1.4.6 Django	7
1.4.7 JSON Web Tokens (JWT)	7
1.4.8 Haversine	8
1.4.9 Model NLP pentru analiza sentimentelor	8
2. Arhitectura sistemului.....	9
2.1 Prezentare generală.....	9
2.1.1 SendGrid.....	9
2.1.2 Leaflet	9
2.2 Diagrama arhitecturii sistemului.....	10
2.3 Prezentarea containerelor	10
2.3.1 Aplicație pe o Singură Pagină.....	11
2.3.2 Aplicația API	11
2.3.3 Baza de Date	12
2.3.4 Django Admin.....	12
2.4 Diagrama arhitecturii containerului.....	13
2.5 Prezentarea componentelor SPA.....	13
2.5.1 Autentificare și Înregistrare	14
2.5.2 Componenta Routing	14
2.5.3 Pagina Dashboard	15
2.5.4 Pagina Chat	15
2.5.5 Serviciul API.....	16
2.5.6 Componenta Notificări.....	16

2.5.7 Pagina Profil	17
2.5.8 Setări Profil.....	17
2.5.9 Componenta hartă	18
2.6 Diagrama arhitecturii componentelor SPA	19
2.7 Prezentarea componentelor Aplicației API.....	19
2.7.1 Componenta Users.....	19
2.7.2 Componenta Matches.....	20
2.7.3 Componenta Chats.....	21
2.7.4 Componenta Notifications	22
2.7.5 Componenta Teams	23
2.7.6 Componenta Emails	23
2.8 Diagrama arhitecturii componentelor Aplicației API	24
3. Implementarea aplicației	25
3.1 Componenta Users.....	25
3.2 Componenta Matches.....	27
3.3 Componenta Chats.....	28
3.4 Componenta Notifications	32
3.5 Componenta Teams	33
3.6 Componenta Emails	33
3.7 Alte Componente	34
3.7.1 Animația cardului pentru swipe.....	34
3.7.2 Conectarea la chat și trimiterea mesajelor	36
4. Manual de Utilizare.....	39
4.1 Autentificare și Înregistrare	39
4.2 Resetarea Parolei	40
4.3 Pagina Dashboard	42
4.4 Notificări.....	42
4.5 Pagina Profil	43
4.6 Pagina Setări Profil	44
4.7 Pagina Chat	45
4.8 Harta Interactivă	45
5. Concluzii	46
6. Bibliografie	47

1. Introducere

În ultimii ani, din cauza ritmului crescut al digitalizării, s-a schimbat semnificativ modul în care oamenii se conectează și interacționează în jurul fotbalului. Chiar dacă tehnologia oferă noi căi de comunicare, aceasta diminuează legătura autentică dintre fani, înlocuind-o cu interacțiuni superficiale și rapide.

Soluția propusă constă în dezvoltarea unei aplicații sociale ce permite utilizatorilor să interacționeze într-un mod natural. Totodată, această aplicație oferă modalitatea de a se împrieteni cu oameni ce împărtășesc aceeași pasiune pentru fotbal. Aplicația oferă posibilitatea utilizatorilor de a-și alege echipa preferată, de a cunoaște alți suporterii din apropiere printr-un sistem intuitiv de swipe și de a comunica în timp real cu aceștia.

Prin intermediul acestei aplicații, suporterii fotbalului pot crea noi prietenii și noi cercuri sociale, restabilind conexiuni autentice într-o lume din ce în ce mai digitalizată și distantă.

1.1 Motivație

Prin această aplicație, am dorit să readuc în prim-plan conexiunile autentice dintre fanii fotbalului, folosind tehnologia ca un mijloc de apropiere, nu de izolare. Platforma creează un mediu în care suporterii acestui sport pot lega prietenii și pot comunica cu alți pasionați care împărtășesc aceleași interese. Aplicația contribuie astfel, la formarea unei comunități unite în jurul pasiunii pentru fotbal.

1.2 Contribuție

Acest capitol este dedicat evidențierii contribuțiilor mele în contextul dezvoltării aplicației sociale destinate iubitorilor de fotbal. Pornind de la o idee generală propusă de îndrumător, am

realizat întregul proces de proiectare și implementare, ocupându-mă atât de backend, cât și de frontend.

Aplicația web Find Football Friends oferă un sistem de autentificare și înregistrare sigur, o interfață grafică prietenoasă, curată și scalabilă, mesagerie privată între utilizatori, posibilitatea de a alege echipa preferată, posibilitatea de a customiza profilul personal și un sistem intuitiv de swiping pentru a alege cu cine se dorește a interacționa și cu cine nu.

Ce opțiuni noi aduce Find Football Friends?

Find Football Friends vine cu o hartă interactivă pe care utilizatorul poate vedea ce alți utilizatori are în proximitate și echipa de fotbal susținută de aceștia. Aplicația vine cu posibilitatea de a alege echipa preferată și adaugă un sistem de swiping focusat pe preferințele fotbalistice. Totodată, Find Football Friends implementează un sistem ce analizează sentimentul conversației dintre doi utilizatori pentru a aminti acestora să fie politicoși, respectuoși și pozitivi.

1.3 Aplicații Similare

1.3.1 KickChat [1]

KickChat este o aplicație socială construită pentru iubitorii de fotbal. Aceasta oferă utilizatorilor comunicarea unu la unu, în grup sau audio în timp real și conține un sistem în care utilizatorii pot posta, pot comenta, pot crea voturi, și interacționa cu acestea, pot trimite poze și pot urmări alți utilizatori. Aplicația mai conține și rezultate în timp real ale meciurilor și evenimentelor, care facilitează conversațiile dintre suporterii. Totodată, aceasta conține în dotare un sistem de notificări push ce alertează utilizatorul în funcție de acțiune: a primit o urmărire de la alt utilizator, a primit un mesaj de la un alt utilizator sau detalii legate de anumite evenimente sportive.

1.3.2 TheFans [2]

TheFans este o altă aplicație socială similară, care este construită pentru suporterii ce doresc să fie prezenți pe stadioane. Aceasta prezintă utilizatorului meciurile ce vor urma în proximitatea sa, de la meciuri de fotbal locale, la evenimente de nivel înalt cum ar fi Liga

Campionilor. Aplicația permite utilizatorilor să își bifeze prezența la meciul de fotbal și să socializeze cu alți suporterii din jurul său. TheFans nu conține conversații private, ci se bazează pe comunități mai mari unde utilizatorii pot comunica deschis, pot împărtăși amintiri, poze și crea noi momente de neuitat.

1.4 Tehnologii folosite

1.4.1 React [3]

React este o bibliotecă, ce aparține limbajului JavaScript [4], care este destinată dezvoltării de interfețe grafice dinamice și interactive ale aplicațiilor software. Am folosit această bibliotecă datorită componentelor reutilizabile, performanței și flexibilității. Prin intermediul bibliotecii React am construit întreaga interfață a aplicației și a componentelor acesteia.

1.4.2 Material UI [5]

Material UI este o bibliotecă React de componente deja construite, gata de utilizat și stilizată. Această bibliotecă este concepută să ajute dezvoltatorii de interfețe grafice cu aceste componente scalabile și ușor de editat. Am decis să utilizez Material UI datorită integrării cu ușurință, personalizării intuitive a componentelor, gamei variate de modificări disponibile și a design-ului scalabil. Printre componentele pe care le-am folosit se numără: butoane, avatare, cutii, containere, iconițe, stive, o listă virtualizată, text, un dialog pentru confirmare, o bară de căutare cu autocompletare și o casuță de bifat.

Câteva exemple relevante:

- Buton:



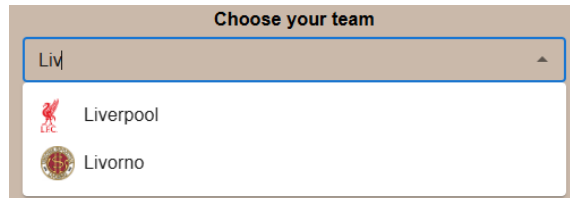
- Avatar:



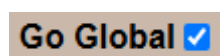
- Iconițe:



- Bara de căutare cu autocompletare:



- Căsuța de bifat:



1.4.3 Framer Motion [6] și @use-gesture/react [7]:

Framer Motion este o librărie de animații React. Am utilizat-o pentru a îmbunătăți swiping-ul cartonașelor utilizatorilor disponibili. @use-gesture/react este o altă librărie din React ce se ocupă cu acțiuni de tipul swiping sau tragere. Am utilizat-o pentru a face posibil swiping-ul cartonașelor. Folosite împreună, cele două librării creează o animație estetică și fluentă a cardurilor.

1.4.4 WebSockets [8]

WebSockets oferă un canal de comunicare peste o singură conexiune TCP între server și client. Am utilizat acest canal de comunicare pentru a conecta utilizatorii la camere private în care aceștia pot comunica prin mesaje în timp real.

1.4.5 React-Leaflet [9]

React-Leaflet este o componentă ce utilizează librăria externă Leaflet [10], care nu este creată special pentru React. Dat fiind acest fapt, React-Leaflet transformă funcționalitățile librăriei Leaflet în componente reutilizabile React. Am utilizat această componentă pentru a reda o hartă dinamică și interactivă ce arată locația utilizatorului, a altor utilizatori aflați în proximitatea acestuia și clubul lor de fotbal preferat.

1.4.6 Django [11]

Django este un framework, ce aparține limbajului Python [12], care facilitează crearea aplicațiilor software într-un mod rapid și curat. Framework-ul are deja introdus în acesta funcții de autentificare, interfața de administrator și multe altele. Am decis să folosesc Django datorită numărului mare de funcții oferite și a mediului de dezvoltare rapid și curat. Prin intermediul acestui framework am manipulat și calculat toate datele din spatele interfeței grafice. În această parte este gestionată autentificarea, sunt diferite modele pentru baza de date, sunt căutați utilizatorii din proximitatea selectată de utilizator ca preferință și multe altele.

Django REST Framework [13] este o librărie Django folosită pentru a crea apeluri API. Prin intermediul acestor apeluri API interfața grafică React comunică cu serverul Django, putând prelua, trimite, actualiza sau șterge date într-un mod eficient și securizat. Am utilizat această librărie pentru a îmi conecta interfața grafică cu serverul prin intermediul apelurilor API la diferite endpoint-uri.

Django Channels [14] este o altă librărie Django ce ajută framework-ul să se ocupe de WebSockets și de funcții asincrone. Am folosit Django Channels pentru a realiza conexiunea și pentru a crea camere private în care utilizatorii pot discuta în timp real.

1.4.7 JSON Web Tokens (JWT) [15]

JSON Web Tokens sunt chei digitale sigure ce conțin informații despre utilizator. Aceste chei sunt folosite pentru autentificare și autorizare, care permit efectuarea apelurilor API securizate. Am utilizat JSON Web Tokens datorită nivelului de securitate înalt și gestionării eficiente a sesiunii utilizatorului. La o autentificare realizată cu succes, serverul generează două chei digitale: una de acces și una pentru reîmprospătarea cheii de acces. Cheia de acces expiră după treizeci de minute pentru a nu păstra sesiunea activă cât utilizatorul nu folosește aplicația. Dacă utilizatorul încă folosește aplicația și cheia expiră, intervine cheia de regenerare ce creează o nouă cheie de acces. Cheia de reîmprospătare expiră după șapte zile, iar la expirarea acesteia, utilizatorul trebuie să se autentifice din nou.

1.4.8 Haversine [16]

Haversine este o librărie care se ocupă cu formulele și calculele distanțelor dintre două coordonate de tip latitudine și longitudine ținând cont de curbura Pământului. Am utilizat Haversine pentru a calcula distanțele dintre utilizatorul principal și ceilalți utilizatori pentru a decide ce carduri îi afișăm acestuia în funcție de distanța maximă preferată.

1.4.9 Model NLP pentru analiza sentimentelor [17]

Un model NLP, sau model de procesare al limbajului natural, este un model de învățare automată învățat să interpreteze, să înțeleagă și să genereze limbaj uman. În acest caz am folosit un model pre-antrenat pe seturi de date mari, făcut să recunoască sentimentul unei conversații dintre doi utilizatori. Am utilizat modelul “clapAI/modernBERT-base-multilingual-sentiment” de pe platforma [huggingface.co](https://huggingface.co/clapAI/modernBERT-base-multilingual-sentiment) pentru a analiza și a recunoaște sentimentul ultimelor zece mesaje din conversația a doi utilizatori.

Link: <https://huggingface.co/clapAI/modernBERT-base-multilingual-sentiment>

2. Arhitectura sistemului

2.1 Prezentare generală

Find Football Friends este o aplicație web socială menită să conecteze fanii fotbalului și să creeze noi prietenii cu oamenii care împart aceleași preferințe fotbalistice. Sistemul permite utilizatorilor să își creeze un cont, să își selecteze echipa preferată de fotbal, să descopere alți suporterii din jurul său și să interacționeze cu aceștia printr-un sistem intuitiv de swiping și camera de chat.

În această aplicație, există două roluri principale de utilizator:

- Utilizatorul, cel care interacționează cu aplicația prin intermediul interfeței web.
- Administratorul, cel care poate vedea și gestiona datele sistemului.

Pentru a asigura un sistem modern de resetare a parolei contului și pentru a oferi o hartă interactivă și precisă, sistemul interacționează cu două servicii externe:

2.1.1 SendGrid [18]

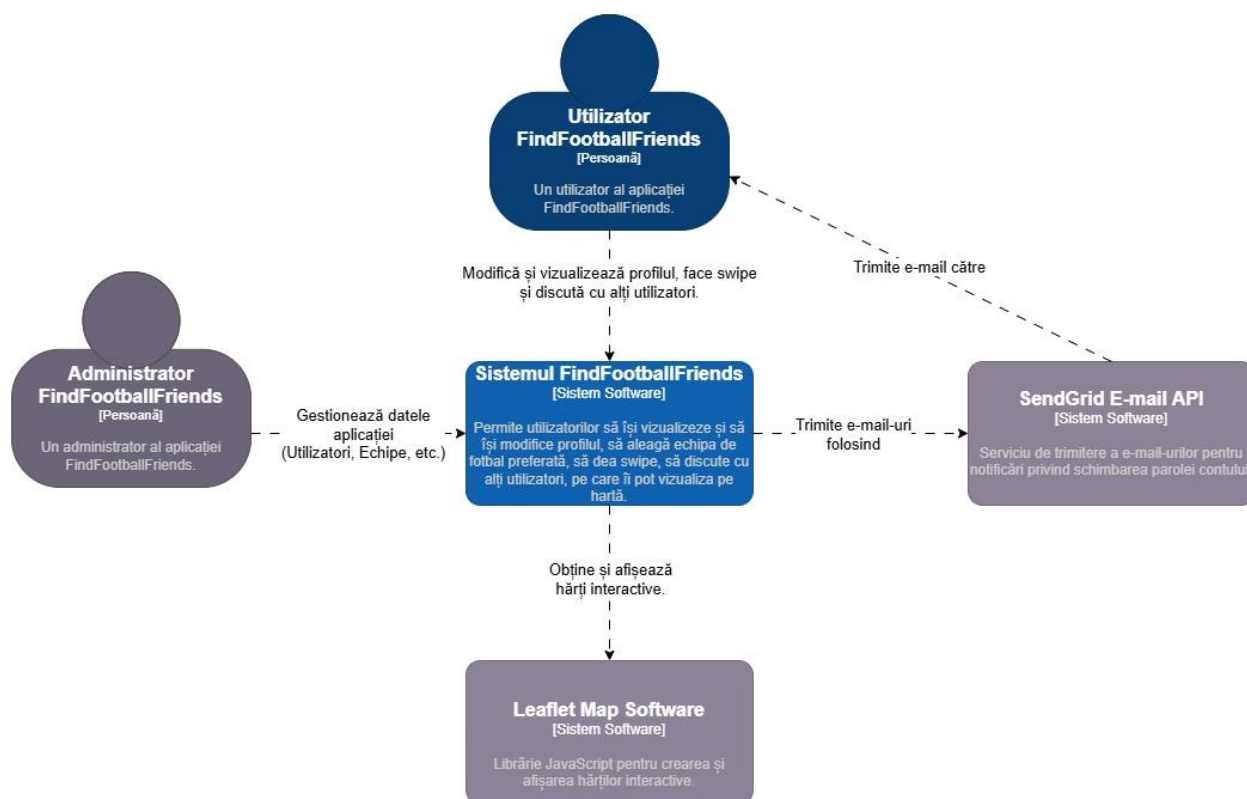
SendGrid este un serviciu de poșta electronică folosit pentru a trimite e-mail-uri către utilizatori. Acest serviciu asigură livrarea e-mail-urilor într-un mod sigur și rapid, prin intermediul unui domeniu propriu configurat. Serviciul funcționează pe orice tip de adresă validă. Am utilizat SendGrid pentru a trimite utilizatorilor aplicației Find Football Friends e-mail-uri privind resetarea parolei contului.

2.1.2 Leaflet

Leaflet este o librărie ce aparține limbajului JavaScript folosit pentru crearea și vizualizarea hărților interactive. Aceasta nu este construită direct pentru framework-ul React, dar poate fi folosită după ce librăria React-Leaflet o transformă în componente reutilizabile. Am integrat acest

sistem pentru a oferi utilizatorilor o privire de ansamblu asupra posibilităților de a crea noi conexiuni, atât locale cât și globale.

2.2 Diagrama arhitecturii sistemului



[System Context] Diagrama de Context a Sistemului pentru FindFootballFriends
Diagrama de context a sistemului FindFootballFriends

Figură 1: Diagrama arhitecturală de nivel 1

2.3 Prezentarea containerelor

Sistemul Find Football Friends este compus din mai multe containere majore, fiecare având roluri distincte și bine definite. Când interacționează între ele și lucrează împreună, într-un

mod atent structurat, aceste containere formează funcționalitatea completă a aplicației Find Football Friends.

Aceste containere fac parte din arhitectura aplicației și sunt un stâlp în dezvoltarea Find Football Friends. Fiind patru la număr, acestea sunt:

2.3.1 Aplicație pe o Singură Pagină

Containerul Aplicația pe o Singură Pagină este construit cu ajutorul tehnologiilor: limbajul JavaScript, librăria React și librăria Material UI.

O interfață web scalabilă cu o singură pagină, în care componenta activă se schimbă și se afișează, în funcție de endpoint-ul ales. Acest tip de pagină schimbă ce este afișat în mod dinamic, fără să se reîmprospăteze la fiecare schimbare. În această interfață grafică web, utilizatorul interacționează cu aceasta în mod activ prin intermediul opțiunilor intuitive: acesta își poate crea un cont, își poate vizualiza și modifica profilul, poate vizualiza utilizatorii din jurul său, atât pe carduri swipe-abile cât și pe harta interactivă și poate să comunice cu alți utilizatori pe camere de chat.

Acest container se ocupă cu obținerea și afișarea hărții interactive și cu realizarea de apeluri API către containerul Aplicația API. Aplicația pe o Singură Pagină atașează automat în header-ul apelului API o cheie digitală JWT pentru autorizare. Acest apel API este de tip HTTP și conține un obiect de tip JavaScript Object Notation, JSON.

2.3.2 Aplicația API

Aplicația API este concepută prin intermediul tehnologiilor: limbajul Python, framework-ul Django și librăria JSON Web Tokens sau JWT.

O aplicație de backend sigură, care este concepută să se ocupe de partea logică și de gestionarea datelor aplicației web Find Football Friends. Partea de gestionare de date constă în manipularea: detaliilor utilizatorului, preferințelor acestuia, echipelor de fotbal, notificărilor, swipe-urilor, împrietenirilor și a mesajelor. Partea logică are un rol de bază, deoarece, prin intermediul acesteia, aplicația conține: logica de autentificare, cea pentru trimiterea e-mail-urilor utilizatorilor pentru resetarea parolelor, cea de căutare a utilizatorilor din apropiere și cea de confirmare a împrietenirilor. Totodată, această aplicație se ocupă partea de concepere a

notificărilor în cazurile corecte și de gestionarea conexiunii cu clientul web pentru a avea o conversație cu un alt utilizator în timp real.

Aplicația API primește apelurile API făcute de către containerul Aplicație pe o Singură Pagină, le autorizează pe baza unei chei digitale de acces validă, realizează logica și după întoarcere ca răspuns codul de status. În funcție de codul de status, aplicația poate întoarce eroare și doar un mesaj aferent, sau poate întoarce succes și să trimită un obiect JSON la containerul care se ocupă cu interfața web.

Acest container citește din și scrie în containerul Baza de date. Aplicația poate prelua, modifica, scrie și șterge date din această bază de date.

Aplicația API se ocupă în mod direct de configurarea și aplicarea sistemului de poștă electronică. Acest container primește de la interfața web un obiect JSON ce conține e-mail-ul utilizatorului ce își dorește să își schimbe parola. Cu acest e-mail serverul creează un link sigur pentru a oferi siguranța utilizatorului. Acest link îl includem într-un șablon de e-mail pe care i-l trimitem mai departe clientului prin intermediul serviciului SendGrid configurat propriu.

2.3.3 Baza de Date

Containerul Baza de Date este bazat pe tehnologia SQLite3.

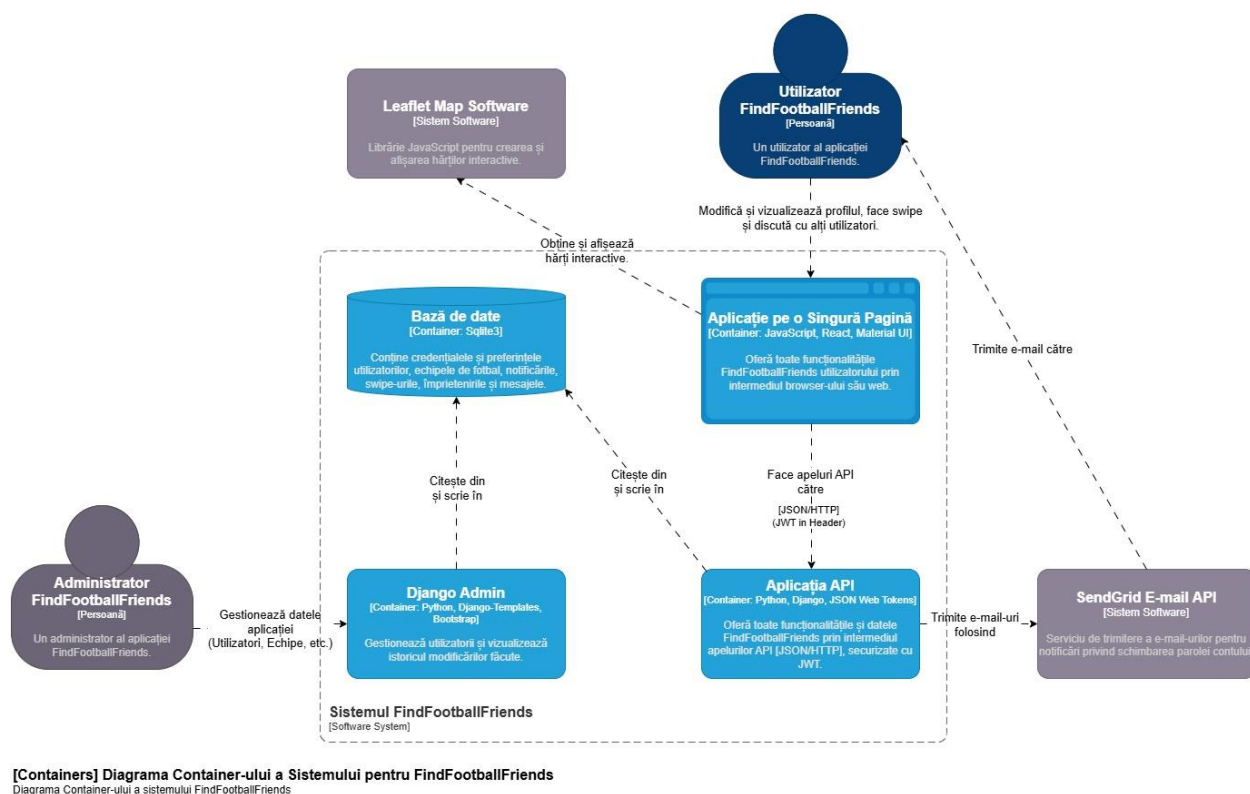
Acest container conține credențialele utilizatorilor, preferințele acestora, echipele de fotbal disponibile, notificările necitite, swipe-urile, împrietenirile și mesajele dintre utilizatori pentru a putea reține istoricul conversațiilor.

2.3.4 Django Admin

Acest container este construit prin intermediul următoarelor tehnologii: limbajul Python, librăria Django-Templates și librăria Bootstrap.

Containerul Django Admin este un panou pentru utilizatorul administrator, ce poate vizualiza și gestiona datele din baza de date. Totodată, administratorul poate vedea istoricul modificărilor făcute asupra datelor stocate.

2.4 Diagrama arhitecturii containerului



Figură 2: Diagrama arhitecturală de nivel 2

2.5 Prezentarea componentelor SPA

Termenul de Aplicație pe o Singură pagină (SPA), constă într-o singură pagină web care încarcă dinamic conținutul și componentele bazat pe interacțiunea cu utilizatorul. Acest tip de pagină web nu necesită reîncărcare a întregii pagini, deoarece, prin intermediul librăriei React-Router-DOM [19], aplicația încarcă conținutul și componentele în funcție de endpoint-ul activ.

Containerul Aplicație pe o Singură Pagină (SPA) este compus din mai multe componente importante, care, puse într-o arhitectură bine structurată, assemblează și afișează utilizatorului interfața web. Aceste componente sunt împărțite pe roluri distincte și bine structurate pentru a acoperi fiecare necesitate și fiecare opțiune. Dat fiind acest fapt, voi prezenta în detaliu fiecare componentă, deoarece acestea au un rol crucial în arhitectura interfeței web:

2.5.1 Autentificare și Înregistrare

Această componentă este construită cu ajutorul tehnologiilor următoare: limbajul JavaScript, librăria React, librăria JSON Web Tokens și librăria Material UI.

Componenta Autentificare și Înregistrare este responsabilă cu autentificarea și înregistrarea utilizatorilor pe platforma web Find Football Friends.

Autentificarea se face prin intermediul paginii de login unde utilizatorul poate completa un formular scurt ce conține doar e-mail-ul și parola necesare pentru logare. În cazul unui e-mail și parole valide, utilizatorul primește accesul la aplicație și îi sunt asignate două chei virtuale: una de acces și una pentru reîmprospătarea cheii de acces. Această cheie de acces este necesară pentru autorizarea apelurilor API realizate de componenta Serviciul API. Pe lângă acest formular mai există o rubrică de resetare a parolei, pentru cei care și-au uitat parola contului. La acea categorie există un mic formular unde, utilizatorul în nevoie poate introduce o adresă de e-mail. Adresa este preluată de componenta Serviciul API și pregătită cu cheia de acces, urmând să fie făcut un apel API către server.

Înregistrarea se face prin intermediul paginii de signup unde utilizatorul poate completa un formular ce conține un câmp pentru e-mail, și două pentru parolă, respectiv confirmarea parolei. După introducerea unei adrese de e-mail valide, îndeplinirea condiției parolei și confirmarea acesteia, utilizatorul este înscris în aplicație. Aplicația ia e-mailul și parola, iar prin intermediul Serviciului API le trimite cu un apel API către server pentru înscriere.

Această componentă comunică direct doar cu o altă componentă, Serviciul API, prin intermediul căreia trimite sau preia date de la server.

2.5.2 Componenta Routing

Această componentă este alcătuită cu ajutorul următoarelor tehnologii: limbajul JavaScript, librăria React și librăria React-Router-DOM.

Componenta Routing se ocupă de navigarea prin aplicație. Această componentă este crucială în definirea dinamicii și a termenului Aplicație pe o Singură Pagină. Utilizatorul interacționează direct cu această componentă, iar prin intermediul endpoint-urilor diferite sunt încărcate și afișate contextul și componenta aferentă.

Această componentă comunică cu Serviciul API pentru a asigura faptul că token-ul de acces nu este expirat. Această cheie de acces este necesară pentru a fi autorizat apelul API. Dacă cheia expiră, se realizează un apel API care regenerează cheia.

2.5.3 Pagina Dashboard

Pagina Dashboard este construită prin intermediul tehnologiilor: limbajul JavaScript, librăria React și librăria Material UI.

Componenta ce reprezintă pagina dashboard se ocupă de afișarea cardurilor utilizatorilor din apropiere. Utilizatorul interacționează direct cu această componentă prin intermediul cardurilor interactive. Aceste carduri interactive oferă detalii despre utilizatori, cum ar fi: poza lor de profil, echipa de fotbal preferată a acestora, numele lor, vârsta și distanța la care se află față de utilizatorul principal. Există două posibilități prin care utilizatorul poate da swipe: să tragă cardul stânga sau dreapta, sau să apese pe butoanele de pe carduri. Pentru a spune da cuiva, utilizatorul trebuie să dea swipe cardului spre stânga prin intermediul unui sistem de animații și recunoaștere de gesturi, sau să apese butonul verde care creează animația automat. Totodată, utilizatorul poate să dea swipe spre dreapta prin intermediul aceluiași sistem sau să apese butonul roșu, pentru a spune cuiva nu.

Pagina Dashboard se folosește de sistemul API pentru a trimite către server swipe-ul ales de un utilizator. Prin intermediul aceluiași sistem, această componentă poate să preia de la server lista de utilizatori disponibili și detaliile acestora, pentru a crea un card complet și detaliat.

2.5.4 Pagina Chat

Componenta ce reprezintă Pagina Chat este alcătuită cu ajutorul următoarelor tehnologii: limbajul JavaScript, librăria React, librăria Material UI și Web Sockets.

Pagina chat este o componentă complexă care se ocupă de conectarea canalului de comunicare la server, pentru a putea avea o comunicare fluentă în timp real. Totodată, componenta se ocupă de afișarea paginii de chat, unde utilizatorul are afișați oamenii cu care s-a împrietenit. Acolo poate alege cu cine să converseze, apăsând pe iconița lor. Odată apăsată iconița omului, se deschide chat-ul privat, împreună cu istoricul mesajelor dintre ei. Camerele de chat sunt private și pot fi accesate doar de cei doi. În această componentă este prezent și modelul de analiză de sentiment care a fost adăugat pentru a analiza conversațiile și pentru a împinge oamenii să fie mai

pozitivi și mai respectuoși. Istoricul conversației este configurat ca o listă virtualizată unde sunt randate doar mesajele prezente pe ecran.

Această componentă comunică și ea cu Serverul API pentru a lua de la server: sentimentul conversației, pentru a lua utilizatorii cu care utilizatorul principal s-a împrietenit, detaliile acestora și pentru a prelua și încărca istoricul de mesaje al unei conversații.

2.5.5 Serviciul API

Această componentă este construită prin intermediul următoarelor tehnologii: limbajul JavaScript, librăria React, librăria Axios și librăria JSON Web Tokens.

Componenta Serviciul API este de departe stâlpul aplicației web Find Football Friends, deoarece, aceasta reprezintă podul dintre Aplicația pe o Singură Pagină și Aplicația API. Acest “pod” ilustrează ideea că, cele două containere comunică între ele, iar fluxul datelor este bidirecțional. Fără de această componentă, legătură dintre server și client nu ar mai exista.

Serviciul API este responsabil cu gestionarea și pregătirea apelurilor API primite de la toate celelalte componente ale containerului Aplicație pe o Singură Pagină. Această pregătire constă în atașarea automată a unei chei de acces în header-ul apelului API pentru a avea autorizare la opțiunile serverului. După această atașare, prin intermediul librăriei Axios, apelul se trimite către Aplicația API.

2.5.6 Componenta Notificări

Această componentă este alcătuită cu ajutorul tehnologiilor: limbajul JavaScript, librăria React și librăria Material UI.

Componenta Notificări este responsabilă cu preluarea și afișarea notificărilor pe interfața grafică a utilizatorului. Această componentă se află în bara de navigare a paginii și este activată de un buton sub formă de clopoțel. Odată activată, se deschide un meniu în care sunt toate notificările primite. Utilizatorul poate vedea cine i-a scris sau cine a realizat împrietenirea cu el, după ce el a dat swipe “da” primul. Notificările pot fi șterse atât individual, cât și în totalitate.

Această componentă are legătură directă cu componenta Serviciul API, prin intermediul căreia face apeluri API pentru a obține notificările disponibile pentru utilizatorul principal. Componenta mai realizează două tipuri de apeluri API pentru a șterge una sau toate notificările.

2.5.7 Pagina Profil

Pagina Profil este o componentă construită cu ajutorul tehnologiilor următoare: limbajul JavaScript, librăria React și librăria Material UI.

Această componentă este responsabilă cu încărcarea datelor și cu afișarea paginii de profil a utilizatorului. În această parte, utilizatorul își poate vizualiza datele, preferințele în legătură cu distanța maximă în care aplicația caută utilizatori disponibili pentru acesta, echipa preferată de fotbal și poza de profil actuală. În josul paginii există două butoane importante, unul pentru modificarea profilului, care redirecționează la componenta Setări Profil și unul pentru deconectare. Acest buton deconectează și redirecționează la pagina de start. Totodată, această acțiune șterge cheia de acces și de reîmprospătare.

Componenta ce este responsabilă cu pagina de profil comunică și ea la rândul ei cu Serviciul API. Prin intermediul acestui serviciu, componenta preia de la server detalii legate de utilizatorul principal.

2.5.8 Setări Profil

Setări Profil este o componentă a containerului Aplicație pe o Singură Pagină care este alcătuită prin intermediul următoarelor tehnologii: limbajul JavaScript, librăria React și librăria Material UI.

Componenta paginii Setări Profil este responsabilă cu afișarea și configurarea datelor și preferințelor personale. Această pagină de setări a profilului constă într-un formular care încarcă datele prezente ale utilizatorului și care, ulterior, pot fi modificate. Utilizatorul își poate modifica poza de profil, își poate modifica numele, descrierea, data nașterii și sexul. Tot în acest formular, utilizatorul își poate alege echipa preferată prin intermediul unei bare de căutare cu autocompletare. Aceasta afișează în mod activ ca rezultate, numele echipelor de fotbal disponibile, al căror început se potrivește cu valoarea introdusă. În final, mai există două componente ale formularului care sunt legate între ele. Este vorba despre o bară care permite utilizatorului să aleagă o valoare între zero și o sută, valoare ce reprezintă distanța maximă în care vrea să descopere oameni noi. Cealaltă componentă a formularului este o căsuță de bifat concepută pentru a alege dacă utilizatorul vrea sau nu să descopere oameni la nivel global. Dacă căsuța este bifată, atunci bara pentru distanța maximă este dezactivată, iar utilizatorul va putea vedea oamenii la nivel global. După ce debifăm căsuța, bara îți revine și poate fi din nou folosită, distanța devenind din nou locală.

La finalul formularului se află butonul de salvare a modificărilor. Acest buton trimite un apel API de tip PATCH către Serverul API. Acest apel este ulterior trimis către server, unde se actualizează baza de date cu detaliile introduse.

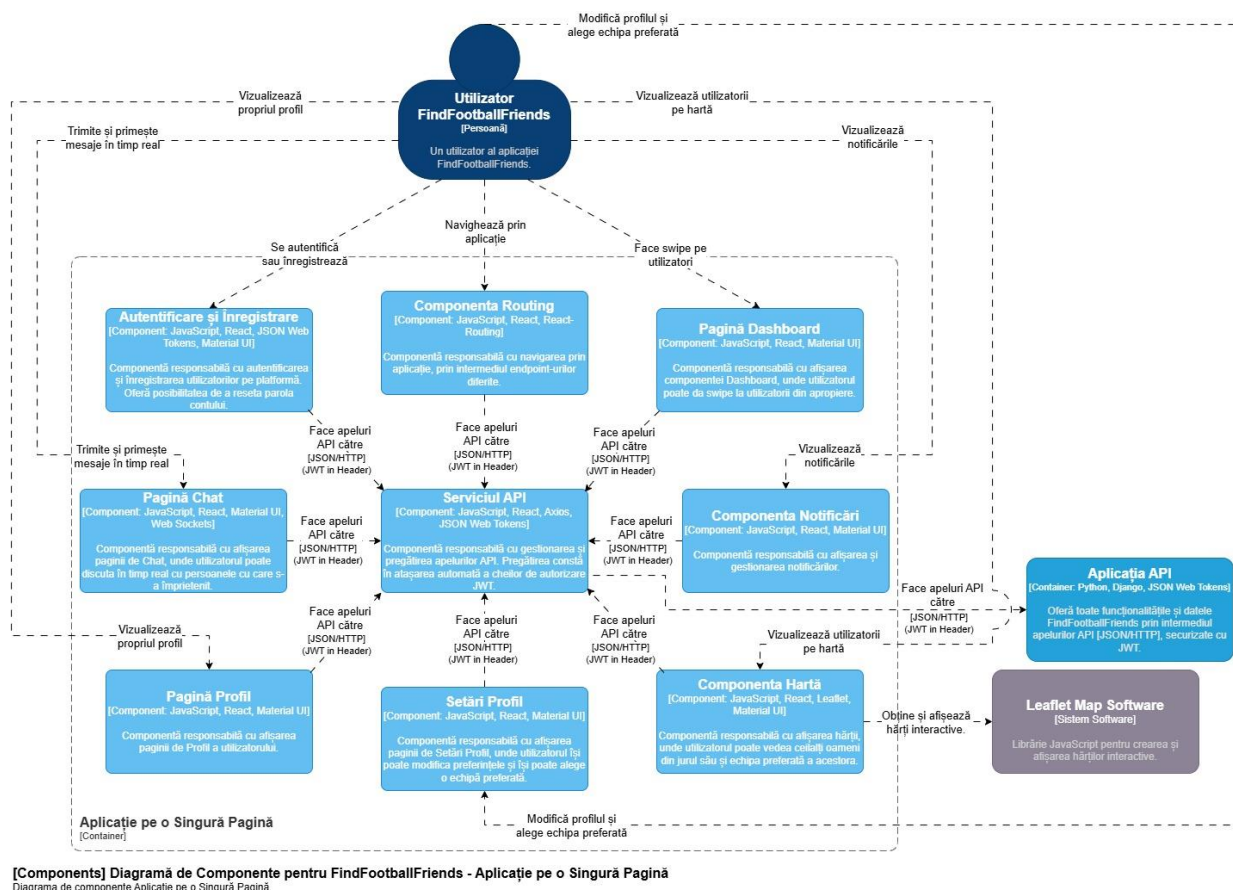
2.5.9 Componenta hartă

Componenta hartă este realizată folosind următoarele tehnologii: limbajul JavaScript, librăria React, librăria Leaflet și librăria Material UI.

Această componentă este responsabilă cu implementarea și afișarea unei hărți interactive, unde utilizatorii pot vedea oamenii din jurul lor și echipa de fotbal pe care aceștia o susțin. Inițial, această librărie JavaScript Leaflet, nu era concepută pentru framework-ul React. Totuși, prin intermediul librăriei React-Leaflet, biblioteca Leaflet se transformă în componente reutilizabile și ușor de folosit.

Componenta ce ține de harta interactivă interacționează și ea la rândul ei cu Serviciul API. Aceasta dorește să preia de la server numele, echipa de fotbal susținută și locația oamenilor din proximitate.

2.6 Diagrama arhitecturii componentelor SPA



Figură 3: Diagrama arhitecturală de nivel 3, Componenta Aplicație pe o Singură Pagină

2.7 Prezentarea componentelor Aplicației API

Containerul Aplicația API este alcătuit din câteva componente esențiale, care, folosite într-o arhitectură bine structurată, redau logica din spatele interfeței grafice. Aceste componente sunt împărțite pe roluri distincte și bine definite pentru a avea o structură organizată și curată. Fiecare componentă are un rol important în arhitectura aplicației, iar în următoarea parte, vi le voi prezenta:

2.7.1 Componenta Users

Această componentă este realizată prin intermediul următoarelor tehnologii: limbajul Python, framework-ul Django și bibliotecia JSON Web Tokens.

Componenta Users este responsabilă cu gestionarea autentificării și a înregistrării, gestionarea datelor utilizatorilor și cu logica din spatele căutării oamenilor disponibili din proximitatea utilizatorului principal. Această componentă are două modele de mapare obiect-relațională (ORM). Una este cea deja construită de framework-ul Django, ce conține câmpurile e-mail și parola. Celălalt este unic și conține preferințele utilizatorului: nume, data nașterii, echipa preferată, etc. Aceste modele sunt ca niște șabloane pe baza cărora se creează obiecte sau înregistrări în baza de date. În cazul acestei componente pentru a separa credențialele de preferințe, am decis să le împart în două modele diferite, care sunt conectate între ele prin intermediul unei chei străine.

Pentru început vom discuta despre gestionarea înregistrării. Users primește credențialele utilizatorului de la Serviciul API. Serverul verifică datele și le validează, urmând să creeze și să salveze în baza de date un obiect ce conține credențialele utilizatorului. Parola nu este salvată într-un format normal, ce este criptată automat de framework-ul Django. Odată cu crearea acestui obiect, serverul creează și un obiect gol ce urmează să conțină preferințele și datele publice ale utilizatorului.

În cazul autentificării, componenta primește credențialele utilizatorului prin intermediul unui apel API POST de la Serviciul API. Aceste date sunt verificate și validate, iar serverul poate întoarce două tipuri de răspuns. Primul tip de răspuns este cel de 401 neautorizat. Acest răspuns reprezintă faptul că, datele introduse nu sunt corecte sau nu se potrivesc. Al doilea tip de răspuns este 200 ok. Acest răspuns reprezintă faptul că, datele introduse sunt corecte, iar pe lângă acest răspuns serverul trimite către Serviciul API, al interfeței grafice, un obiect JSON ce conține cheile de acces și de reîmprospătare.

În partea de gestionare a datelor utilizatorului, serverul poate prelua și poate scrie sau actualiza datele din baza de date.

Această componentă Users are o parte care se ocupă de căutarea celor mai apropiați utilizatori în limita maximă aleasă de acesta. Users se folosește de componenta Matches pentru a prelua lista utilizatorilor deja văzuți și swipe-uiți de acesta. Componenta preia această listă pentru a îi exclude din lista utilizatorilor disponibili și neswipe-uiți de utilizatorul principal.

2.7.2 Componenta Matches

Această componentă este construită cu ajutorul tehnologiilor: limbajul Python și framework-ul Django.

Componenta Matches este responsabilă cu gestionarea swipe-urilor și a împrietenirilor. Această componentă conține două modele de mapare obiect-relațională (ORM). Prima este cea de Swipe, model care conține numele utilizatorului care a dat swipe, numele celui care a primit swipe și tipul de swipe. Al doilea model este cel de Match, sau împrietenire care conține numele primului utilizator, numele celui de-al doilea, ora și data la care a fost realizată împrietenirea și un cod uuid. Acest cod uuid este un cod de identificare unic ce reprezintă codul camerei private de chat.

Partea de gestionare a swipe-urilor constă în preluarea și crearea swipe-urilor date de utilizatorul principal. Această parte de creare a swipe-urilor pornește atunci când un utilizator interacționează cu cardul de pe componentă Aplicației pe o Singură Pagină, Pagina Dashboard. Serverul primește prin intermediul Serviciului API, un obiect JSON ce conține în header cheia de acces pentru autorizare, iar obiectul în sine deține detalii despre utilizatorul care a dat swipe, cel care a primit swipe și tipul de swipe. Odată primit și autorizat, obiectul este serializat, salvat și ulterior adăugat în baza de date.

Partea de gestionare a împrietenirilor pornește de la un swipe de tip pozitiv. Dacă un swipe este de tip pozitiv, serverul caută în baza de date dacă utilizatorul a primit și el la rândul său, un swipe pozitiv. Dacă nu, se merge mai departe și nu se întâmplă nimic. Dacă și celălalt utilizator a oferit un swipe pozitiv, atunci se creează un obiect de tip Match și este adăugat în baza de date. Odată cu obiectul Match este conceput și obiectul Notification ce conține numele utilizatorului care a dat swipe-ul, care a dus la împrietenire, numele celui alt utilizator și tipul notificării. În cazul dat, tipul notificării este de împrietenire.

Tot în partea de gestionare a împrietenirilor mai avem partea de preluare a împrietenirilor care este un punct important pentru componenta Pagina Chat din containerul Aplicație pe o Singură Pagină.

2.7.3 Componenta Chats

Această componentă este construită și dezvoltată cu ajutorul următoarelor tehnologii: limbajul Python, framework-ul Django și librăria Django-Channels.

Componenta Chats este responsabilă cu gestionarea mesajelor, preluarea istoricului conversațiilor, analiza sentimentului conversației, crearea și configurarea conexiunii camerelor de chat. Această componentă conține și ea la rândul ei un model de mapare obiect-relațională numit Message. Modelul conține numele celui care trimite mesajul, numele celui care primește mesajul, conținutul mesajului și data și ora la care acesta este trimis. Acest model permite și ordonarea

mesajelor în funcție de dată și oră, pentru a compune un chat modern și pentru a avea o ordine cronologică a mesajelor.

Partea de gestionare a mesajelor constă în crearea și preluarea mesajelor dintre doi utilizatori. Crearea obiectului mesaj se face atunci când web socket-ul din componenta Pagina Chat a containerului SPA, trimite un mesaj către server. Mesajul este preluat ca un obiect JSON, este descompus în detalii precum: numele utilizatorului care a trimis mesajul, numele celui care a primit și conținutul mesajului. Cu aceste detalii, se creează și salvează în baza de date obiectul de tip Message. Atunci când un obiect de tip Message este creat, cu acele detalii construim și obiectul Notification care o să conțină: numele celui care a trimis mesajul, numele celui care l-a primit, și tipul va fi mesaj. După ce este creat și salvat mesajul este trimis instant către cealaltă persoană. Partea de preluare a mesajelor necesită numele celor doi utilizatori care conversează. Când facem rost de aceste două nume, putem prelua istoricul conversației dintre aceștia. Când preluăm mesajele le ordonăm ca în partea de jos să avem cel mai nou mesaj.

Analiza sentimentului se face printr-un model de procesare a limbajului natural NLP pre-antrenat pe seturi de date uriașe. Pentru a configura această parte trebuie să introduc numele modelului, să realizez canalul de comunicare și să selectez ultimele zece mesaje din conversația a doi utilizatori. După, îi trimit prin canalul de comunicare aceste mesaje, serverul returnează sentimentul printr-un obiect JSON. Extrag sentimentul și îl trimit mai departe către Serviciul API ca răspuns.

Partea de creare și configurare a conexiunii camerelor de chat constă în: configurarea Interfeței de Poartă a Sistemului Asincron (ASGI) și a protocolului Daphne folosit pentru a porni conexiunea serverului. Totodată, această parte constă în construirea camerelor private și în pregătirea endpoint-ului prin intermediul căreia interfața grafică se conectează.

Această componentă comunică direct cu Users pentru a realiza, prin intermediul acesteia, validarea identității utilizatorilor ce participă la conversație.

2.7.4 Componenta Notifications

Această componentă este dezvoltată prin intermediul următoarelor tehnologii: limbajul Python și framework-ul Django.

Componenta Notifications este responsabilă cu gestionarea notificărilor utilizatorului. Această componentă conține un model de mapare obiect-relațională numit Notification care are ca și câmpuri: numele celui care realizează acțiunea, numele celui care primește acțiunea și tipul

acțiunii. În prezent, există doar două tipuri de acțiuni care produc notificări în aplicația web Find Football Friends. Este vorba despre împrieteniri și mesaje. Aceste notificări după ce sunt create sunt salvate în baza de date.

Partea de gestionare a notificărilor constă în crearea, preluarea și ștergerea, atât a unei notificări, cât și a întregului grup de notificări al unui utilizator. Crearea notificărilor se realizează exact după ce un utilizator produce swipe-ul pozitiv ce creează împrietenirea. Celălalt mod de creare a notificării se realizează exact după ce un utilizator trimite un mesaj altui utilizator. Preluarea grupului de notificări necesită numele utilizatorului și returnează notificările care au ca și corespondent, numele acestuia. Serviciul API trimite un apel către server, unde acesta verifică și autorizează cheia de acces. Dacă răspunsul este pozitiv, serverul returnează grupul de notificări. Partea de ștergere oferă utilizatorului modalitatea de a șterge doar o notificare individual, sau modalitatea de a șterge întreaga poșta de notificări. Aceste două tipuri de ștergere se realizează prin intermediul a două apeluri API diferite, realizate de către Serviciul interfeței web.

2.7.5 Componenta Teams

Această componentă este construită prin intermediul tehnologiilor: limbajul Python și framework-ul Django.

Componenta Teams este responsabilă cu gestionarea echipelor de fotbal disponibile. Teams conține un model de mapare obiect-relațională intitulat Team, ce conține un identificator unic, numele echipei de fotbal și adresa la care găsim stema acesteia. Componenta comunică cu baza de date pentru a prelua detalii legate de echipele de fotbal.

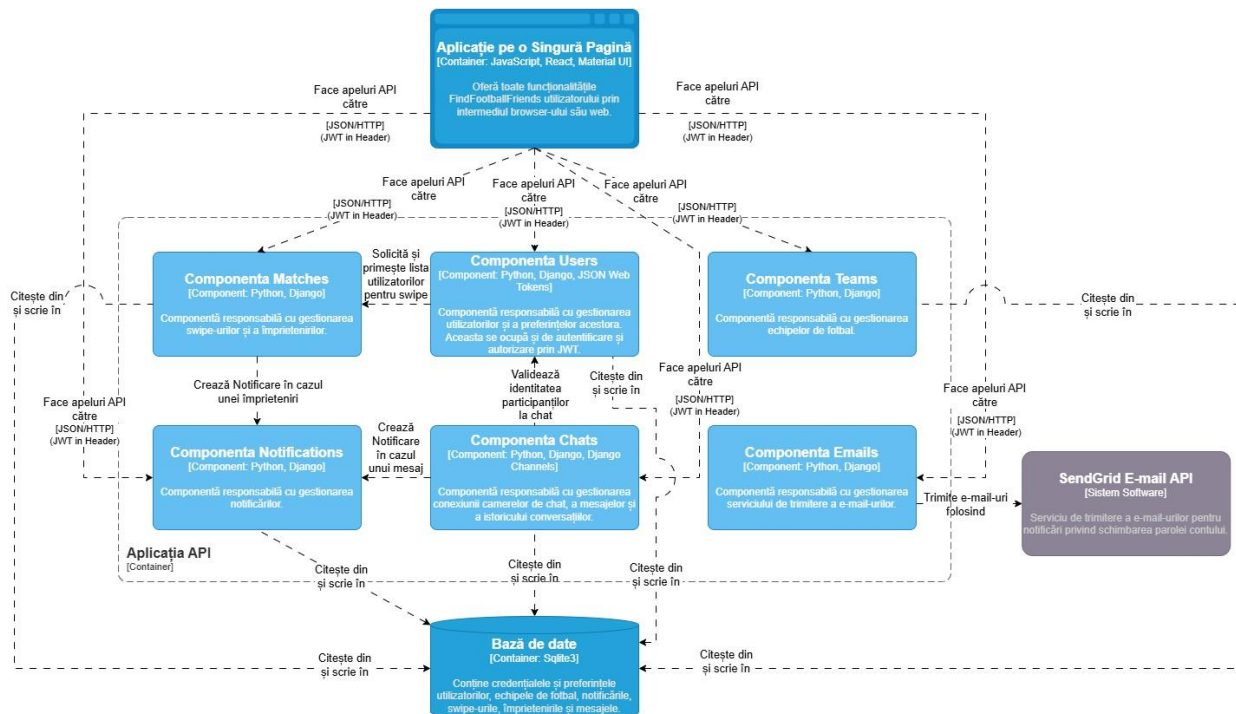
O parte importantă a acestei componente este logica din spatele barei de căutare cu autocompletare. Această bară trimite valoarea introdusă prin intermediul Serviciului API către server, acesta preia valoarea și returnează toate obiectele de tip Team în care este prezentă valoarea. Tot prin intermediul Serviciului API, interfața grafică obține informații despre echipe pentru a le afișa pe hartă sau pe cartonașul utilizatorului. Aceasta trimite un apel API HTTP GET, care odată verificat și autorizat returnează adresa la care găsim stema clubului de fotbal.

2.7.6 Componenta Emails

Această componentă este construită și dezvoltată cu ajutorul următoarelor tehnologii: limbajul Python și framework-ul Django.

Componenta Emails este responsabilă cu gestionarea serviciului de trimitere a e-mail-urilor către utilizatori. Această componentă se folosește direct de un serviciu extern numit SendGrid. Acest serviciu este propriu configurat și securizat. Când un utilizator dorește să își reseteze parola contului, acesta trimite prin intermediul Serviciului API un e-mail valid. Se verifică și se autorizează, iar în cazul pozitiv se trece la următoarea etapă. Serverul verifică e-mailul dacă este existent în baza de date. Ulterior creează un id și o cheie unică, cu care formează o adresă unică pentru resetarea parolei de forma: “http://localhost:3000/reset-password/id/cheie”. Această adresă este adăugată într-un șablon de e-mail, ce urmează să fie trimis către utilizatorul în nevoie. Prin intermediul serviciului SendGrid, e-mail-ul este trimis și ajunge către utilizator în maximum zece secunde. Utilizatorul acum poate accesa adresa către formularul de resetare. Acolo acesta își introduce noua parolă, care, după apăsarea butonului este trimisă înapoi serverului prin Serviciul API. Într-un final, serverul preia noua parolă și o salvează în dreptul e-mail-ului utilizatorului.

2.8 Diagrama arhitecturii componentelor Aplicației API



[Components] Diagramă de Componente pentru FindFootballFriends - Aplicația API
Diagrama de componente Aplicația API

Figură 4: Diagrama arhitecturală de nivel 3, Componenta Aplicație API

3. Implementarea aplicației

Aplicația web Find Football Friends este o aplicație socială a cărei tematică se învârtă în jurul fotbalului. Această aplicație permite utilizatorilor să se conecteze între ei pe baza preferințelor lor fotbalistice. Utilizatorii își pot modifica profilul și preferințele după bunul plac și își pot alege echipa preferată. În partea de descoperire a oamenilor avem cartonașe interactive care prezintă fiecare utilizator aflat în raza maximă de căutare. Totodată, Find Football Friends oferă și o hartă interactivă care prezintă locația oamenilor care au cont pe aplicație din jurul utilizatorului principal. Odată ce oamenii se împrietenesc prin sistemul intuitiv de swipe, aceștia pot comunica într-un chat privat.

Find Football Friends este implementată folosind o arhitectură modulară care împarte responsabilități diferite componentelor. Aplicația urmează un șablon de tip server - client, unde utilizatorul interacționează cu interfața grafică sau clientul, iar logica și funcționalitatea principală se află în server. Fiecare funcționalitate este folosită prin intermediul unui endpoint API bine definit. Aceste endpoint-uri sunt cruciale în comunicarea dintre client și server, deoarece prin intermediul acestora clientul poate prelua sau trimite date către server. Totodată, clientul poate apela diferite funcții prin intermediul acestor endpoint-uri delimitate logic.

În acest capitol voi prezenta fiecare endpoint al fiecărei componente și ce funcționalitate are acesta. Fiecare endpoint ce apelează funcțiile din server necesită autorizare prin intermediul unei chei de acces. Această cheie este trimisă de către Serviciul API în header-ul apelului.

3.1 Componenta Users

- GET “/api/user/<str:username>”

Această funcție preia din baza de date datele utilizatorului, în funcție de username-ul trimis de către Serviciul API. Serverul preia numele și îl caută în baza de date. Dacă există, acesta extrage obiectul utilizatorului îl transformă în obiect JSON serializându-l și îl trimite înapoi ca răspuns către Serviciul API.

- PATCH “/api/user/<str:username>/settings/”

Această funcție este responsabilă cu actualizarea preferințelor utilizatorului. Prin intermediul unui username trimis de către Serviciul API, serverul caută profilul utilizatorului. Dacă există acest profil, serverul preia de la Serviciul API detaliile actualizate și le salvează doar pe cele care au suferit vreo modificare.

- GET “/api/get-users-for/<str:username>/”

Funcția se ocupă cu obținerea oamenilor din proximitatea utilizatorului. Serverul primește de la Serviciul API username-ul, iar cu acesta se obține obiectul cu profilul utilizatorului. Dacă acest profil există, se obține din baza de date și o listă de utilizatori swipe-uiți de utilizatorul principal. Preluăm această listă pentru a nu pune înapoi oamenii ce au fost deja swipe-uiți în lista utilizatorilor disponibili.

```
start_point = (main_user.latitude, main_user.longitude)

...

close_users = []

for user in users:

    if user.user.username != username and user.user.username not in swiped_usernames:

        if user.latitude is not None and user.longitude is not None:

            end_point = (user.latitude, user.longitude)

            distance = haversine(start_point, end_point, unit=Unit.KILOMETERS)

            if distance < main_user.max_distance:

                ...

                close_users.append(user)
```

Această bucată de cod preia dintr-o listă cu toți utilizatorii, câte unul, verifică dacă este diferit de utilizatorul principal și dacă nu a fost deja swipe-uit de acesta. Apoi calculează distanța dintre cei doi prin intermediul librăriei Haversine ce calculează distanțe dintre coordonate reale, luând în considerare curbura Pământului. Această distanță este comparată cu distanța maximă preferată de utilizatorul principal. Dacă este mai mică, atunci adăugăm utilizatorul în listă și imediat va fi afișat pe interfața grafică. Această funcție returnează Serviciului API o listă cu utilizatori disponibili și eligibili.

- POST “/api/user/<str:username>/upload_picture/”

Această funcție se ocupă de actualizarea pozei de profil a utilizatorului. Serviciul API trimite numele utilizatorului către server. Cu acest nume preluăm obiectul utilizator din baza de date. Dacă obiectul există, preluăm de la apelul Serviciului API fișierul cu poza, pe care o salvăm într-un fișier media în server. În baza de date vom salva adresa pentru a obține acea poză.

- POST “/api/user/register/”

Acest endpoint creează obiectul ORM User ce conține credențialele utilizatorului înregistrat. Pe lângă acest obiect mai este creat și modelul gol MyUser ce va conține preferințele utilizatorului.

- POST “/api/token/”

Acest endpoint apelează o funcție deja construită, datorită librăriei JWT, în framework-ul Django. Această funcție primește de la serviciul API credențialele unui utilizator în urma autentificării. Serverul le primește și le verifică dacă sunt existente și corecte. Ulterior serverul generează o cheie de acces și una de reîmprospătare JWT, le adaugă într-un obiect JSON și le trimite înapoi către client.

- POST “/api/token/refresh”

Endpoint-ul acesta apelează o funcție deja construită, cu ajutorul librăriei JWT, în framework-ul Django. Această funcție primește de la Serviciul API care a făcut apelul, o cheie de reîmprospătare. Serverul verifică dacă este validă și neexpirată. Dacă este totul în regulă, întoarce un obiect JSON către client ce conține o cheie de acces nou generată.

3.2 Componenta Matches

- GET “/api/swipe/<str:username>/”

Acest endpoint apelează o funcție ce este responsabilă cu preluarea swipe-urilor făcute de utilizator. Serverul primește de la Serviciul API un username, cu care acesta caută utilizatorul în baza de date. Dacă există, preia toate obiectele de tip Swipe care au ca expeditor acel username. Într-un final, serverul returnează către client o listă cu toate Swipe-urile realizate de utilizator.

- POST “/api/swipe-create/”

Endpoint-ul acesta apelează o funcție care este responsabilă cu crearea Swipe-urilor. Această funcție primește de la client un obiect JSON, ce este serializat. Serverul verifică dacă tipul swipe-ului este cel pozitiv. Dacă da, atunci acesta caută în baza de date dacă utilizatorul care a primit swipe-ul pozitiv, a dat deja la rândul lui, un swipe pozitiv. Dacă da, atunci funcția creează un obiect de tip Match în care pune cei doi utilizatori și generează un identificator unic care va deveni codul camerei lor de chat. Odată cu crearea obiectului Match se construiește și obiectul Notification pentru a putea informa utilizatorul care a primit ultimul swipe pozitiv că aceștia s-au împrietenit. Funcția nu trimite date către client. Ea trimite doar coduri de status în funcție de ce se întâmplă în funcție, pentru a înștiința clientul de eroare sau succes.

- GET “/api/get-matches/<str:username>/”

Acest endpoint apelează o funcție care primește de la Serviciul API al clientului, un username. După ce primește acest username, caută utilizatorul în baza de date. Dacă există, atunci serverul va căuta în baza de date toate obiectele de tip Match din care utilizatorul în cauză face parte. Această funcție va returna către client o listă de oameni care sunt împrietenți cu utilizatorul principal.

- DELETE “/api/delete-match/<str:user1>/<str:user2>/”

Endpoint-ul apelează o funcție care primește de la client, prin intermediul Serviciul API, două nume de utilizator. Serverul caută să vadă dacă cei doi sunt împrietenți. Dacă da, atunci serverul șterge împrietenirea și caută și șterge și mesajele dintre ei. De asemenea, Aplicația API caută și notificările dintre cei doi, pe care le va șterge ulterior. Această funcție va returna către client un cod de status pentru a îl înștiința dacă operațiunea a avut succes sau nu.

3.3 Componenta Chats

- GET “/msg/get-messages/<str:sender_username>/<str: receiver_username>/”

Endpoint-ul apelează o funcție ce se ocupa cu preluarea istoricului conversației dintre doi utilizatori. Această funcție se mai ocupă și cu configurarea modelului NLP, folosirea acestuia pe ultimele zece mesaje și gestionarea răspunsului.

Configurarea modelului, utilizarea acestuia și gestionarea răspunsului:

```
model_name = "clapAI/modernBERT-base-multilingual-sentiment"
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name)
```



```

model = AutoModelForSequenceClassification.from_pretrained(model_name)

sentiment_pipeline = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer,
device=-1)

...

sentiment = sentiment_pipeline(message_list_for_sentiment_analysis)[0]

```

Cealaltă parte a funcției constă în preluarea istoricului conversației dintre doi utilizatori. Funcția primește de la client, prin intermediul apelului API, cele două nume. Pe baza acelor nume, serverul caută în baza de date mesajele dintre cei doi și le ordonează, astfel încât cel mai nou mesaj să fie cel mai jos. Într-un final funcția returnează către client lista cu mesajele ordonate și sentimentul rezultat din analiza ultimelor zece mesaje.

- WebSocket “/ws/chat/<str:room_name>/”

Această parte constă în configurarea conexiunii camerelor de chat în timp real. Aceasta constă în configurarea Interfeței de Poartă a Sistemului Asincron (ASGI) și a protocolului Daphne utilizat pentru a porni conexiunea serverului.

```

application = ProtocolTypeRouter({

    "http": get_asgi_application(),

    "websocket": AuthMiddlewareStack(

        URLRouter(websocket_urlpatterns)

    ),

})

```

Această configurare spune că dacă cererea este de tip HTTP, Django să se ocupe normal de ea. Totodată, dacă cererea este de tip WebSocket, atunci verifică dacă utilizatorul este autentificat și este ulterior redirecționat către adresele din “websocket_urlpatterns”.

```

websocket_urlpatterns = [

    path('ws/chat/<str:room_name>/', ChatConsumer.as_asgi()),

]

```

Aceasta este ruta prin intermediul căreia, serverul primește conexiunea de la client. Gestionarea acestei conexiuni este foarte importantă pentru a păstra sesiunea activă pe parcursul conversației.

```
class ChatConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        ...

        self.room_name = self.scope['url_route']['kwargs'].get('room_name', None)

        if self.room_name is None:

            await self.close()

            return

        self.room_group_name = f"chat_{self.room_name}"

        await self.channel_layer.group_add(

            self.room_group_name,

            self.channel_name

        )

        await self.accept()

        ...
```

Aceasta funcție accesează identificatorul unic al camerei trimis de către client și verifică dacă acesta există. Dacă este existentă, atunci serverul conectează utilizatorul la grup sau cameră.

```
async def disconnect(self, close_code):

    await self.channel_layer.group_discard(

        self.room_group_name,

        self.channel_name

    )
```

Funcția disconnect părăsește grupul de chat sau camera de conversație.

```
async def receive(self, text_data):
```

```

...

text_data_json = json.loads(text_data)

message = text_data_json['message']

sender_username = text_data_json['sender']

receiver_username = text_data_json['receiver']

sender = await sync_to_async(User.objects.get)(username=sender_username)

receiver = await sync_to_async(User.objects.get)(username=receiver_username)

await sync_to_async(Message.objects.create)(

    sender=sender,

    receiver=receiver

    content=message

)

await sync_to_async(Notification.objects.create)(

    sender=sender,

    receiver=receiver,

    notification_type='message'

)

await self.channel_layer.group_send(

    self.room_group_name,

    {

        'type': 'chat_message',

        'message': message,

        'sender': sender_username,

        'receiver': receiver_username

    }

```

)

...

Funcția `receive` se ocupă de receptarea și gestionarea mesajului primit. Serverul preia obiectul și îl împarte pe categorii. Ulterior, identifică utilizatorul care a trimis mesajul și pe cel care l-a primit. Funcția creează un obiect de tip `Message` care va avea ca și conținut numele celor doi utilizatori și conținutul mesajului. După ce acest obiect este creat, este construit și obiectul de tip `Notification` pentru a înștiința utilizatorul care a primit mesajul în legătură cu acțiunea. Într-un final serverul trimite mesajul mai departe către grup sau camera de chat.

```
async def chat_message(self, event):
```

```
    message = event['message']

    await self.send(text_data=json.dumps({

        'message': message,

        'sender': event['sender'],

        'receiver': event['receiver']

    })))
```

Funcția `chat_message` este responsabilă cu primirea mesajului de la grup și trimiterea acestuia către `WebSocket`.

3.4 Componenta Notifications

- GET “/not/get-notifications-for/<str:username>”

Acest endpoint apelează o funcție care returnează notificările unui utilizator. Serverul primește `username`-ul de la Serviciul API și caută utilizatorul în baza de date. Dacă acesta există atunci funcția caută în baza de date obiecte de tip `Notification` în care acesta este trecut ca și receptorul notificării. Într-un final transformă lista de notificări într-un obiect JSON și îl returnează către client pentru a putea fi afișate.

- DELETE “/not/delete-notification/<int:notification_id>”

Endpoint-ul apelează o funcție ce este responsabilă cu ștergerea unei singure notificări, identificat printr-un identificator unic. Serverul primește de la Serviciul API un apel care trimite

în parametru un identificator unic al notificării care se dorește a fi ștearsă. Acesta caută în baza de date obiectul de tip Notification după identificator. Dacă obiectul există, atunci îl putem șterge. La final, clientul va primi un cod de status pentru a anunța dacă ștergerea a fost cu succes sau nu.

- DELETE “/not/delete-all-notifications/<str:username>/”

Acest endpoint este responsabil cu apelarea unei funcții ce șterge toate notificările care au ca și receptor utilizatorul trimis ca parametru. Serverul primește de la Serviciul API acel username și caută utilizatorul în baza de date. Dacă acesta există, atunci serverul caută în baza de date toate obiectele de tip Notification care îl au pe acest utilizator ca și corespondent. După ce preia toate aceste obiecte, serverul le șterge pe toate. În final, clientului îi va fi returnat un cod de status pentru a îl înștiința dacă ștergerea a fost realizată sau nu.

3.5 Componenta Teams

- GET “/api/teams/”

Endpoint-ul apelează o funcție ce este responsabilă cu căutarea echipelor în baza de date, după numele echipei introduse ca parametru. Când clientul folosește bara de căutare cu autocompletare, Serviciul API trimite ce este scris la server prin intermediul acestui endpoint. Serverul preia numele introdus și caută obiectele de tip Team în baza de date. Dacă există obiecte, atunci funcția le serializează și le trimite înapoi către client ca și răspuns.

- GET “/api/get-team-photo/<str:team_name>/”

Acest endpoint apelează o funcție care are ca scop obținerea adresei unde se găsește logo-ul unei echipe de fotbal din baza de date. Funcția aceasta primește ca parametru numele echipei de la Serviciul API. Serverul caută în baza de date și verifică dacă există echipa. Dacă aceasta există, atunci funcția returnează către client un obiect ce conține adresa logo-ului.

3.6 Componenta Emails

- POST “/api/emails/forgot-password/”

Endpoint-ul este responsabil cu apelarea funcției care se ocupă cu construirea adresei unice pentru resetarea parolei. Funcția primește prin intermediul Serviciul API, un obiect JSON ce conține e-mail-ul la care va fi trimisă adresa formată. Odată primită, serverul caută în baza de date

dacă există un utilizator cu acest email. Dacă există, atunci îi creează un identificator unic și o cheie unică pentru a forma adresa de resetare. Ulterior, dacă e-mail-ul introdus este valid, link-ul este adăugat într-un șablon și trimis către utilizator. La final, clientul primește doar un cod de status pentru a fi înștiințat de starea apelului.

- POST “/api/emails/reset-password-confirm/”

Acest endpoint este responsabil cu apelarea unei funcții ce confirmă și salvează parola trimisă ca parametru. După ce un utilizator a accesat e-mail-ul primit și și-a introdus noua parolă, Serviciul API îl va trimite către server, alături de identificatorul unic și o cheie. Serverul va prelua aceste date și prin intermediul identificatorului unic, găsește utilizatorul. Dacă utilizatorul există, atunci parola este salvată și actualizată în baza de date.

3.7 Alte Componente

3.7.1 Animația cardului pentru swipe

Această animație este o parte fundamentală din aplicația web Find Football Friends. Animația cardului pentru swipe se realizează prin intermediul unei funcții din cadrul componentei Pagina Dashboard. Această funcție calculează limitele swipe-ului, animează cardul și recunoaște gesturile făcute de utilizator pentru a asigura un design scalabil, fluent și plăcut.

```
function SwipeCard({ user, onSwipe, zIndex }) {  
  ... //definiri de variabile  
  
  const handleSwipe = (direction) => {  
    animate(x, direction === "right" ? swipeDistance : -swipeDistance, {  
      duration: 0.3,  
      ease: "easeOut",  
      onComplete: () => onSwipe(user, direction),  
    });  
  };  
  
  ... //funcție de detectare de dimensiune a ferestrei
```

```

const swipeDistance = screenWidth * 0.2;

const bind = useGesture({

  onDrag: ({ down, movement: [mx], last }) => {

    const maxDragDistance = swipeDistance * 0.4;

    const minSwipeThreshold = swipeDistance * 0.15;

    const limitedX = Math.max(

      -maxDragDistance,

      Math.min(mx, maxDragDistance)

    );

    if (!down && last) {

      if (Math.abs(limitedX) > minSwipeThreshold) {

        const direction = limitedX > 0 ? "right" : "left";

        animate(x, direction === "right" ? swipeDistance : -swipeDistance, {

          duration: 0.3,

          ease: "easeOut",

          onComplete: () => {

            onSwipe(user, direction);

          },

        });

      } else {

        animate(x, 0, {

          duration: 0.2,

          ease: "easeOut",

        });

      }

    }

  }

});

```

```

    } else {

      x.set(limitedX);

    }

  },

});

... //afișarea și stilizarea cardului

```

Această funcție calculează limita de tragere sau swipe a cardului și minimul de tragere pentru a se considera swipe-ul valid. Dacă minimul de tragere nu este îndeplinit, printr-o animație, cardul se întoarce înapoi în poziția inițială. Funcția permite această tragere să fie făcută doar în direcțiile stânga și dreapta.

3.7.2 Conectarea la chat și trimiterea mesajelor

Această parte este parte din componenta Pagina Chat și este fundația componentei, deoarece, fără de această parte de conectare și trimitere de mesaje, aceasta nu ar mai exista.

```

const ChatRoom = ({ roomName, sender, receiver }) => {

... // definirea variabilelor

useEffect(() => {

  const ws = new WebSocket(`ws://127.0.0.1:8000/ws/chat/${roomName}`);

  setSocket(ws);

  ws.onmessage = (e) => {

    const data = JSON.parse(e.data);

    const message = {

      sender: data.sender,

      receiver: data.receiver,

      content: data.content || data.message,

      timestamp: new Date().toLocaleTimeString(),

```



```

    };

    setChat((prev) => [...prev, message]);

    };

    return () => {

        ws.close();

    };

    }, [roomName, sender, receiver]);

const sendMessage = () => {

    if (socket) {

        if (message === "") {

        } else {

            socket.send(

                JSON.stringify({

                    sender: sender,

                    receiver: receiver,

                    message: message,

                })

            );

        }

        setMessage("");

    }

};

...

```

Această funcție se conectează la camera de chat configurată în server prin intermediul unui WebSocket. Acest WebSocket se conectează printr-o adresă URL specială care are ca și endpoint codul unic al camerei. Adresa URL specială este: “ws://127.0.0.1:8000/ws/chat/\${roomName}/”.

Funcția setează socket-ul și gestionează și salvează mesajul primit. Cu ajutorul liniei de cod `setChat((prev) => [...prev, message])`, funcția salvează mesajele primite pe baza unei stări anterioare, ceea ce permite mesajelor asincrone să fie stocate corect. Într-un final, funcția mai are și responsabilitatea de a prelua mesajul ce dorește să fie trimis, îl construiește ca un obiect JSON ce conține numele emițătorului, numele receptorului și contextul mesajului. Acest mesaj este trimis către server, unde urmează să fie gestionat și trimis către celălalt participant la discuție.

4. Manual de Utilizare

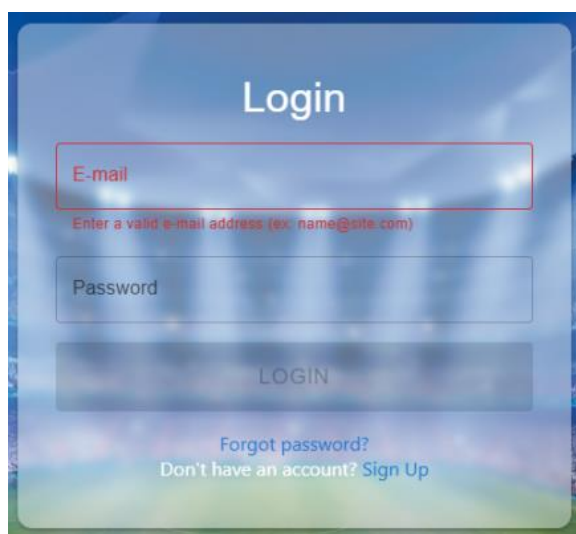
Bine ai venit la Manualul de Utilizare al aplicației web Find Football Friends, aplicația gratis și disponibilă pe orice browser și dispozitiv, care aduce suporterii mai aproape.

4.1 Autentificare și Înregistrare



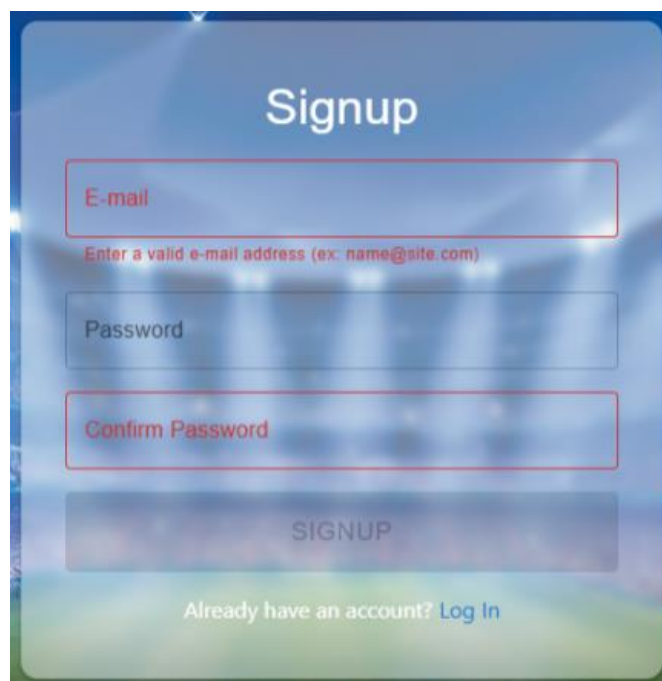
Figură 5: Pagina de start

Când intrați pentru prima dată pe această aplicație web, veți fi întâmpinați de pagina de start. Aici puteți alege dacă doriți să vă autentificați cu un cont deja existent sau să vă înregistrați.



Figură 6: Pagina de autentificare

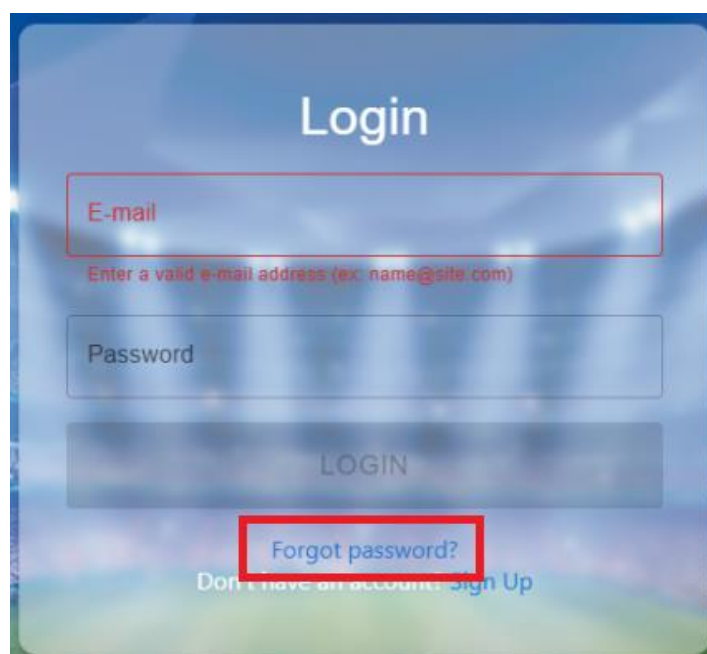
Aceasta este pagina de autentificare, unde trebuie să introduceți o adresă de e-mail validă a unui cont existent.

A screenshot of a web application's 'Signup' page. The page has a blue header with the word 'Signup' in white. Below the header, there are three input fields: 'E-mail', 'Password', and 'Confirm Password'. The 'E-mail' field has a red border and a red error message below it: 'Enter a valid e-mail address (ex: name@site.com)'. The 'Password' and 'Confirm Password' fields have blue borders. Below the input fields is a large blue button with the text 'SIGNUP'. At the bottom of the page, there is a link that says 'Already have an account? Log In'.

Figură 7: Pagina de înregistrare

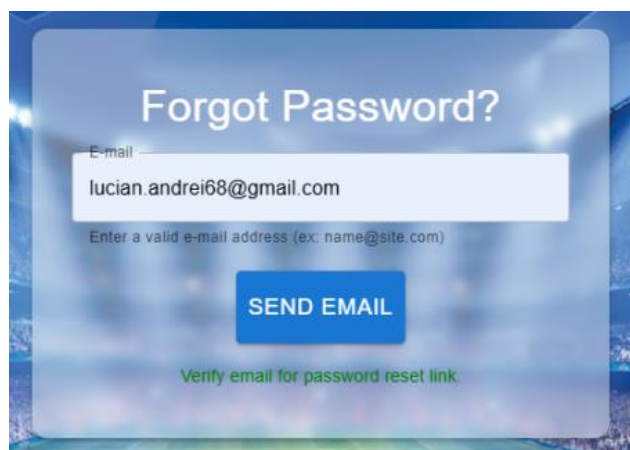
Această pagină este cea de înregistrare. Aici vă puteți crea un nou cont. Trebuie să introduceți o adresă de e-mail validă și să confirmați parola.

4.2 Resetarea Parolei

A screenshot of a web application's 'Login' page. The page has a blue header with the word 'Login' in white. Below the header, there are two input fields: 'E-mail' and 'Password'. The 'E-mail' field has a red border and a red error message below it: 'Enter a valid e-mail address (ex: name@site.com)'. The 'Password' field has a blue border. Below the input fields is a large blue button with the text 'LOGIN'. At the bottom of the page, there is a link that says 'Forgot password?' which is highlighted with a red rectangular box. Below this link is another link that says 'Don't have an account? Sign Up'.

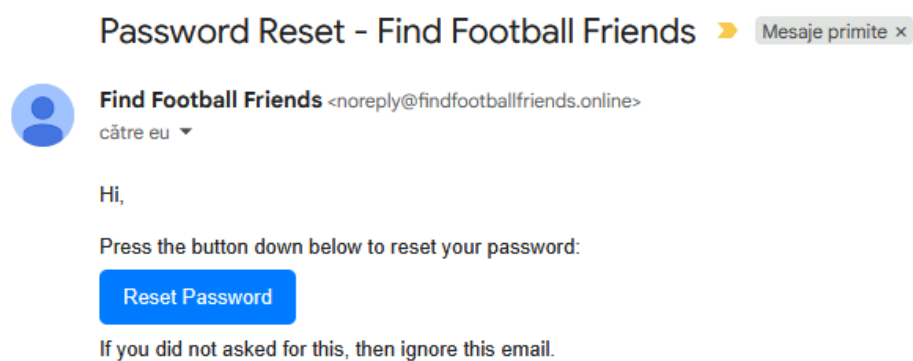
Figură 8: Redirecționare către formularul de resetare a parolei

Pagina de resetare a parolei se găsește în josul paginii de autentificare. Dacă apăsați pe acel text, veți fi redirecționat către formularul de resetare.



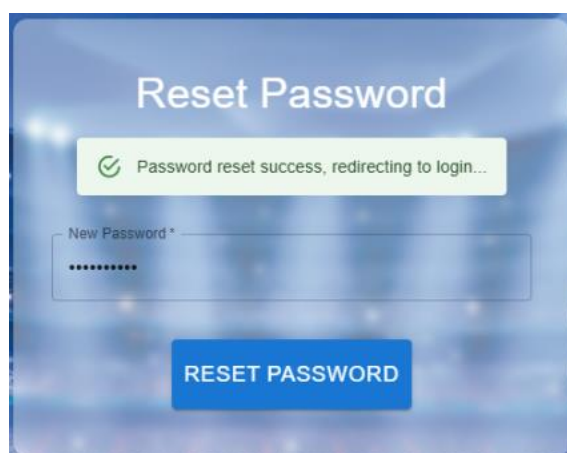
Figură 9: Pagina de resetare a parolei

Aceasta este pagina pentru resetarea parolei. Aici trebuie să introduceți un e-mail valid pentru a primi e-mail-ul pentru resetare.



Figură 10: Exemplu de e-mail primit

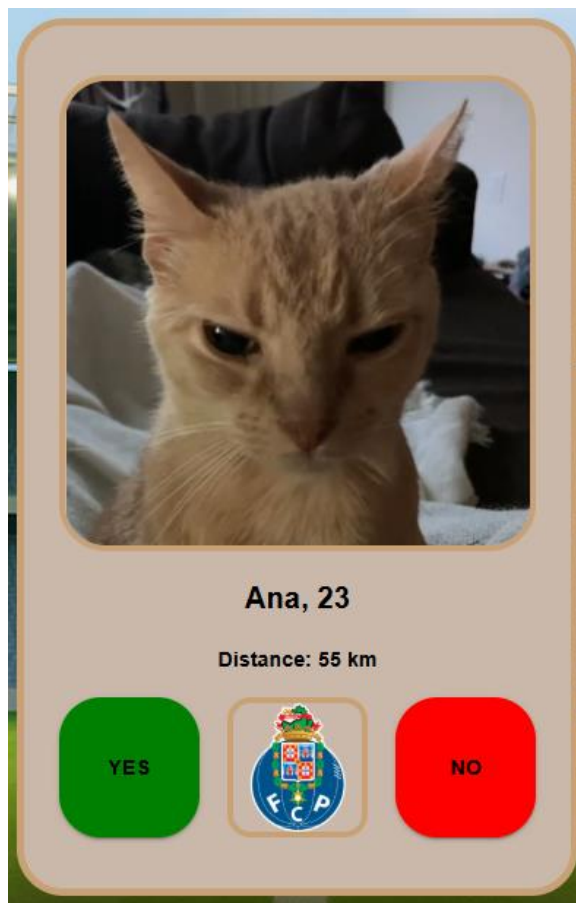
Acesta este un exemplu de e-mail primit.



Figură 11: Formularul noii parole

Dacă apăsați acel buton veți fi redirecționați aici, unde puteți introduce noua parolă.

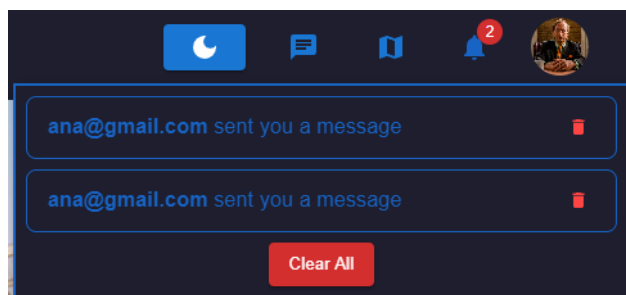
4.3 Pagina Dashboard



Figură 12: Cardul de utilizator

Acesta este cardul de utilizator de pe pagina de dashboard. Cardul conține atât detalii despre utilizator, cât și distanța față de acesta. Cardul poate fi tras spre stânga sau spre dreapta, reprezentând alegerea dumneavoastră. Totodată, puteți apăsa pe butoanele de față, care fac automat swipe-ul pentru utilizator.

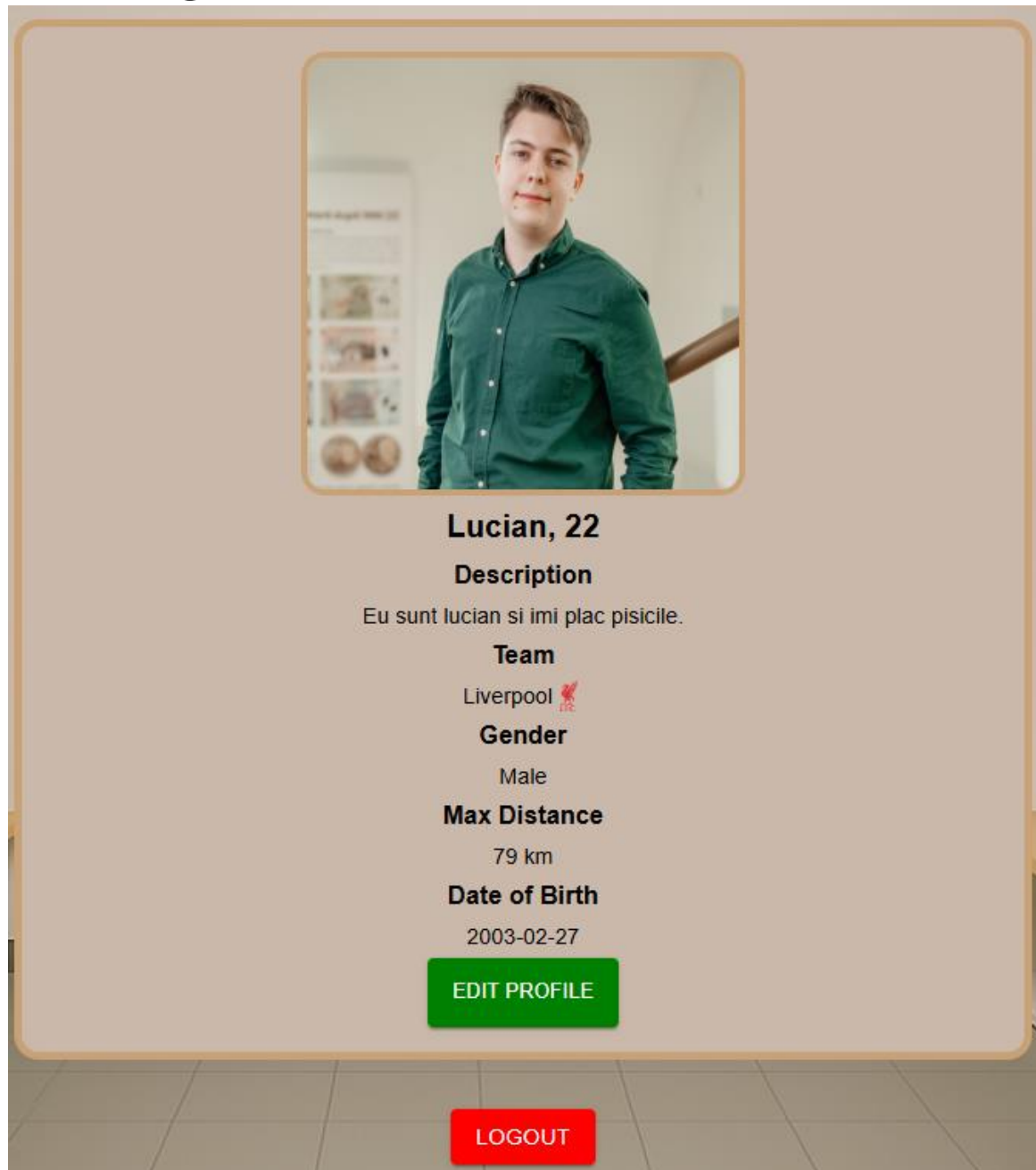
4.4 Notificări



Figură 13: Meniu notificări

Acesta este meniul cu notificări. Aici puteți vedea cine v-a trimis un mesaj sau cine s-a împrietenit cu dumneavoastră. Odată văzute, notificările pot fi șterse atât individual, cât și în totalitate.

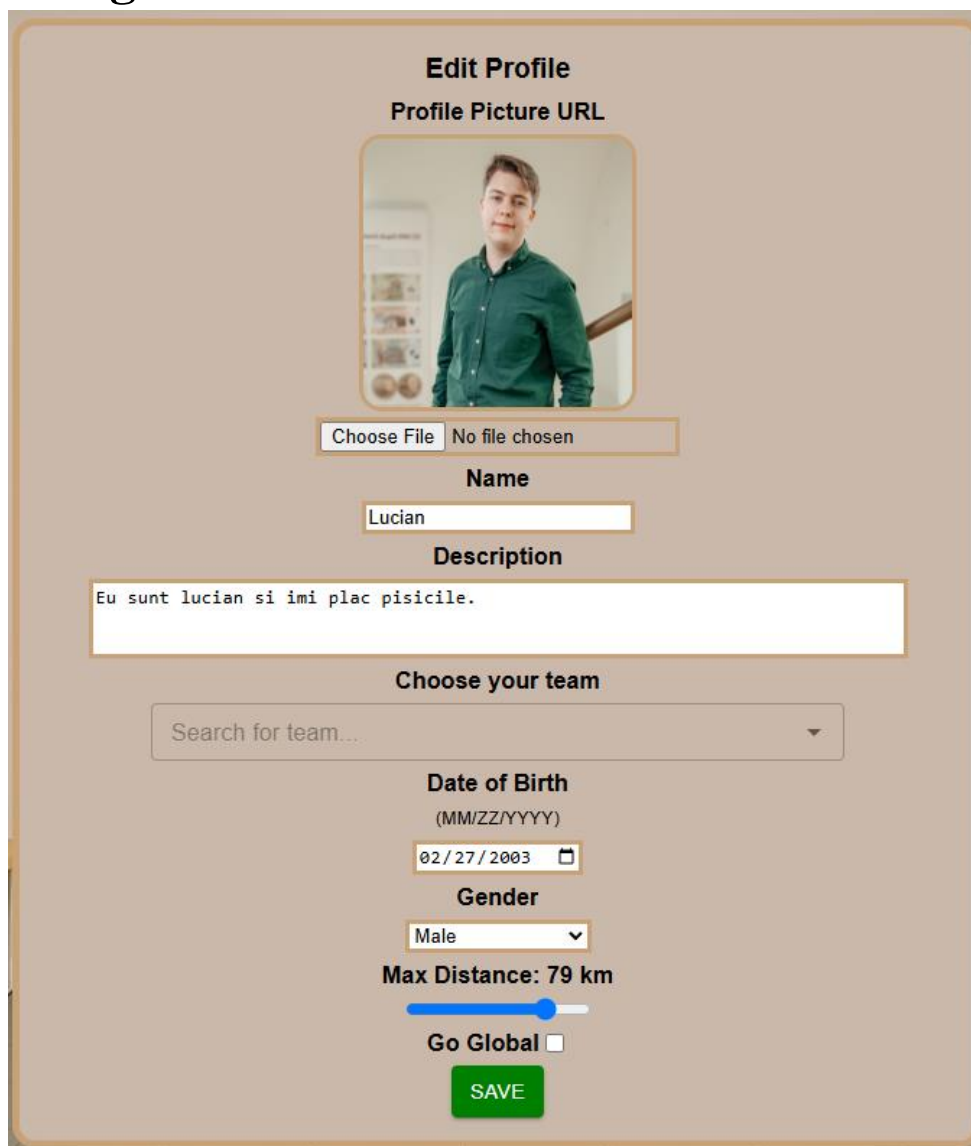
4.5 Pagina Profil



Figură 14: Pagina Profil

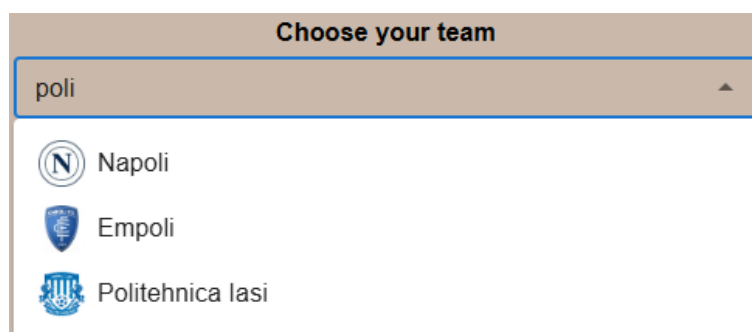
Aceasta este pagina de profil. Acestea sunt detalii publice care vor putea fi văzute de ceilalți utilizatori. Le poți customiza apăsând pe butonul: “EDIT PROFILE”.

4.6 Pagina Setări Profil



Figură 15: Pagina de Setări Profil

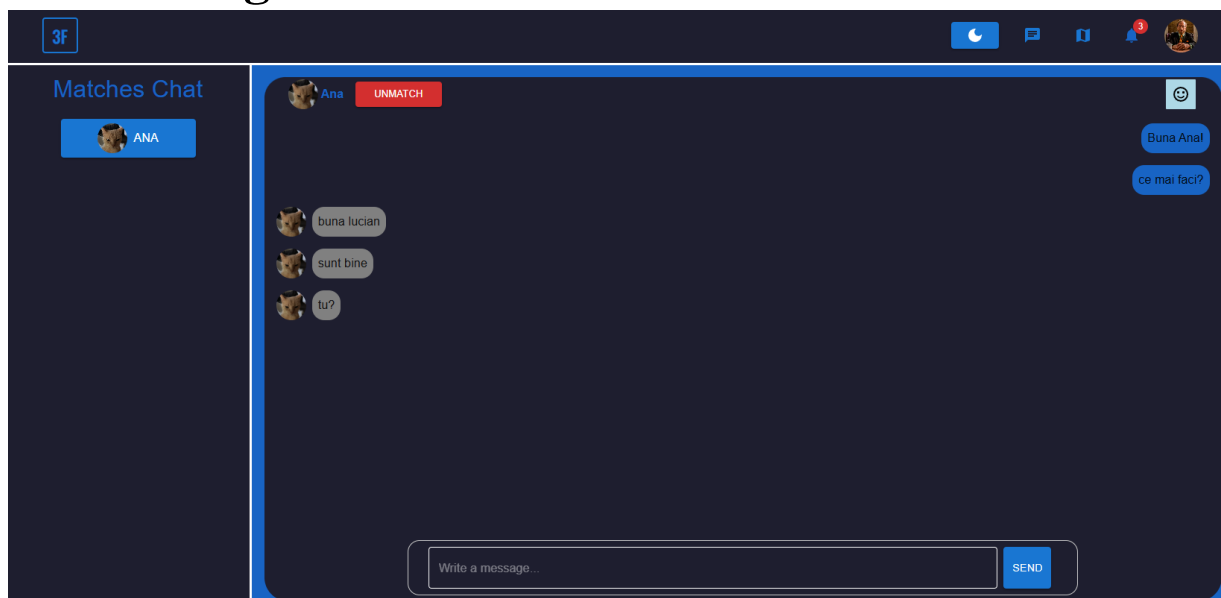
Aceasta este pagina de setări profil. Aici vă puteți modifica preferințele și detaliile publice.



Figură 16: Bara de căutare cu autocompletare

Aici vă puteți alege echipa preferată. Introduceți un nume și selectați o echipă.

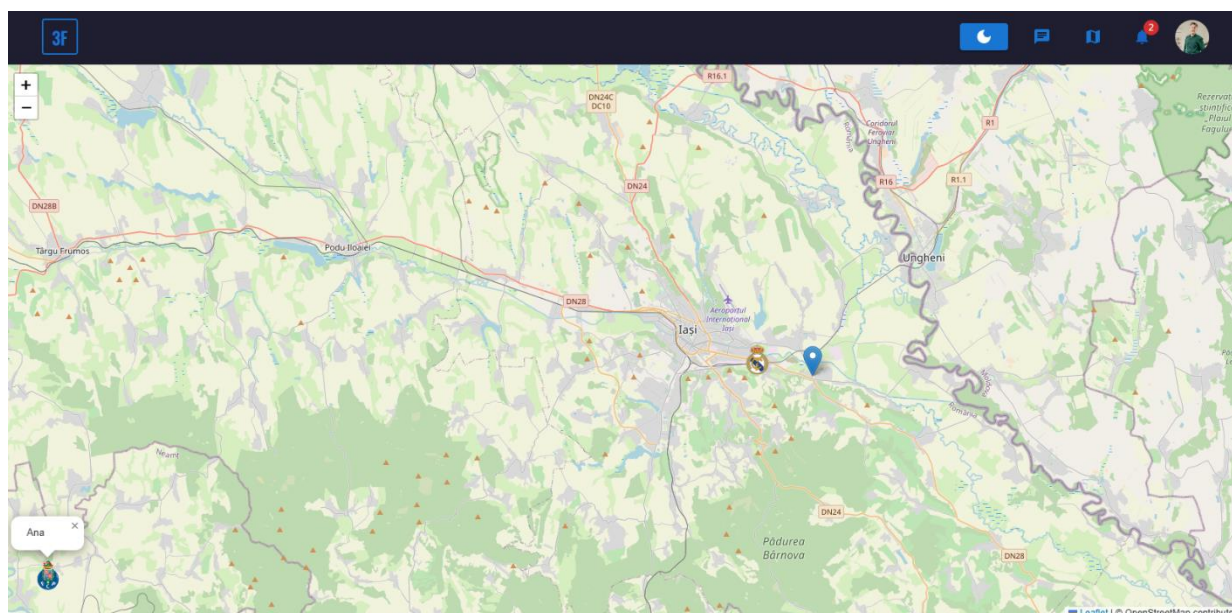
4.7 Pagina Chat



Figură 17: Pagina de Chat

Aceasta este pagina de chat. Aici puteți alege din lista din stânga cu cine să discutați. În partea din dreapta, sus, este un emoticon care reprezintă analiza sentimentului. Acesta analizează conversația pentru a atrage atenția în cazul unei conversații negative.

4.8 Harta Interactivă



Figură 18: Harta Interactivă

Aceasta este harta interactivă. Aici puteți vizualiza utilizatorii și echipa lor preferată din jurul dumneavoastră în funcție de distanța maximă, pe care ați ales-o la setările profilului.

5. Concluzii

Aplicația Find Football Friends combină cu succes tehnologia, sub forma unei aplicații sociale, cu pasiunea oamenilor pentru sport și pentru fotbal.

Prin intermediul unei interfețe prietenoase și intuitive bazate pe swiping, Find Football Friends oferă o soluție unică și relevantă pentru suporterii fotbalului. Această aplicație oferă utilizatorilor posibilitatea de a alege cu cine să interacționeze, de a discuta cu aceștia, să își vizualizeze și customizeze profilul personal și să își aleagă echipa preferată de fotbal.

Fie că utilizatorul dorește doar să converseze online, că dorește să creeze noi prietenii, că dorește să găsească un amic cu care să vadă meciul de fotbal, ori chiar toate acestea la un loc, Find Football Friends oferă un mediu plăcut și prietenos în care aceste dorințe pot fi rezolvate într-un mod simplu și eficient.

6. Bibliografie

- [1] KickChat, „KickChat,” -. [Interactiv]. Available: <https://www.kickchatapp.com/>.
- [2] TheFans, „TheFans,” -. [Interactiv]. Available: <https://thefans.io/>.
- [3] Meta Platforms, „React,” -. [Interactiv]. Available: <https://react.dev/>.
- [4] B. Eich, „JavaScript,” -. [Interactiv]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [5] Material UI SAS, „Material UI,” -. [Interactiv]. Available: <https://mui.com/material-ui/>.
- [6] Motion Division Ltd, „Framer Motion,” -. [Interactiv]. Available: <https://motion.dev/>.
- [7] Poimandres, „@use-gesture/react,” -. [Interactiv]. Available: <https://use-gesture.netlify.app/>.
- [8] Meta Platforms, „WebSockets,” -. [Interactiv]. Available: <https://ably.com/blog/websockets-react-tutorial>.
- [9] Paul Le Cam, „React-Leaflet,” -. [Interactiv]. Available: <https://react-leaflet.js.org/>.
- [10] OpenStreetMap contributors, „Leaflet,” -. [Interactiv]. Available: <https://leafletjs.com/>.
- [11] Django Software Foundation, „Django,” -. [Interactiv]. Available: <https://www.djangoproject.com>.
- [12] G. v. Rossum, „Python,” -. [Interactiv]. Available: <https://www.python.org/>.
- [13] Encode OSS Ltd, „Django REST Framework,” -. [Interactiv]. Available: www.django-rest-framework.org.
- [14] Django Software Foundation, „Django Channels,” -. [Interactiv]. Available: <https://channels.readthedocs.io/en/latest/>.
- [15] Okta, „JSON Web Tokens,” -. [Interactiv]. Available: <https://jwt.io/>.
- [16] Mapado, „Haversine,” -. [Interactiv]. Available: <https://pypi.org/project/haversine/>.
- [17] ClapAI, „Model NLP,” -. [Interactiv]. Available: <https://huggingface.co/clapAI/modernBERT-base-multilingual-sentiment>.
- [18] Twilio, „SendGrid,” -. [Interactiv]. Available: <https://sendgrid.com/en-us>.
- [19] Shopify, „React-Router-DOM,” -. [Interactiv]. Available: <https://reactrouter.com>.
- [20] diagrams.net, „diagrams.net,” -. [Interactiv]. Available: <https://app.diagrams.net/>.

Mențiuni

În procesul de dezvoltare a lucrării de licență, am folosit instrumente de inteligență artificială în următorul fel:

- ChatGPT pentru generarea de imagini folosite pe post de fundal al paginilor: Dashboard, Profil și Setări Profil, atât pentru modul întunecat, cât și pentru modul luminos.
- Microsoft Copilot, de asemenea pentru generarea de imagini folosite pe post de fundal al paginilor: Dashboard, Profil și Setări Profil.
- ChatGPT pentru identificarea greșelilor de tehnoredactare: lipsa diacriticelor sau litere lipsă.