**SPORTPLANNER**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Bogdan Zagan** |
|  |  |  |
|  | Coordonator științific: | **Dr. ING. CONF. Emil Stefan Chifu** |

**2024**

Cuprins

[Capitolul 1. Introducere 1](#_Toc171523270)

[1.1 Contextul proiectului 1](#_Toc171523271)

[1.2 Poziția produsului 1](#_Toc171523272)

[1.3 Pentru cine este această aplicație 2](#_Toc171523273)

[1.4 Metoda propusă 2](#_Toc171523274)

[1.5 Structura lucrării 2](#_Toc171523275)

[Capitolul 2. Obiectivele proiectului 4](#_Toc171523276)

[2.1. Obiective principale 4](#_Toc171523277)

[2.2. Obiective secundare 4](#_Toc171523278)

[2.3 Cerințe funcționale si nefuncționale 5](#_Toc171523279)

[2.3.1 Cerințe funcționale 5](#_Toc171523280)

[2.3.2 Cerințe nefuncționale 5](#_Toc171523281)

[Capitolul 3. Studiu bibliografic 7](#_Toc171523282)

[3.1 Aplicația Calendis 7](#_Toc171523283)

[3.1.1. Informații generale 7](#_Toc171523284)

[3.1.2. Mod de funcționare 7](#_Toc171523285)

[3.1.3. Aspecte specifice 8](#_Toc171523286)

[3.2 Aplicația FootyAddicts 8](#_Toc171523287)

[3.2.1 Informații generale 8](#_Toc171523288)

[3.2.2 Mod de funcționare 8](#_Toc171523289)

[3.2.3 Aspecte specifice 9](#_Toc171523290)

[3.3 Aplicația Fubles 9](#_Toc171523291)

[3.3.1 Informații generale 9](#_Toc171523292)

[3.3.2 Mod de funcționare 10](#_Toc171523293)

[3.3.3 Aspecte specifice 10](#_Toc171523300)

[4 Analiză și fundamentare Teoretică 11](#_Toc171523301)

[4.1. Tehnologii utilizate 11](#_Toc171523302)

[4.2. Mediu de dezvoltare 12](#_Toc171523303)

[4.3 Protocoale utilizate 13](#_Toc171523304)

[4.3.1 Protocolul de encriptare al parolelor 13](#_Toc171523305)

[4.3.2 Protocolul Hypertext Transfer (HTTP) 14](#_Toc171523306)

[4.3.3 Protocolul Simple Mail Transfer (SMTP) 16](#_Toc171523307)

[4.3.4 Protocolul WebSockets 17](#_Toc171523308)

[4.3.5 JSON Web Token (JWT) 20](#_Toc171523309)

[4.4 Arhitectura microservicii 22](#_Toc171523310)

[4.5 Design Pattern Model-View-Controller (MVC) 25](#_Toc171523311)

[5 Proiectare de detaliu și implementare 27](#_Toc171523312)

[5.1 Arhitectura aplicației 28](#_Toc171523313)

[5.2 Nivel baze de date 29](#_Toc171523314)

[5.2.1 Baza de date SportPlanner 30](#_Toc171523315)

[5.2.2 Baza de date SportPlanner\_Auth 32](#_Toc171523316)

[5.3 Nivel de server 32](#_Toc171523317)

[5.3.1 SportPlanner Service 33](#_Toc171523318)

[5.3.2 Auth Service 41](#_Toc171523319)

[5.2.3 Notification Service 45](#_Toc171523324)

[5.3 Nivel de client 47](#_Toc171523325)

[6 Testare și validare 49](#_Toc171523326)

[7 Manual de instalare si utilizare 51](#_Toc171523327)

[7.1 Resurse hardware 51](#_Toc171523328)

[7.2 Resurse software 51](#_Toc171523329)

[7.3 Instalarea și configurarea mediului de dezvoltare 51](#_Toc171523330)

[7.4 Manual de utilizare 52](#_Toc171523331)

[8 Concluzii 57](#_Toc171523332)

[8.1 Contribuțiile personale 57](#_Toc171523336)

[8.2 Analiza critică 57](#_Toc171523337)

[8.3 Posibile îmbunătățiri 58](#_Toc171523338)

[Bibliografie 59](#_Toc171523339)

[Anexa 1 61](#_Toc171523340)

[Anexa 2 62](#_Toc171523341)

[Anexa 3 63](#_Toc171523342)

# Introducere

În acest capitol, se prezintă contextul proiectului, domeniul din care face parte acest proiect, poziția produsului. Identificăm publicul țintă căruia i se adresează produsul și oferim o scurtă prezentare a metodologiei alese pentru a aborda problema.

## Contextul proiectului

Scopul acestui proiect este de a usura procesul de organizare a partidelor de fotbal, de a gasi jucatori care sa participle la meciuri, dar si de a-ti gasi mai usor partide de fotbal la care sa participi.

În era digitală, conectarea cu alți jucători și organizarea meciurilor poate fi o provocare. Aplicațiile existente se concentrează adesea pe aspecte precum scoruri și știri, neglijând nevoia jucătorilor de a găsi coechipieri și adversari. Astfel, se naște necesitatea unei platforme dedicate, care să faciliteze interacțiunea între jucători și să simplifice procesul de organizare a partidelor de fotbal. Un astfel de instrument ar aduce un plus de valoare comunității fotbalistice, încurajând participarea active, dar si un stil de viata mai sanatos prin sustinerea miscarii si a unui stil de viata sanatos.

Dezvoltarea unei platforme dedicate pentru organizarea partidelor de fotbal reprezintă un pas important în adaptarea acestui sport tradițional la nevoile și cerințele contemporane. Având în vedere popularitatea fotbalului la nivel global, o astfel de aplicație ar avea un impact semnificativ, facilitând accesul la joc și încurajând participarea activă a unui număr cât mai mare de jucători. Astfel, proiectul își propune să nu doar să sprijine comunitatea fotbalistică, ci și să contribuie la un stil de viață mai sănătos și mai activ pentru utilizatorii săi.

## Poziția produsului

Acest proiect este o aplicație web, făcută în Java, utilizand framework-ul SpringBoot care are posibilitatea să fie publicată ca produs final. Aplicația noastră, concepută pentru a conecta pasionații de fotbal și a facilita organizarea de meciuri, se diferențiază în mod clar pe piața existentă. De mentionat este faptul că acest produs va fi comparat doar cu altele exact pe aceeși tema și nu cu cele aflate pe idei asemănătoare.

În urma unei analize a competitorilor direcți, am identificat câteva aplicații similare, precum "TeamSnap" [1], "Heja" [2] si "TacticalPad" [3]. Acestea oferă funcționalități precum gestionarea echipelor, programarea meciurilor și comunicarea între jucători. Cu toate acestea, aplicația noastră se distinge printr-o serie de caracteristici inovatoare și o abordare centrată pe utilizator.

Prin combinarea acestor elemente, aplicația noastră se poziționează ca o soluție ideala și inovatoare pentru pasionații de fotbal care doresc își doresc să participe sau să organizeze meciurile simplu și eficient.

## Pentru cine este această aplicație

Aplicația noastră se adresează tuturor pasionaților de fotbal, indiferent de vârstă sau nivel de experiență. Aplicația noastră oferă o platformă incluzivă și ușor de utilizat pentru toți cei care împărtășesc pasiunea pentru fotbal.

Înțelegem că fotbalul este mai mult decât un sport, este un stil de viață. De aceea, aplicația noastră este concepută pentru a răspunde nevoilor diverse ale comunității fotbalistice, oferind o gamă largă de funcționalități care să faciliteze conectarea, organizarea și participarea la meciuri de fotbal. Fie că ești în căutarea unui coechipier pentru un meci rapid sau dorești să te alături unei echipe competitive, aplicația noastră îți oferă instrumentele necesare pentru a-ți atinge obiectivele.

## Metoda propusă

Varianta pentru care am optat în rezolvarea cerințelor specifice este dezvoltarea unei aplicații web. Aceasta va permite utilizatorilor să se conecteze, să formeze echipe și să planifice meciuri direct din browserul lor, fără a necesita instalarea unor programe suplimentare.

După o cercetare amănunțită a pieței, am constatat că există un deficit semnificativ de soluții web dedicate organizării meciurilor de fotbal. Multe platforme necesită o implicare ridicată din partea utilizatorilor și nu oferă instrumente eficiente pentru găsirea și coordonarea jucătorilor sau a locatiilor, ceea ce poate fi frustrant și consumator de timp.

Prin urmare, am decis să dezvoltăm o aplicație web originală, care să ofere o soluție completă pentru organizarea meciurilor de fotbal. Aplicația va permite utilizatorilor să adauge meciuri sau să invite alți utilizatori la meciurile sale sau să se înscrie la alte meciuri organizate de alți utilizatori . De asemenea, utilizatorii vor fi informați prin notificări. Aceste notificări vor asigura o comunicare eficientă și promptă între utilizatori.

## Structura lucrării

Capitolul 2 prezintă situația actuală, evidențiind necesitatea acestui proiect în viața de zi cu zi, obiectivele și scopurile sale. Se analizează contextul organizării meciurilor de fotbal și nevoile nesatisfăcute ale jucătorilor amatori.

Capitolul 3 este destinat studiului bibliografic și generalităților legate de acest proiect. Aici se detaliază diferite metode și tehnologii utilizate în dezvoltarea aplicațiilor web, cu un accent special pe platformele dedicate sportului.

În cadrul capitolului 4 se explică la nivel teoretic funcționarea arhitecturii și componentelor principale ale aplicației dezvoltate în Java Spring Boot. Se abordează structura aplicației, fluxurile de date și modul de gestionare a cererilor utilizatorilor.

Capitolul 5 detaliază implementarea funcționalităților și algoritmilor folosiți în cadrul proiectului. Se explică realizarea funcțiilor principale ale aplicației

Capitolul 6 prezintă elementele utilizate pentru testarea și validarea aplicației. Se descriu metodele de testare aplicate, scenariile de testare și rezultatele obținute.

Capitolul 7 oferă un ghid pas cu pas pentru utilizarea aplicației. Se explică modul de utilizare a diferitelor funcții ale aplicației.

Capitolul 8 reprezintă concluzia proiectului și se discută impactul aplicației asupra comunității fotbalistice și potențialele îmbunătățiri pe care le poate avea aplicația.

# Obiectivele proiectului

Acest capitol prezintă o imagine de ansamblu asupra aplicației, detaliind atât obiectivele principale pe care își propune să le atingă, cât și beneficiile suplimentare obținute în procesul dezvoltării. Sunt prezentate, diverse scenarii de utilizare, precum și cerințele funcționale și non-funcționale pe care aplicația trebuie să le îndeplinească.

## Obiective principale

Obiectivul principal al acestui proiect este de a crea un sistem care să faciliteze organizarea meciurilor de fotbal. Aplicația web va permite utilizatorilor să se conecteze, să-și găsească coechipieri și adversari, și să planifice eficient partidele de fotbal. Prin crearea unui cont, fiecare utilizator va putea beneficia de funcționalitățile aplicației.

În mod tradițional, organizarea meciurilor de fotbal amatori implică o comunicare îndelungată și o coordonare manuală complexă. Cu toate acestea, cu ajutorul îmbunătățirilor constante ale tehnologiilor web și a aplicațiilor de gestionare, sistemul nostru va folosi o platformă web pentru a automatiza și simplifica acest proces.

## Obiective secundare

Din obiectivele principale descrise în capitolul anterior, derivă următoarele obiective secundare care vor aduce beneficii semnificative proiectului:

* Dezvoltarea aplicației respectând un model arhitectural modern și scalabil, care să asigure ușurința în întreținerea și extinderea funcționalităților pe termen lung.
* Studierea aprofundată a framework-ului Java Spring Boot pentru a utiliza cele mai bune practici în dezvoltarea aplicațiilor web robuste și scalabile.
* Aplicația va include un sistem robust pentru a actualiza datele în mod constant și eficient direct în baza de date, asigurând că utilizatorii au acces la informații corecte și relevante.
* Oferirea utilizatorilor posibilitatea de a-și actualiza datele personale, cum ar fi poza de profil, nume, locație în cazul în care acestea se schimbă.
* Aplicarea cunoștințelor despre designul sistemelor acumulate în timpul studiilor universitare, împărțind proiectul în sub-task-uri mai mici și gestionând eficient resursele pentru livrarea la timp a proiectului.

## Cerințe funcționale si nefuncționale

### Cerințe funcționale

* **Autentificare și înregistrare**

Un serviciu dedicat va asigura autentificarea securizată a utilizatorilor în sistem, protejând conturile active.

* **Actualizarea informațiilor personale**

Utilizatorii își pot actualiza informațiile personale, cum ar fi inclusiv imaginea de profil, din galeria personală, schimbarea parolei, cat și recuperarea acesteia.

* **Căutare și afișare**

Utilizatorii pot filtra meciuri și pot cauta locații sau utilizatori pentru a-i invita, iar după găsirea informațiilor dorite, pot vedea detalii suplimentare.

* **Vizualizare locație pe hartă**

Utilizatorii trebuie să poată vizualiza locația meciurilor pe o hartă integrată în aplicație, pentru a facilita găsirea terenurilor și a locațiilor meciurilor.

* Operații pentru meci

Utilizatorii trebuie să poată crea, citi, actualiza și șterge detalii despre meciuri. Acest lucru include setarea datei, orei, locației și descrierii meciului.

* Cerere de participare la meci

Utilizatorii trebuie să poată solicita participarea la un meci specific prin trimiterea unei cereri organizatorului meciului.

* Invitare alți utilizatori la meci

Organizatorul meciului trebuie să poată invita alți utilizatori să participe la meciuri, trimițând invitații prin intermediul aplicației.

* Notificări

Utilizatorii vor primi notificări pentru diverse evenimente, cum ar fi atunci când cineva dorește să participe la un meci organizat de ei sau când sunt acceptați pentru a participa la un meci.

### Cerințe nefuncționale

* Fiabilitate

Sistemul trebuie să fie capabil să gestioneze diverse funcționalități în scenarii specifice. În cazul unei erori neașteptate în timpul unui proces de plată, sistemul trebuie să fie capabil să salveze coșul de cumpărături și să fie pregătit să reîncarce datele pentru următoarea încercare.

* Utilizabilitate

Sistemul trebuie să ofere o interfață intuitivă, astfel încât atât utilizatorii obișnuiți, cât și organizatorii să poată naviga și utiliza funcționalitățile cu ușurință.

* Mentenabilitate

Sistemul trebuie să fie proiectat pentru a minimiza efortul de extindere a funcționalităților pe măsură ce apar noi tehnologii. Arhitectura bine structurată va oferi modularitate și reutilizabilitate, facilitând astfel întreținerea.

* Performanță

Sistemul trebuie să minimizeze resursele utilizate pentru a oferi servicii optimizate. Acesta va realiza apeluri la baza de date în cloud numai atunci când este necesar și va salva informațiile temporare în memoria cache locală.

# Studiu bibliografic

Într-o eră dominată de tehnologie, unde stilul de viață sedentar este din ce în ce mai răspândit, aplicațiile care promovează activitatea fizică devin cruciale pentru sănătate și bunăstare. În special, aplicațiile dedicate organizării meciurilor de fotbal pot juca un rol esențial în încurajarea unui stil de viață activ și în facilitarea socializării, în special pentru cei nou-veniți într-un oraș.

Cu toate acestea, în ciuda importanței lor, aceste aplicații nu sunt încă suficient de promovate și există o lipsă de opțiuni variate pe piață. Acest lucru poate reprezenta o barieră pentru cei care caută o modalitate ușoară și convenabilă de a găsi parteneri de joc și de a se implica în activități sportive regulate.

## Aplicația Calendis

### Informații generale

Calendis [4] este o platformă online românească dedicată programărilor, care facilitează interacțiunea dintre clienți și furnizorii de servicii din diverse domenii, precum înfrumusețare, sport și agrement, medicină umană și veterinară, stomatologie, psihologie și multe altele. Platforma oferă o modalitate simplă și eficientă de a găsi și rezerva programări online, eliminând nevoia de a suna sau de a trimite e-mailuri pentru a stabili o întâlnire.

### Mod de funcționare

Aplicațiase dovedește a fi un instrument valoros și în domeniul sportiv, facilitând programarea meciurilor de fotbal într-un mod simplu și eficient. Acestea sunt cateva din functionalitatile aplicatiei:

* **Căutare terenuri de fotbal:** Poți căuta terenuri de fotbal disponibile în orașul tău sau în zona dorită. Platforma îți permite să filtrezi rezultatele în funcție de tipul terenului, dimensiuni, dotări și alte criterii importante pentru tine.
* **Programul terenurilor:** Odată ce ai găsit un teren care te interesează, poți consulta programul acestuia pentru a vedea intervalele orare disponibile.
* **Rezervare teren:** Dacă dorești să rezervi un teren pentru echipa ta, poți face acest lucru direct prin Calendis, alegând data și ora dorite.
* **Înscriere la meciuri organizate:** Dacă vrei să participi la un meci organizat, poți să te înscrii online, confirmând astfel prezența ta. În prezent, nu există o funcționalitate directă de invitare în cadrul aplicației. Organizatorii trebuie să copieze link-ul rezervării și să îl trimită manual potențialilor participanți, care apoi trebuie să acceseze link-ul și să își confirme prezența.
* **Notificări:** Calendis îți va trimite notificări și mementouri înainte de meci.

### Aspecte specifice

Calendis se distinge prin faptul că este, probabil, singura platformă din România care permite rezervarea online a terenurilor de sport. Această caracteristică inovatoare elimină necesitatea apelurilor telefonice sau a vizitelor la fața locului, oferind o soluție modernă și convenabilă pentru iubitorii de sport.

De asemenea, utilizarea Calendis este complet gratuită pentru clienți, ceea ce înseamnă că poți căuta și rezerva terenuri de sport fără niciun cost suplimentar.

Un alt lucru pozitiv al acestei aplicatii este ca e disponibil, atat mobile, cat si web. Designul prietenos al platformei, atât pe desktop, cât și pe mobil, facilitează navigarea și rezervarea rapidă a terenurilor, chiar și pentru cei mai puțin familiarizați cu tehnologia.

Calendis oferă o gamă variată de terenuri de sport, de la fotbal și tenis la baschet și volei, acoperind astfel o paletă largă de preferințe sportive.

Aplicația îți permite să îți organizezi și să îți gestionezi cu ușurință rezervările, oferind opțiuni de anulare, reprogramare și mementouri personalizate. Prin rezervarea online a terenurilor, eviți drumurile inutile și timpul pierdut cu apelurile telefonice, având mai mult timp pentru a te bucura de sportul preferat.

Prin facilitarea accesului la terenuri de sport și prin încurajarea practicării activității fizice, Calendis contribuie la îmbunătățirea sănătății și bunăstării utilizatorilor săi.

## Aplicația FootyAddicts

### Informații generale

Footy Addicts [5] este o platformă socială inovatoare, creată special pentru a aduce împreună iubitorii fotbalului și a facilita organizarea de meciuri amicale sau competiții. Cu o prezență atât în mediul online, cât și pe dispozitive mobile, Footy Addicts oferă o soluție completă pentru pasionații de fotbal de toate nivelurile, indiferent dacă sunt începători sau jucători experimentați. Deși momentan nu este disponibilă în România, această platformă promite să revoluționeze modul în care oamenii se conectează și se bucură de fotbal în întreaga lume.

### Mod de funcționare

Aplicația demonstrează un real potențial și în contextul sportiv, ușurând și optimizând procesul de organizare a meciurilor de fotbal. Printre caracteristicile sale cheie se numără:

* Crearea și căutarea de meciuri: Utilizatorii pot crea rapid și ușor evenimente sportive, specificând detalii precum data, ora, locația, formatul jocului și nivelul de joc dorit. De asemenea, pot căuta meciuri existente în funcție de preferințe, facilitând astfel găsirea rapidă a unui joc potrivit.
* Rezervare teren: Aplicația permite utilizatorilor să rezerve terenuri de fotbal direct prin platformă, alegând data, ora și locația dorită.
* Plata teren sau meci: Utilizatorii pot efectua plăți sigure și rapide pentru rezervarea terenului sau pentru înscrierea la meciuri, direct prin intermediul aplicației
* Înscrierea și gestionarea participării: Jucătorii interesați se pot înscrie la meciurile create de alții, iar organizatorii pot gestiona lista de participanți, confirmând sau respingând cererile de înscriere.
* Comunicare și interacțiune: Footy Addicts încurajează interacțiunea dintre utilizatori prin intermediul funcțiilor de chat și mesagerie privată. Aceasta permite jucătorilor să se cunoască mai bine, să discute despre meciuri și să stabilească strategii de joc.
* Evaluări și recenzii: După fiecare meci, participanții pot evalua și oferi feedback despre ceilalți jucători, contribuind astfel la construirea unei comunități bazate pe fair-play și respect reciproc.
* Profiluri personalizate**:** Fiecare utilizator are un profil personalizat unde își poate afișa informații despre experiența sa în fotbal, poziția preferată, echipa favorită și alte detalii relevante.

### Aspecte specifice

Footy Addicts este mai mult decât o simplă aplicație de organizare a meciurilor de fotbal. Este o comunitate vibrantă, un loc unde pasionații de fotbal se întâlnesc, interacționează și își fac prieteni noi. Prin funcțiile de socializare integrate, utilizatorii pot urmări alți jucători, pot comunica și pot construi relații în jurul pasiunii lor comune pentru fotbal.

Unul dintre punctele forte ale Footy Addicts este interfața sa intuitivă și designul modern. Aplicația este concepută pentru a fi ușor de utilizat de către oricine. Navigarea este simplă și fluidă, iar funcțiile sunt ușor de găsit și de utilizat.

Footy Addicts se adresează în special jucătorilor amatori, oferindu-le o platformă accesibilă și incluzivă pentru a-și organiza meciuri și a găsi parteneri de joc. Indiferent de nivelul tău de experiență sau de abilitățile tale, Footy Addicts te ajută să găsești jocul potrivit pentru tine.

Pentru cei care doresc să își ducă experiența la următorul nivel, Footy Addicts oferă și opțiuni de membership premium. Acestea deblochează o serie de beneficii exclusive, cum ar fi accesul la evenimente și competiții speciale, antrenamente personalizate, reduceri la echipamente sportive și multe altele.

## Aplicația Fubles

### Informații generale

Fubles [6] este o aplicație inovatoare, dedicată organizării și gestionării meciurilor de fotbal amator. Inițial lansată în Italia, platforma a câștigat rapid popularitate la nivel global, devenind un instrument esențial pentru jucătorii de fotbal amatori care doresc să se conecteze și să participe la meciuri locale. Fubles facilitează accesul la fotbalul de masă, oferind o soluție intuitivă și eficientă pentru a găsi parteneri de joc și a organiza meciuri, încrajând astfel un stil de viață activ și sănătos, susținând astfel mișcarea.

### Mod de funcționare

Fubles funcționează pe baza unui sistem de geolocalizare care permite utilizatorilor să găsească și să se alăture meciurilor de fotbal din apropierea lor. Aplicația este disponibilă atât pe dispozitive mobile, cât și pe web, oferind o interfață ușor de utilizat. Utilizatorii pot crea un profil personal, unde își pot adăuga informații despre nivelul lor de joc, poziția preferată pe teren și disponibilitatea pentru meciuri. Aplicația prezinta mai multe funcționalitați, precum :

* Înregistrare: Utilizatorii se pot înregistra folosind adresa de email sau conturile de social media.
* Creare Profil: Fiecare jucător își poate crea un profil detaliat, unde poate împărtăși informații despre experiența sa în fotbal, poziția preferată, echipele favorite și alte detalii relevante.
* Căutare Meciuri: Utilizatorii pot căuta meciuri disponibile în apropierea lor folosind funcția de geolocalizare.
* Înscriere la Meci: Odată găsit un meci potrivit, utilizatorii se pot înscrie și pot confirma participarea.
* Organizare Meciuri: Utilizatorii au posibilitatea de a organiza propriile meciuri, invitând prieteni sau lăsând meciul deschis pentru alți utilizatori ai aplicației.
* **Știri și informații:** Aplicația oferă acces la știri și informații relevante din lumea fotbalului, ținându-te la curent cu cele mai recente evenimente și rezultate.



### Aspecte specifice

Aplicația Fubles dispune de un calendar integrat, o funcționalitate esențială pentru a-ți organiza activitatea fotbalistică. Prin intermediul acestuia, poți să îți planifici participarea la meciuri, să îți gestionezi timpul eficient și să nu ratezi nicio ocazie de a juca. În plus, vei primi notificări automate pentru a-ți aminti de meciurile viitoare, asigurându-te că ești mereu la curent cu programul tău sportiv și că nu ratezi nicio ocazie de a te bucura de fotbal.

De asemenea, Fubles include un sistem de evaluare prin care jucătorii pot fi evaluați de către colegii de echipă și adversari după fiecare meci. Acest sistem ajută la menținerea unei comunități de jucători serioși și la crearea unor meciuri echilibrate.

O altă funcționalitate necesară este aceea că aplicația este disponibilă în mai multe limbi, inclusiv în română, engleză, italiană și spaniolă, pentru a facilita accesul și utilizarea de către o comunitate globală de pasionați de fotbal. Poate fi descărcată gratuit de pe App Store și Google Play, iar pentru cei care preferă să acceseze serviciul de pe un computer, există și o versiune web ușor de utilizat.

# Analiză și fundamentare Teoretică

În cadrul acestui capitol, vom explora în detaliu principiile funcționale care stau la baza aplicației dezvoltate. Vom analiza și fundamenta teoretic soluția propusă, prezentând algoritmii utilizați pentru căutarea și filtrarea meciurilor, protocoalele de comunicare implementate pentru interacțiunea dintre utilizatori și server, precum și modelele abstracte care au stat la baza designului aplicației. Vom oferi explicații și argumentări logice pentru alegerile făcute în procesul de dezvoltare, evidențiind avantajele și beneficiile soluției alese. Totodată, vom detalia structura logică și funcțională a aplicației, prezentând modul în care diferitele componente interacționează pentru a furniza o experiență optimă utilizatorilor.

## Tehnologii utilizate

Java [7] este un limbaj de programare orientat-obiect lansat în anul 1995 de către James Gosling la Sun Microsystem (acum filiala Oracle). La nivel global Java este platforma pe care se dezvoltă cele mai multe aplicații software. Unul dintre principalele avantaje ale programelor scrise în Java este acela că pot fi rulate fară modificări pe orice platformă care are instalată o mașină virtuală Java. Am ales pentru dezvoltarea backend-ului aplicației datorită maturității, stabilității și performanței sale. Java 17 oferă o serie de îmbunătățiri față de versiunile anterioare, inclusiv noi funcționalități de limbaj, optimizări ale performanței și îmbunătățiri ale securității.

Spring Boot [8] este un framework open-source care simplifică semnificativ dezvoltarea de aplicații Java, eliminând o mare parte din configurația manuală și boilerplate-ul specific framework-ului Spring, astfel încât programatorul să se poată concentra asupra dezvoltării efective de cod. Spring are la bază un principiu arhitectural important, și anume inversiunea controlului. Inversiunea controlului, cunoscută și ca injectarea dependinței oferă mijloacele necesare pentru a organiza obiectele intr-o aplicație Java. Obiectele iși definesc dependențele, alte obiecte către care păstrează referințe cu ajutorul argumentelor de constructori, argumentelor de metode ‘factory’ sau a proprietăților care sunt setate după ce obiectul a fost construit. Injectarea acestor dependențe este realizată de containerul care se ocupă de gestiunea obiectelor.

Maven [9] reprezintă un instrument de gestiune și integrare a unui proiect software. Bazându-se pe conceptul de POM (Project Object Model), Maven poate gestiona construirea, raportarea si documentarea unui proiect. Un POM este unitatea fundamentala de lucru in Maven si este de fapt un fișier XML, ce contine informații despre două aspecte ale unui proiect: în primul rând este specificat modul în care este construit proiectul prin detaliile de configurare din fisierul POM și în al doilea rând sunt specificate in pom dependințele proiectului. Referitor la dependințe, Maven descarcă dinamic într-un repository local librăriile descrise în fișierul POM din repository-uri centrale cum ar fi Maven 2 Central Repository

MySQL [10] este un sistem de gestionare a bazelor de date relaționale (RDBMS) care rulează ca server oferind acces multi-utilizator la diferite baze de date. SQL este o abreviere de la “Structured Query Language”, adică limbaj de interogare structurat. SQL a fost inițial dezvoltat pentru a opera pe date în bazele de date care urmează modelul relațional. Este un limbaje de programare pentru interogarea, modificarea și gestionarea datelor. MySQL este cel mai comun instrument de baze de date open source. Este considerat un program simplu și ușor de folosit în comparație cu alte programe de baze de date. MySQL oferă diverse diferite programe care sunt legate de baza de date.

Vue.js [11] este un framework JavaScript progresiv, conceput pentru a simplifica și eficientiza dezvoltarea interfețelor utilizator (UI) moderne. Flexibilitatea sa permite utilizarea atât pentru proiecte mici, cât și pentru aplicații web complexe, scalabile și performante.Unul dintre pilonii fundamentali ai Vue.js este arhitectura sa bazată pe componente. Aplicațiile Vue.js sunt construite din componente reutilizabile, fiecare componentă având propriul său șablon HTML, logică JavaScript și stiluri CSS. Această modularitate face codul mai ușor de organizat, întreținut și reutilizat, îmbunătățind astfel productivitatea și scalabilitatea aplicației.

TypeScript [12] este un limbaj de programare dezvoltat si susținut de către Microsoft. Este un superset al limbajului de programare Javascript. Javascript este un limbaj de programare bazat pe conceptul prototipurilor. Limbajul este binecunoscut pentru folosirea sa in dezvoltarea aplicațiilor web.

Bootstrap [13] este un set de instrumente folosit în dezvoltarea aplicațiilor care folosesc Javscript, HTML si CSS. Este foarte folositor la crearea interfeței cu utilizatorul, oferind implementare pentru diferite componenete (butoane, tabele, etc).

CSS (Cascadinf Style Sheets) [14] este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului, prin elementul <style> și/sau atributul style. CSS se poate utiliza si pentru formatarea elementelor XHTML, XML si SVGL.

## Mediu de dezvoltare

Intellij IDEA [15] este un mediu de dezvoltare integrat Java pentru dezvoltarea de aplicații software. Este dezvoltat de către JetBrains și este disponibil atât ca și variantă gratuită cât și cu licență comercială. Ambele versiuni pot fi folosite pentru dezvoltarea de aplicații software comerciale. Prima versiune de Intellij IDEA a fost lansată în ianuarie 2001, și a fost unul din primele medii de dezvoltare integrate Java care a oferit facilități de restructurare a codului unei aplicații. Este cel mai popular mediu de dezvoltare integrat Java.

MySQL Workbench [16], Instrumentul Esențial pentru Modelarea și Administrarea Bazelor de Date MySQL. Acesta este un mediu de dezvoltare integrat vizual, special conceput pentru a facilita lucrul cu baze de date MySQL. Dezvoltat de Oracle, MySQL Workbench oferă o gamă largă de instrumente și funcționalități care acoperă întregul ciclu de viață al unei baze de date, de la proiectare și modelare până la dezvoltare, administrare și întreținere.

Visual Studio Code [17] este un editor de cod open-source, dezvoltat de Microsoft, care oferă suport pentru o gamă largă de limbaje de programare și tehnologii. Datorită extensibilității sale și a numeroaselor extensii disponibile, Visual Studio Code poate fi personalizat pentru a se adapta nevoilor specifice ale fiecărui dezvoltator. În contextul aplicației Sport Planner, Visual Studio Code este utilizat pentru dezvoltarea frontend-ului Vue.js.

## Protocoale utilizate

### Protocolul de encriptare al parolelor

Un aspect esențial al securității datelor dintr-o bază de date este reprezentat de criptarea acestora, în special a parolelor utilizatorilor. În cadrul aplicației (backend), acest mecanism va fi implementat înainte de salvarea datelor în baza de date atunci când un utilizator se înregistrează. Criptarea parolelor este crucială pentru a asigura protecția credentialelor utilizatorilor, deoarece fără aceasta, oricine are acces la baza de date, autorizat sau nu (inclusiv hackeri), ar putea vizualiza cu ușurință parolele stocate, rulând un simplu script SQL de selectare a datelor.

Indiferent de cât de complexă este o parolă, cu multiple caractere speciale ($, /, @, etc.), generată, de exemplu, cu ajutorul unui generator de parole, dacă aceasta nu este criptată înainte de a fi stocată în baza de date, este inutilă. Dacă cineva poate citi parola din baza de date, o poate utiliza pur și simplu prin copierea și lipirea acesteia – indiferent de complexitatea ei. Criptarea schimbă radical parola înainte de a fi stocată, astfel încât, în cazul unui atac asupra serverului, în loc de un text clar precum „parola123”, atacatorul va găsi o serie aleatorie de litere și cifre.

Criptarea parolelor implică mai multe componente esențiale:

* Cheie: Utilizată pentru criptarea și decriptarea parolelor, aceasta constă într-un șir aleatoriu de biți. Algoritmul se bazează pe chei private și publice pentru a cripta și decripta datele diferit.
* Biți: O stare logică cu una dintre cele două valori posibile (1/0, da/nu),
* Cifră pe bloc: Un algoritm determinist care funcționează pe grupuri de biți de lungime fixă, denumite blocuri.
* Funcția hash: Algoritmul care folosește cheia pentru a cripta și decripta parola. O funcție hash este o bucată de cod care rulează de fiecare dată când o parolă este salvată sau când un utilizator se conectează la aplicație.
* Hash: O serie aleatorie de litere și cifre care reprezintă parola criptată. Funcția hash folosește acest hash în locul parolei brute pentru autentificare.

Procesul de criptare funcționează astfel: de fiecare dată când se salvează o parolă nouă la înregistrare, o funcție hash creează un hash al parolei și îl salvează în baza de date. La fiecare încercare de autentificare a utilizatorului, funcția hash recreează hash-ul pentru a verifica dacă acesta se potrivește cu cel stocat. Dacă hash-urile coincid, autentificarea este validată și utilizatorul este conectat.

A diagram of a computer virus

Description automatically generated

Figura 4.1 Procesul de encriptare si decriptare [18]

Cu toate acestea, criptarea simplă a parolelor, doar cu funcții hash, nu este suficient de rezistentă la hacking. Funcția hash generează un hash unic pentru fiecare parolă, nu pentru fiecare utilizator. Astfel, dacă mai mulți utilizatori au aceeași parolă, de exemplu „Pa$$w0rd123”, hash-ul parolei va fi identic pentru fiecare utilizator. Pentru a depăși această vulnerabilitate, se poate folosi un pas suplimentar în criptarea parolelor, numit „salting”, care adaugă o valoare unică de 8 octeți (16 caractere) la parolă înainte de crearea hash-ului. În acest mod, chiar și parolele identice vor avea hash-uri unice, asigurând astfel un nivel superior de securitate.

### Protocolul Hypertext Transfer (HTTP)

HTTP [18], cunoscut sub numele de Hypertext Transfer Protocol, este protocolul utilizat de browserele web (clientul) și serverele web (gazda aplicației) pentru a permite accesul și interacțiunea cu aplicațiile web. Acest protocol text-based facilitează cererea și recepția resurselor, precum documente HTML sau fișiere text (de exemplu, PDF). Odată ce serverul gazdă a deservit cererea, conexiunea dintre client și server este întreruptă, ceea ce înseamnă că pentru fiecare nouă cerere trebuie inițiată una noua.

Comunicarea între clienți și servere se realizează prin schimbul de mesaje individuale. O cerere este inițiată de un user-agent, care de obicei este un browser web, dar poate fi și un bot sau un scraper. Serverul procesează cererea și răspunde în consecință. Pot exista diverse proxy-uri sau cache-uri care acționează ca gateway-uri.

HTTP este un protocol fără stare (stateless), ceea ce înseamnă că nu menține informațiile între cereri. Totuși, pentru a facilita sesiuni persistente, HTTP folosește cookie-uri.

Metodele HTTP sunt comenzi utilizate de client pentru a interacționa cu serverul web. Cele mai comune metode sunt:

* GET: Această metodă este folosită pentru a solicita și prelua informații de la server. De exemplu, atunci când accesezi o pagină web, browser-ul tău trimite o cerere GET către server pentru a obține conținutul paginii.
* POST: Această metodă este utilizată pentru a trimite date către server, de obicei pentru a crea o nouă resursă sau pentru a actualiza o resursă existentă. De exemplu, atunci când completezi un formular online și îl trimiți, datele sunt trimise către server printr-o cerere POST.
* PUT: Această metodă este similară cu POST, dar este folosită în principal pentru a actualiza o resursă existentă. PUT înlocuiește complet resursa existentă cu datele noi trimise în cerere.
* DELETE: Această metodă este folosită pentru a șterge o resursă specificată de pe server. De exemplu, dacă dorești să ștergi un comentariu pe o pagină web, poți trimite o cerere DELETE către server pentru a elimina comentariul respectiv.

**Structura unei cereri HTTP:**

* **Metoda HTTP:** Definește operațiunea dorită de client.
* Calea resursei: Identifică resursa solicitată.
* Versiunea HTTP: Specifică versiunea protocolului utilizat.
* Anteturi opționale: Transmit informații suplimentare către server.
* Corpul mesajului: Pentru metodele care îl necesită, cum ar fi POST.

A screenshot of a computer

Description automatically generated

Figura 4.2 Structura unei cereri http [20]

**Structura unui Răspuns HTTP:**

Versiunea HTTP: Specifică versiunea protocolului utilizat în răspuns.

Cod de stare: Indicativ pentru succesul sau eșecul cererii și motivul acestuia.

Mesaj de stare: O scurtă descriere a codului de stare.

Anteturi HTTP: Similar cu cele din cereri, conțin informații adiționale.

Corpul mesajului: Opțional, conține resursa solicitată.

A black and grey rectangular sign with white text

Description automatically generated

Figura 4.3 Structura unui răspuns http [21]

Acest protocol este fundamentul comunicării pe web, facilitând interacțiunea între clienți și servere prin metodele sale bine definite.

### Protocolul Simple Mail Transfer (SMTP)

Simple Mail Transfer Protocol (SMTP) [19] este un protocol de rețea fundamental, situat la nivelul aplicației în stiva de protocoale TCP/IP, care guvernează transmiterea mesajelor de e-mail între servere și agenți de transfer de mesaje (MTA). În esență, SMTP este responsabil pentru trimiterea eficientă și fiabilă a e-mailurilor de la expeditor la destinatar, asigurându-se că mesajele ajung la destinație în siguranță și în timp util. Principalele caracteristici ale acestui protocol sunt:

* **Modelul Client-Server**: SMTP funcționează pe un model client-server în care clientul de e-mail (expeditorul) acționează ca un client SMTP, iar serverul de e-mail (destinatarul) acționează ca un server SMTP.
* Protocol de Tip Store-and-Forward: SMTP utilizează un model de tip store-and-forward, ceea ce înseamnă că mesajele de e-mail sunt stocate temporar pe servere intermediare înainte de a fi retransmise către serverul de destinație finală.
* Protocol Stateless: SMTP fără stare, ceea ce înseamnă că fiecare sesiune SMTP este independentă și nu menține informațiile între conexiuni succesive.
* Coduri de Răspuns SMTP: SMTP folosește coduri de răspuns numerice pentru a indica succesul sau eșecul diferitelor operațiuni. De exemplu, 250 OK indică succesul unei operațiuni.
* **Fiabilitate**: SMTP asigură livrarea fiabilă a e-mailurilor prin mecanisme precum reîncercări și gestionarea erorilor. Dacă un e-mail nu poate fi livrat imediat, serverul încearcă să-l retrimită la intervale regulate.

A diagram of a computer network

Description automatically generated

Figura 4.4 Flux-ul de protocolului SMTP [23]

Procesul de trimitere a unui e-mail folosind SMTP poate fi împărțit în mai mulți pași:

* **Stabilirea Conexiunii**: Clientul inițiază o conexiune la serverul SMTP folosind TCP.
* **Comanda HELO/EHLO**: Odată ce conexiunea este stabilită, clientul trimite o comandă HELO sau EHLO către server pentru a se identifica.
* Serverul confirmă cu un răspuns 250 OK dacă adresa expeditorului este acceptabilă.
* **Comanda MAIL FROM**: Clientul trimite comanda MAIL FROM pentru a specifica adresa de e-mail a expeditorului.
* Serverul confirmă cu un răspuns 250 OK dacă adresa expeditorului este acceptabilă.
* **Comanda RCPT TO**: Clientul trimite comanda RCPT TO pentru a specifica adresa de e-mail a destinatarului.
* Serverul răspunde cu un răspuns 250 OK dacă adresa destinatarului este validă și poate fi acceptată.
* **Comanda DATA**: Clientul trimite comanda DATA pentru a indica faptul că conținutul e-mailului va urma.
* Serverul răspunde cu un răspuns 354 Start mail input, solicitând clientului să trimită datele e-mailului.
* **Trimiterea Conținutului E-mailului**: Clientul trimite antetele și corpul e-mailului, iar sfârșitul datelor este indicat printr-un rând care conține doar un punct (.).
* Serverul răspunde cu un răspuns 250 OK dacă e-mailul a fost primit cu succes.
* **Comanda QUIT**: Clientul trimite comanda QUIT pentru a încheia sesiunea SMTP.
* A diagram of a mail server

  Description automatically generatedServerul confirmă cu un răspuns 221 Bye și apoi închide conexiunea.

Figura 4.5 Etapele trimiterii unui email [24]

### Protocolul WebSockets

WebSocket-urile [20] reprezintă un protocol de comunicare care oferă canale de comunicare bidirecțională printr-o singură conexiune TCP. Spre deosebire de HTTP-ul tradițional, care urmează un model cerere-răspuns, WebSocket-urile permit atât clientului (browser-ul web) cât și serverului să inițieze comunicarea, trimițând date în mod continuu, fără a fi nevoie de interogări constante (polling).

Web-ul timpuriu a fost construit în principal pe HTTP, ideal pentru preluarea de conținut static. Cu toate acestea, pe măsură ce web-ul a evoluat într-o platformă pentru aplicații dinamice și interactive, limitările HTTP-ului au devenit evidente. Tehnici precum polling-ul și long-polling-ul au fost utilizate pentru a simula actualizări în timp real, dar erau ineficiente și introduceau latență. WebSocket-urile au apărut ca o soluție, permițând conexiuni persistente și eficiente pentru o comunicare reală în timp real.

Caracteristicile de baza prin care acest protocol s-a remarcat sunt:

* **Comunicație** : WebSockets permit atât clientului, cât și serverului să trimită date în același timp, fără a fi nevoie să aștepte răspunsuri, spre deosebire de modelul tradițional request-response al HTTP.
* **Persistența Conexiunii**: Odată ce conexiunea WebSocket este stabilită, aceasta rămâne deschisă, eliminând nevoia de a reconstrui conexiunea pentru fiecare schimb de date.
* **Eficiență**: WebSockets reduc semnificativ overhead-ul comparativ cu HTTP, deoarece evită repetarea headerelor de fiecare dată când se trimit date. Aceasta face ca WebSockets să fie ideale pentru aplicații în timp real, cum ar fi jocurile multiplayer, aplicațiile de chat, și actualizările live ale pieței de acțiuni.

A close-up of a computer screen

Description automatically generated

Figura 4.6 Comunicare client-server prin websockets [26]

WebSocket-urile, prin designul lor ingenios, permit un schimb dinamic și eficient de informații între client și server. Principalele operații implicate în funcționarea WebSocket-urilor sunt:

* Strângere de mână (Handshaking): Conexiunea WebSocket începe cu o cerere HTTP standard, inclusiv un antet Upgrade care indică dorința de a trece la protocolul WebSocket. Dacă serverul este de acord, trimite un răspuns care confirmă upgrade-ul, iar conexiunea trece la WebSocket-uri.
* Încadrarea mesajelor (Message Framing): Mesajele WebSocket constau dintr-unul sau mai multe cadre (frames). Fiecare cadru are un antet care indică tipul său (text, binar, control), lungimea și dacă este cadrul final al mesajului. Această încadrare permite o transmitere eficientă și fiabilă a datelor.
* Gestionarea erorilor și deconectări: Protocolul definește coduri specifice pentru erori și închideri corecte, permițând atât clientului, cât și serverului să gestioneze corect deconectările.

În lumea WebSocket-urilor, interacțiunea dintre client și server este esențială. Clientul, reprezentat de obicei de browser-ul web, inițiază conexiunea, trimite mesaje către server și primește răspunsuri în timp real. Pentru a facilita această comunicare, browser-ul utilizează API-ul WebSocket, un instrument puternic care permite schimbul bidirecțional de date. De partea cealaltă, serverul are rolul de a asculta cererile de conexiune, de a gestiona toți clienții conectați și de a facilita comunicarea între aceștia. Fie că este vorba de mesaje individuale sau de difuzarea de informații către toți clienții simultan, serverul orchestrează întregul proces de comunicare.

Securitatea este o prioritate în cadrul protocolului WebSocket. Pentru a proteja confidențialitatea și integritatea datelor transmise, WebSocket-urile utilizează două mecanisme esențiale. Primul este schema "wss://", similară cu HTTPS. Aceasta stabilește o conexiune securizată prin TLS (Transport Layer Security), criptând toate datele în tranzit pentru a preveni interceptarea și accesul neautorizat. Al doilea mecanism este verificarea originii. Serverul poate verifica antetul "Origin" al unei cereri WebSocket pentru a se asigura că conexiunea provine dintr-o sursă autorizată, prevenind astfel atacurile de tip cross-site scripting (XSS) și alte vulnerabilități.

Acest protocol nou dezvoltat, conceptului WebSocket, se remarcă printr-o serie de beneficii si avantaje competitive unice, cum ar fi:

* Performanță Îmbunătățită: Prin reducerea "overhead-ului" (informații suplimentare) asociat protocolului, WebSocket-urile permit un transfer rapid și eficient al datelor, ceea ce este esențial pentru aplicațiile care necesită actualizări frecvente și în timp real.
* Scalabilitate: WebSocket-urile pot gestiona un număr mare de conexiuni simultane, făcându-le ideale pentru aplicații care trebuie să suporte un număr mare de utilizatori activi în același timp.
* Simplitate: Modelul de programare al WebSocket-urilor este relativ simplu, permițând dezvoltatorilor să creeze aplicații în timp real fără să se confrunte cu complexitățile tradiționale ale comunicării pe internet.

WebSocket-urile reprezintă o tehnologie versatilă, utilizată pe scară largă pentru a îmbunătăți experiența utilizatorului în diverse domenii. În aplicațiile de chat, acestea permit comunicarea instantanee, asigurând schimbul rapid de mesaje între utilizatori. În jocurile online, WebSocket-urile oferă o experiență multiplayer fluidă, sincronizând în timp real acțiunile jucătorilor și starea jocului. Pe piețele financiare, acestea sunt esențiale pentru a furniza actualizări live ale cotațiilor și a permite luarea rapidă a deciziilor. În plus, instrumentele de colaborare online, precum editoarele de text și aplicațiile de design, utilizează WebSocket-uri pentru a permite mai multor utilizatori să lucreze simultan la același document, vizualizând în timp real modificările efectuate de ceilalți.

Așadar, protocolul WebSockets oferă o metodă eficientă și modernă de a realiza comunicații bidirecționale între un client și un server printr-o conexiune persistentă. Prin eliminarea limitărilor modelului request-response al HTTP și oferind un canal de comunicație cu latență scăzută și overhead minim, WebSockets au devenit esențiale pentru dezvoltarea aplicațiilor care sunt interactive și necesita comunicare în timp real.

### JSON Web Token (JWT)

JSON Web Token (JWT) [21] este un standard deschis (RFC 7519) care definește o metodă compactă și autonomă pentru transmiterea securizată a informațiilor între părți, sub forma unui obiect JSON. Aceste informații pot fi verificate și încredințate deoarece sunt semnate digital, permițând autentificarea și schimbul de informații într-o manieră sigură și eficientă.

Un JSON Web Token este compus din trei părți care sunt codificate în Base64-URL și concatenate cu puncte (.) pentru a forma un JWT complet. Acestea sunt:

* Antet (Header): Conține informații despre tipul token-ului (JWT) și algoritmul de criptare folosit pentru semnătură (de exemplu, HMAC SHA256 sau RSA).
* Sarcină utilă (Payload): Conține setul de revendicări (claims). Revendicările sunt declarații despre o entitate (de obicei, utilizatorul) și metadate suplimentare. Există trei tipuri de revendicări: înregistrate, publice și private.
* Semnătură (Signature): O semnătură digitală creată prin combinarea antetului, sarcinii utile și o cheie secretă cunoscută doar de server. Semnătura este utilizată pentru a verifica autenticitatea și integritatea token-ului.

A diagram of a structure of a json web token

Description automatically generated

Figura 4.7 Stuctura unui JWT [28]

Pentru a înțelege cum funcționează JWT-urile în practică, vom analiza operațiile cheie implicate în utilizarea lor într-o aplicație:

* **Autentificare:** Utilizatorul își introduce datele de autentificare (de exemplu, nume de utilizator și parolă).
* **Generare token:** Dacă autentificarea este reușită, serverul generează un JWT care conține informații despre utilizator (de exemplu, ID utilizator, roluri, permisiuni) și îl trimite înapoi clientului.
* **Stocare token:** Clientul stochează JWT-ul, de obicei în memoria locală (localStorage) sau în cookie-uri.
* **Trimitere token:** Pentru fiecare cerere ulterioară către server, clientul include JWT-ul în antetul HTTP (de obicei, în câmpul Authorization).
* **Verificare token:** Serverul verifică semnătura JWT-ului folosind cheia sa secretă. Dacă semnătura este validă și token-ul nu a expirat, serverul acceptă cererea și acordă acces la resursele solicitate.

A diagram of a server

Description automatically generated



Figura 4.8 Flux de operații pentru login cu jwt [29]

Scopul principal al JSON Web Tokens (JWT) este de a oferi un mecanism sigur, compact și eficient pentru transmiterea informațiilor între părți, în special în contextul autentificării și autorizării utilizatorilor în aplicații web și mobile.

JWT joacă un rol central în procesul de autentificare și autorizare, îndeplinind următoarele funcții:

* **Autentificare:** După ce un utilizator se autentifică cu succes (de exemplu, prin introducerea numelui de utilizator și a parolei), serverul generează un JWT care conține informații despre identitatea utilizatorului și îl trimite înapoi clientului. Acest token servește ca dovadă a autentificării utilizatorului.
* **Autorizare:** JWT-ul poate conține informații despre rolurile și permisiunile utilizatorului. Când utilizatorul încearcă să acceseze o resursă protejată, serverul poate verifica revendicările din JWT pentru a determina dacă utilizatorul are permisiunea necesară.
* **Transmiterea informațiilor:** JWT-urile pot fi utilizate pentru a transmite în siguranță orice tip de informații între părți, cum ar fi date despre profilul utilizatorului, preferințe sau alte date relevante pentru aplicație.

Există mai multe motive pentru care JWT-urile sunt o alegere populară pentru autentificare și autorizare:

* **Fără stocare pe server (Stateless):** JWT-urile sunt auto-conținute, ceea ce înseamnă că serverul nu trebuie să stocheze informații despre sesiune. Acest lucru simplifică arhitectura serverului și îmbunătățește scalabilitatea.
* **Securitate:** JWT-urile pot fi semnate digital pentru a garanta integritatea și autenticitatea datelor. În plus, pot fi criptate pentru a proteja confidențialitatea informațiilor sensibile.
* **Portabilitate:** Fiind doar șiruri de caractere, JWT-urile sunt ușor de transmis între diferite sisteme și platforme.
* **Flexibilitate:** JWT-urile pot conține diverse tipuri de informații și pot fi personalizate pentru a se potrivi nevoilor specifice ale aplicației.
* **Suport larg:** JWT-urile sunt un standard deschis, suportat de o gamă largă de limbaje de programare, framework-uri și biblioteci, ceea ce facilitează implementarea lor.

Așadar, JSON Web Tokens (JWT) oferă o soluție modernă, flexibilă și sigură pentru autentificarea și autorizarea utilizatorilor în aplicații. Prin eliminarea necesității de stocare a sesiunilor pe server, JWT-urile simplifică arhitectura aplicațiilor și îmbunătățesc scalabilitatea. În plus, caracteristicile de securitate și portabilitate ale JWT-urilor le fac o alegere ideală pentru aplicațiile moderne care necesită un mecanism robust de autentificare și autorizare.

## Arhitectura microservicii

Arhitectura microservicii (figura 4.9) este un stil arhitectural care structurează o aplicație ca o colecție de servicii mici, independente și care se pot desfășura separat. Fiecare microserviciu este responsabil pentru o funcționalitate specifică a aplicației și comunică cu alte microservicii prin intermediul API-urilor bine definite. Acest model arhitectural vine în contrast cu arhitectura monolitică tradițională, unde toate componentele aplicației sunt strâns legate și desfășurate împreună.

A diagram of a data flow

Description automatically generated

Figura 4.9 Model de arhitectură cu microservicii [30]

În esență, arhitectura microserviciilor descompune aplicațiile monolitice, adesea dificil de gestionat, în servicii independente, fiecare dedicat unei funcționalități specifice. Un serviciu se poate ocupa de autentificarea utilizatorului, altul de gestionarea catalogului de produse, iar altul de procesarea plăților. Această structură modulară aduce numeroase avantaje.

Dimensiunea redusă a serviciilor constituente permite ca acestea să fie construite de la început de una sau mai multe echipe mici, delimitate de granițele serviciilor. Acest lucru facilitează scalarea efortului de dezvoltare atunci când este necesar. Odată dezvoltate, aceste servicii pot fi implementate independent unul de celălalt, ceea ce simplifică identificarea și scalarea serviciilor intens solicitate, fără a afecta întreaga aplicație.

Fiecare microserviciu funcționează ca un modul autonom, permițând echipelor de dezvoltare să utilizeze tehnologiile și limbajele de programare cele mai potrivite pentru sarcina respectivă. Această independență permite, de asemenea, actualizări și îmbunătățiri izolate, fără a afecta întregul sistem. Astfel, aplicația poate evolua rapid și se poate adapta la cerințele în schimbare ale utilizatorilor sau ale pieței.

Comunicarea între microservicii se realizează prin intermediul unor interfețe de programare a aplicațiilor (API-uri) bine definite. Aceste API-uri acționează ca puncte de contact standardizate, facilitând schimbul de informații și cereri între servicii. Această abordare asigură interoperabilitatea și flexibilitatea în integrarea diferitelor componente ale aplicației.

A diagram of microservices architecture

Description automatically generated

Figura 4.10 Arhitectură monolitică vs arhitectură cu microservicii [31]

Arhitectura microserviciilor oferă o serie de avantaje semnificative în dezvoltarea și întreținerea aplicațiilor complexe:

* Scalabilitate: Posibilitatea de a adăuga instanțe individuale ale microserviciilor specifici, în funcție de cerere, optimizează utilizarea resurselor și asigură o adaptare eficientă la fluctuațiile de trafic.
* Reziliență Sporită: Izolarea erorilor la nivelul microserviciilor limitează impactul asupra întregii aplicații. Reimplementarea serviciului afectat este suficientă pentru remedierea erorilor, fără a fi necesară reimplementarea întregii aplicații.
* Flexibilitate Tehnologică: Echipele de dezvoltare au libertatea de a alege cele mai potrivite tehnologii (limbaje de programare, baze de date etc.) pentru fiecare microserviciu în parte. Această abordare elimină constrângerile unei soluții standardizate și permite utilizarea celor mai eficiente instrumente pentru fiecare funcționalitate specifică.
* Dezvoltare și Implementare Agile: Dimensiunea redusă a microserviciilor facilitează dezvoltarea rapidă și independentă, permițând echipelor să lucreze autonom și să livreze actualizări mai frecvent.
* Mentenabilitate Simplificată: Structura modulară a microserviciilor permite o înțelegere și o gestionare mai ușoară a codului, facilitând întreținerea și depanarea aplicației.

În concluzie, arhitectura microserviciilor reprezintă o abordare modernă și eficientă pentru dezvoltarea de aplicații complexe, oferind flexibilitate, scalabilitate, mentenabilitate sporită și o reziliență mai mare în fața erorilor. Prin descompunerea aplicației în module independente și interconectate, această arhitectură permite echipelor de dezvoltare să lucreze mai eficient și să livreze produse mai bune utilizatorilor finali.

## Design Pattern Model-View-Controller (MVC)

Pattern-ul de design Model-View-Controller (MVC) este un model arhitectural utilizat în dezvoltarea aplicațiilor software, în special a aplicațiilor web. Acest pattern separă aplicația în trei componente principale: Model, View și Controller. Scopul MVC este de a separa logica de afaceri de prezentarea datelor și de interacțiunea cu utilizatorul, facilitând astfel dezvoltarea, testarea și mentenanța aplicațiilor.

Acest design pattern conține trei componente mari și importante:

* **Model**: Reprezintă logica de afaceri și manipularea datelor aplicației. Modelul gestionează accesul la date, validările și interacțiunile cu baza de date sau alte surse de date. Modelul notifică View-ul și Controller-ul atunci când datele sunt actualizate, permițând astfel prezentarea informațiilor actualizate utilizatorului.
* **View**:Reprezintă prezentarea datelor și interfața cu utilizatorul. View-ul este responsabil pentru afișarea datelor din Model într-un format adecvat și pentru captarea inputului utilizatorului. View-ul este pasiv și nu conține logică de afaceri; el doar preia datele furnizate de Model și le prezintă utilizatorului.
* **Controller**: Reprezintă intermediarul între Model și View. Controller-ul gestionează cererile utilizatorului, invocă metodele corespunzătoare din Model și selectează View-ul adecvat pentru a prezenta rezultatele. Controller-ul procesează inputul utilizatorului și actualizează Modelul în consecință, asigurând astfel separarea clară între logica de prezentare și logica de afaceri.

**A diagram of a system

Description automatically generated**

**Figura 4.11 Stuctura pattern-ului MVC** [32]

Interacțiunea dintre utilizator și aplicație se desfășoară într-un flux bine definit, implicând toate cele trei componente:

* **Interacțiunea Utilizatorului**: Utilizatorul interacționează cu aplicația prin intermediul View-ului (de exemplu, prin apăsarea unui buton).
* **Procesarea Cererii**: Controller-ul preia cererea utilizatorului și o procesează, invocând metodele corespunzătoare din Model pentru a manipula datele sau a efectua acțiuni specifice.
* **Actualizarea Modelului**: Modelul actualizează datele și notifică View-ul despre schimbările survenite.
* **Actualizarea View-ului**: View-ul preia datele actualizate din Model și le prezintă utilizatorului într-un format adecvat.

Avantajele pattern-ului MVC sunt numeroase și contribuie semnificativ la eficiența dezvoltării aplicațiilor software. În primul rând, separarea responsabilităților este un aspect crucial, deoarece MVC separă clar logica de afaceri, logica de prezentare și interacțiunea cu utilizatorul, facilitând astfel dezvoltarea și mentenanța aplicației. Testabilitatea este un alt avantaj major, deoarece componenta Model este independentă de View și Controller, permițând testarea unitară și izolată a logicii de afaceri. Flexibilitatea și reutilizarea sunt de asemenea importante, deoarece View-urile și Controller-ele pot fi reutilizate și modificate independent, fără a afecta logica de afaceri din Model. În cele din urmă, MVC oferă o structură organizată și modulară, facilitând scalarea aplicației pe măsură ce cerințele de afaceri evoluează, asigurând astfel o organizare eficientă și scalabilitate pentru aplicațiile complexe.

Pattern-ul de design MVC (Model-View-Controller) oferă o structură clară și modulară pentru dezvoltarea aplicațiilor software, facilitând separarea logicii de afaceri de logica de prezentare și interacțiunea cu utilizatorul. Prin implementarea MVC, dezvoltatorii pot crea aplicații mai ușor de testat, de întreținut și de scalat, beneficiind de avantajele separării responsabilităților și reutilizării componentelor. Deși poate introduce o complexitate inițială, pattern-ul MVC rămâne un standard popular și eficient în dezvoltarea aplicațiilor web și desktop moderne.

# Proiectare de detaliu și implementare

Aplicația va fi structurată într-o arhitectură pe microservicii, cuprinzând trei nivele distincte, dar interconectate, fiecare cu responsabilități specifice, detaliate pe parcursul acestui capitol (figura 5.1):

* Nivel baza de date**:** Având în vedere natura distribuită a microserviciilor, fiecare serviciu va deține propria bază de date, optimizată pentru nevoile sale specifice. Acest lucru asigură o mai mare flexibilitate și scalabilitate, permițând fiecărui serviciu să evolueze independent. Vom explora în detaliu structura fiecărei baze de date, tabelele și relațiile dintre acestea, precum și tehnologiile de stocare alese pentru fiecare microserviciu.
* Nivel de server: Aceste microservicii reprezintă nucleul aplicației, gestionând logica de business și comunicarea între stratul de prezentare și straturile de date. Fiecare microserviciu va fi responsabil pentru o anumită funcționalitate, cum ar fi autentificarea utilizatorilor, gestionarea meniurilor restaurantelor sau procesarea comenzilor. Vom detalia implementarea fiecărui microserviciu, inclusiv tehnologiile utilizate, API-urile expuse și modul în care acestea interacționează între ele.
* Nivel de client: Acest strat este reprezentat de o interfață web accesibilă prin intermediul unui browser (Chrome, Mozilla, Microsoft Edge etc.). Utilizatorii vor putea interacționa cu aplicația doar după autentificare, iar interfața va fi construită folosind HTML, CSS și JavaScript pentru a oferi o experiență optimizată și ușor de utilizat. Vom detalia structura paginilor web, stilurile CSS și funcționalitățile JavaScript implementate pentru a crea o interfață intuitivă și responsivă.

**A diagram of a computer

Description automatically generated**

Figura 5.1 Principalele nivele din arhitectura [33]

## Arhitectura aplicației

Arhitectura aplicației SportPlanner este concepută într-un model de microservicii, o abordare modernă care promovează flexibilitatea, scalabilitatea și mentenabilitatea. Această structură descompune aplicația în componente independente, fiecare cu responsabilități bine definite, comunicând între ele prin intermediul unui API Gateway, după cum se vede și în figura 5.2. Acest API Gateway servește ca punct unic de intrare pentru toate cererile venite de la client, distribuindu-le eficient către microserviciile relevante, reducând complexitatea și izolând clientul de implementarea internă a microserviciilor.

A diagram of a company

Description automatically generated

Figura 5.2 Arhitectura aplicației SportPlanner

Dupa cum se poate observa arhitectura aplicației este formata din trei nivele principale:

* **Nivelul Client (Frontend):** SportPlanner\_Frontend reprezintă interfața cu utilizatorul, oferind o experiență interactivă și intuitivă pentru planificarea activităților sportive. Aceasta comunică cu backend-ul prin intermediul API Gateway-ului, utilizând tehnologii precum HTML, CSS și JavaScript pentru a
* **Nivelul de Servicii (Backend):**
* **API Gateway:** Acționează ca un punct central de intrare pentru toate cererile din frontend, gestionând autentificarea, autorizarea, rutarea și agregarea datelor de la diferite microservicii.
* **Auth Service:** Responsabil pentru autentificarea și autorizarea utilizatorilor, gestionând înregistrarea, conectarea și controlul accesului la resursele aplicației.
* **SportPlanner Service:** Nucleul aplicației, gestionând planificarea activităților sportive, inclusiv crearea de evenimente, căutarea de utilizatori de fotbal și gestionarea meciurilor sau a profilului.
* **Notification Service:** Responsabil pentru trimiterea de notificări către utilizatori, cum ar fi invitatia la un meci, actualizări de stare ale invitațiilor trimise la alți utilizatori.
* **Nivelul de Baze de Date**
* **Auth DB:** Stochează informații despre utilizatori, cum ar fi numele de utilizator, parolele și rolurile.
* **SportPlanner DB:** Stochează date despre meciuri, cereri de participare la meciuri, locații, date ale utilizatorilor și alte informații relevante pentru planificarea activităților.

## Nivel baze de date

Într-o arhitectură de microservicii, gestionarea optimă a bazelor de date este esențială pentru a asigura performanță și scalabilitate. În proiectul nostru, am adoptat o strategie în care fiecare din cele două microservicii are propria bază de date dedicată. Acest design asigură izolarea și autonomia fiecărui microserviciu, simplificând dezvoltarea și mentenanța întregului sistem.

Concret, aplicația noastră utilizează două baze de date separate:

* bază de date pentru microserviciul sportplanner, care gestionează aspectele funcționale principale ale aplicației, cum ar fi datele despre evenimente sportive, utilizatori și locații.
* bază de date distinctă pentru microserviciul de autentificare, responsabil de gestionarea utilizatorilor, autentificare, autorizare și securitate.

### Baza de date SportPlanner

Diagrama bazei de date (figura 5.3) care gestioneaza aspectele principale ale aplicației:

A screenshot of a computer

Description automatically generated

Figura 5.3 Structura bazei de date SportPlanner

Fiecare tabel din baza de date a fost construit pentru a gestiona eficient informațiile specifice necesare pentru funcționarea aplicației de organizare a meciurilor de fotbal. Fiecare tabel are scopuri și funcționalități bine definite, dupa cum urmeaza:

* **Users:** conține detalii despre utilizatori, cum ar fi identificatorul unic pentru fiecare utilizator, cheie primară, vârsta, data nașterii, țara de origine, prenumele, numele de familie, piciorul preferat (stâng sau drept), înălțimea, locația curentă, naționalitatea, numărul de telefon, fotografia, poziția preferată pe teren (ex: portar, atacant), numele de utilizator și greutatea. Această tabelă este esențială pentru gestionarea informațiilor personale, de profil și de contact ale fiecarui utilizatorilor aplicației SportPlanner.
* **Matches:** conține detalii despre meciurile de fotbal organizate prin intermediul aplicației, incluzând identificatorul unic pentru fiecare meci, cheie primară, data meciului, detalii suplimentare despre meci, ora de încheiere, nivelul de dificultate al meciului (ex: amator, semi-profesionist), numărul de jucători per echipă, prețul pentru participare, ora de începere, numărul de echipe participante, titlul meciului, identificatorul locației unde are loc meciul, cheie străină, și identificatorul utilizatorului care a creat meciul, cheie străină. Această tabelă este centrală pentru gestionarea evenimentelor sportive organizate în cadrul aplicației.
* **Locations:** stochează informații despre locațiile de joc disponibile în aplicație, având coloane precum identificatorul unic pentru fiecare locație, cheie primară, adresa locației, o descriere detaliată, latitudinea și longitudinea (pentru GPS), numele locației și fotografia acesteia. Această tabelă este utilizată pentru a gestiona locațiile unde se desfășoară meciurile de fotbal.
* **UserMatches:** reprezintă tabelul de legatură dintre tabelele Users si Matches care au o relație “many-to-many” și asociază utilizatorii cu meciurile la care participă, prin identificatorul utilizatorului și identificatorul meciului, ambele fiind chei străine. Această tabelă este esențială pentru urmărirea participării utilizatorilor la meciuri.
* **Request:** gestionează solicitările de participare la meciuri, incluzând identificatorul unic pentru fiecare solicitare, cheie primară, identificatorul meciului la care se face solicitarea (cheie străină), tipul solicitării (1 - invitație, 2 - cerere de participare), statusul solicitării (1 - acceptată, 2 - refuzată, 0 - în așteptare), identificatorul utilizatorului care primește solicitarea și identificatorul utilizatorului care trimite solicitarea, ambele fiind chei străine. Această tabelă este crucială pentru gestionarea cererilor și invitațiilor de participare la meciuri.

### Baza de date SportPlanner\_Auth

Diagrama bazei de date (figura 5.4) care gestioneaza detaliile principale ale utilizatorilor:

A screenshot of a computer screen

Description automatically generated

Figura 5.4 Structura bazei de date SportPlanner\_Auth

Baza de date “SportPlanner\_Auth” a aplicației include și tabele dedicate gestionării informațiilor utilizatorilor și procesului de recuperare a parolelor.

* **User** stochează detalii esențiale despre utilizatori, având următoarele coloane: id care reprezintă cheia primară a tabelului și asigură identificatorul unic pentru fiecare utilizator, email pentru adresa de e-mail a utilizatorului, care este unică, password pentru parola criptată a utilizatorului și username care este unic.
* **PasswordRecovery** gestionează solicitările de recuperare a parolei, cu următoarele coloane: id, cheia primară a tabelului și identificator unic pentru fiecare solicitare de recuperare a parolei, email, adresa de e-mail asociată solicitării de recuperare, și uuid, un identificator unic universal (UUID) generat pentru fiecare solicitare, care conține token-ul de autorizare pentru ca utilizatorul să-l poata utiliza în link-ul de resetare a parolei.

Pentru managementul utilizatorilor și pentru securitatea procesului de recuperare a parolelor în cadrul aplicației, asigurând o administrare eficientă și securizată a datelor personale și a credentialelor utilizatorilor.

## Nivel de server

Nivelul de server al aplicației SportPlanner este organizat într-o arhitectură de microservicii, care permite o gestionare eficientă și modulară a diferitelor funcționalități ale aplicației. Această arhitectură este detaliată prin trei microservicii principale: Auth Service, SportPlanner Service și Notification Service. Aceste microservicii sunt dezvoltate folosind Spring Boot, un framework care facilitează crearea de aplicații enterprise în Java.

### SportPlanner Service

SportPlanner Service este microserviciul central și cel mai important al aplicației SportPlanner, deoarece conține logica principală de business. Acest microserviciu gestionează toate aspectele legate de organizarea și administrarea meciurilor de fotbal, fiind inima aplicației. Detaliile legate de utilizatori, meciuri, cereri și locații sunt toate administrate prin acest serviciu, asigurând funcționalitatea esențială a platformei. În plus față de funcționalitățile sale principale, microserviciul SportPlanner orchestrează și interacțiunea cu celelalte două microservicii esențiale, Auth Service și Notification Service. Această colaborare asigură funcționarea optimă a aplicației și permite implementarea completă a tuturor funcționalităților, de la autentificarea utilizatorilor până la trimiterea de notificări relevante.

Fiind nucleul aplicației, microserviciul SportPlanner înglobează o gamă largă de funcționalități, de la cele elementare până la cele mai complexe, care implică o logică elaborată sau necesită interacțiunea cu alte microservicii din cadrul aplicației. Principalele funcționalități ale microserviciului sunt:

* **Gestionarea Utilizatorilor:**
* Crearea și actualizarea profilurilor utilizatorilor.
* Administrarea detaliilor personale și de contact.
* Vizualizarea și gestionarea preferințelor utilizatorilor (ex: poziția preferată pe teren, piciorul preferat).
* **Gestionarea Meciurilor:**
* Crearea, actualizarea și ștergerea meciurilor.
* Administrarea detaliilor meciului, cum ar fi data, ora, locația, numărul de echipe și jucători, nivelul de dificultate și vizibilitatea (public/privat).
* **Gestionarea înscrierilor și participării la meciuri:** 
  + Utilizatorii pot solicita să se înscrie la un meci, iar organizatorul poate accepta sau respinge aceste cereri.
* Organizatorii pot invita utilizatori specifici la meciuri, iar aceștia pot accepta sau declina invitațiile.
* **Gestionarea Locațiilor:**
* Adăugarea și actualizarea detaliilor locațiilor unde se vor desfășura meciurile.
* Administrarea adresei, descrierii, coordonatelor GPS și fotografiilor locațiilor.
* **Interacțiunea cu alte Microservicii:**
* Comunicarea cu Auth Service pentru autentificarea și autorizarea utilizatorilor.
* Integrarea cu Notification Service pentru trimiterea de notificări privind confirmările de participare, modificările de program și alte actualizări.

Arhitectura internă a microserviciului SportPlanner este proiectată în straturi distincte, asigurând o separare clară a responsabilităților și facilitând scalabilitatea și mentenabilitatea. Pentru a simplifica interacțiunile dintre aceste straturi și a ascunde complexitatea logicii de business, am implementat modelul de design Façade. Această abordare oferă o interfață unificată și ușor de utilizat pentru accesarea funcționalităților microserviciului, îmbunătățind lizibilitatea și mentenabilitatea codului. Fațada din SportPlanner Service aduce multiple beneficii, precum simplificarea interacțiunilor cu controller-ul prin oferirea unei interfețe concise și coerente, separarea preocupărilor prin decuplarea controller-ului de logica detaliată a serviciilor, permițând modificări independente ale fiecărui strat, și modularitate, prin care schimbările în implementarea serviciilor pot fi făcute fără a necesita modificări în controller, ceea ce sporește flexibilitatea și adaptabilitatea sistemului. Prin utilizarea Façade Design Pattern, am reușit să creăm o arhitectură mai curată, mai ușor de întreținut și mai scalabilă pentru microserviciul SportPlanner.

A computer screen shot of a diagram

Description automatically generated

Figura 5.5 Structura interna a arhitecturii microserviciului SportPlanner

* **Controllers**:
* Clasele controller expun API-urile REST ale microserviciului, gestionând cererile HTTP și returnând răspunsurile adecvate.
* **Services**:
* Conține logica de business principală. Aici sunt implementate toate regulile și procesele de afaceri ale aplicației.
* Service-urile interacționează cu repository-urile pentru a efectua operațiuni CRUD și a implementa logica necesară.
* **Repositories**:
* Interfețele JPA pentru accesul la baza de date, gestionând operațiunile de citire, scriere, actualizare și ștergere.
* **Dtos**
* Obiectele DTO sunt utilizate pentru a transfera date între diferite straturi ale aplicației, asigurând o separare clară a preocupărilor.
* **Entities**:
* Clasele care reprezintă tabelele din baza de date, mapate prin JPA.
* Fiecare entitate este o reprezentare a unui obiect de business, cum ar fi User..
* **Exceptions:**
* Gestionarea excepțiilor pentru a asigura un răspuns informativ la erori.
* Handler-e personalizate pentru a trata diferite tipuri de erori și a oferi feedback util utilizatorilor și dezvoltatorilor.
* **Facade**:
* Acționează ca o interfață simplificată pentru alte componente ale sistemului, agregând apeluri către mai multe servicii sau oferind o logică de business mai complexă.
* Utilizarea pattern-ului Facade ajută la reducerea complexității și la îmbunătățirea ușurinței de utilizare a microserviciilor.
* **Clients**:
* Gestionarea comunicării cu alte microservicii sau API-uri externe.
* Exemple includ clienții HTTP pentru interacțiunea cu Auth Service și Notification Service.

Fluxul de lucru în cadrul microserviciului SportPlanner și interacțiunile sale cu alte componente ale aplicației:

1. **Recepționarea Cererilor:**

* SportPlanner\_Frontend, interfața cu utilizatorul, trimite cereri (de exemplu, pentru a crea un meci) prin intermediul API Gateway-ului.
* API Gateway-ul acționează ca un punct central de acces, gestionând rutarea cererilor către microserviciul potrivit, în acest caz, SportPlanner Service.

1. **Procesarea Logicii de Business:**

* Odată ce cererea ajunge la SportPlanner Service, acesta își folosește logica de business pentru a o procesa. Aceasta poate implica validarea datelor primite, aplicarea logicii specifice și pregătirea datelor pentru a fi utilizate în continuare.

1. **Operații asupra Bazei de Date:**

* Dacă cererea necesită interacțiunea cu baza de date (de exemplu, pentru a salva un nou eveniment, a actualiza informații despre un utilizator sau a prelua lista de meciuri disponibile), SportPlanner Service folosește repository-uri JPA.

1. **Comunicare Între Servicii:**

* În timpul procesării cererii, microserviciul poate avea nevoie să comunice cu alte microservicii:
* Auth Service: Pentru a verifica dacă utilizatorul este autentificat și are permisiunea de a efectua acțiunea solicitată.
* Notification Service: Pentru a trimite notificări către utilizatori (de exemplu, pentru a confirma înscrierea la un meci).

A diagram of a computer

Description automatically generated

Figura 5.6 Flux-ul unei cereri din partea utilizatorului

In figura 5.6, este reprezentat fluxul detaliat pentru un request care ajunge la microserviciul sportPlanner:

1. Actorul inițiază o cerere către aplicație prin intermediul clientului frontend (SportPlanner\_Frontend). Acesta poate fi o cerere pentru a crea un nou meci, a vizualiza detaliile unui meci existent, a trimite o invitație sau altă operațiune specifică.
2. **Cererea este procesata de către controller**. După ce ajunge la unul dintre endpoint-urile expuse de controller , acesta este responsabil de maparea cererii către metoda corespunzătoare. Controllerul apelează metoda corespunzătoare din clasa fațadă.
3. **Apelul metodei corespunzătoare de catre fațadă**. Fațada orchestrează apelurile către unul sau mai multe servicii din pachetul “service”. Aceasta poate implica apeluri secvențiale sau paralele către diferite servicii pentru a colecta toate datele necesare sau pentru a efectua diverse acțiuni necesare pentru a îndeplini cererea.
4. **Interacțiunea serviciilor cu repository-ul**. Serviciile interacționează cu repository-urile din pachetul ”repositories” pentru a prelua sau modifica datele din baza de date. Repository-urile sunt responsabile de operațiunile CRUD și utilizează entitățile sau dto-urile pentru a mapara datele din baza de date în obiecte Java.
5. Serviciile returnează rezultatele operațiunilor lor către fațadă. Fațada primește rezultatele de la servicii, le procesează dacă este necesar și le returnează controllerului. Controllerul trimite răspunsul înapoi către actor. Utilizatorul primește astfel feedback-ul sau datele solicitate prin intermediul clientului frontend.

Diagrama prezentată in FIGURA..., ilustrează un caz de utilizare pentru un utilizator care încearcă să se alăture unui meci sportiv într-o aplicație.

**Flux principal:**

* + 1. Start: Utilizatorul deschide aplicația SportPlanner.

2. Autentificare user: Utilizatorul introduce datele de autentificare (nume de utilizator și parolă) pentru a se conecta la contul său.

2.1 Succes: Autentificarea reușește, iar utilizatorul este direcționat către pagina principală cu lista de meciuri.

2.2 Eșec (Flux alternativ 2.2.1): Autentificarea eșuează, iar utilizatorul primește un mesaj de eroare și posibilitatea de a reîncerca.

3. Încărcare vizualizare meciuri: Aplicația afișează o listă cu meciurile disponibile.

3.1 Succes**:** Meciurile sunt încărcate cu succes, iar utilizatorul poate vedea detaliile acestora.

3.2 Eșec (Alternative Flow 1.): Încărcarea meciurilor eșuează, iar utilizatorul primește un mesaj de eroare și opțiunea de a reîncărca pagina.

4. Încărcare vizualizare detalii meci: Utilizatorul selectează un meci și aplicația încarcă detaliile acestuia (dată, oră, locație, participanți etc.).

4.1 Succes: Detaliile meciului sunt încărcate cu succes.

4.2 Eșec: Utilizatorul primește un mesaj de eroare și opțiunea de a reîncărca pagina.

5. Trimite cerere de participare la meci: Utilizatorul decide să se alăture meciului și trimite o cerere de participare.

5.1 Succes: Cererea este trimisă cu succes, iar aplicația trimite o notificare către organizatorul meciului.

5.2 Eșec (Alternative Flow 2.): Trimiterea cererii eșuează, iar utilizatorul primește un mesaj de eroare și opțiunea de a reîncerca.

6. Trimite notificare către organizator**:** Aplicația notifică organizatorul meciului despre noua cerere de participare.

6.1 Succes: Notificarea este trimisă cu succes.

6.2 Eșec: Se afișeaza un mesaj de eroare.

7. Continuă? Utilizatorul alege să continue navigarea în aplicație sau să se delogheze.

7.1 Da: Utilizatorul continuă să utilizeze aplicația.

7.2 Nu: Utilizatorul se deloghează.

8. Delogare user: Utilizatorul se deconectează de la aplicație.

9. Stop: Aplicația se închide sau utilizatorul navighează către o altă secțiune.

**Fluxuri alternative (Figura 5.7):**

1. Eșec la autentificarea userului: Autentificarea eșuează, sistemul îi solicită să verifice datele de autentificare și să încerce din nou.

2. Eșec la trimiterea de join request sau la încărcarea meciurilor/detaliilor meciului: Utilizatorul poate repeta actiunea pana cand va avea succes

A diagram of a process

Description automatically generated with medium confidence

Figura 5.7 Diagrama de activitate pentru trimiterea unei cereri de participare la meci

În proiectul SportPlanner, mecanismul de tratare a excepțiilor este implementat prin clasa GlobalControllerAdvice, prin intermediul anotării @ControllerAdvice oferită de framework-ul Spring. Aceasta reprezintă o abordare centralizată și elegantă pentru gestionarea excepțiilor la nivelul aplicației, oferind mai multe avantaje față de tratarea clasică a excepțiilor.

Avantajele utilizării @ControllerAdvice în gestionarea excepțiilor sunt multiple și semnificative. În primul rând, centralizarea excepțiilor într-un singur loc, cum este clasa GlobalControllerAdvice (figura 5.8), simplifică mentenabilitatea și îmbunătățește lizibilitatea codului. Astfel, fiecare controller nu mai trebuie să gestioneze individual excepțiile, eliminând redundanțele și facilitând un comportament uniform în toată aplicația. În al doilea rând, reutilizarea metodelor de gestionare a excepțiilor pentru mai multe controllere elimină duplicarea codului și asigură consistență. Această abordare permite separarea clară a logicii de business de gestionarea erorilor, menținând logica de business curată și concentrată pe funcționalitatea principală. În plus, extensibilitatea oferită de @ControllerAdvice facilitează adăugarea de noi tipuri de excepții și gestionarea lor într-un mod simplu și direct, doar prin adăugarea unei noi metode anotate cu @ExceptionHandler în clasa GlobalControllerAdvice. Aceste avantaje contribuie la crearea unei aplicații robuste, ușor de întreținut și extensibilă.

A computer screen shot of a program code

Description automatically generated

Figura 5.8 Clasa GlobalControllerAdvice

În fluxul de gestionare a excepțiilor, să considerăm exemplul procesului de login în care se aruncă o excepție de tip UserNotFoundException:

* Aruncarea Excepției: În timpul procesului de login, dacă utilizatorul nu este găsit în baza de date, serviciul de autentificare aruncă o excepție de tip UserNotFoundException.
* Capturarea Excepției de către Controller Advice: Excepția aruncată este interceptată de mecanismul @ControllerAdvice. Metoda handleNotFoundException din clasa GlobalControllerAdvice este invocată automat pentru a gestiona această excepție.
* Generarea și Trimiterea Răspunsului: Metoda handleNotFoundException generează un răspuns ResponseEntity cu un mesaj de eroare specific, de exemplu, "User not found", și un cod de status HTTP 400 Bad Request. Acest răspuns este trimis înapoi către clientul care a inițiat cererea.
* Recepționarea Răspunsului de către Client: Clientul (frontend-ul) primește răspunsul și afișează mesajul de eroare utilizatorului, oferindu-i astfel informații clare despre problema apărută.

Acest flux permite o gestionare centralizată și consistentă a erorilor, oferind utilizatorilor feedback util și detaliat în caz de probleme.

### Auth Service

Auth-Service este un component esențial al aplicației, responsabil pentru gestionarea autentificării și autorizării utilizatorilor. Acesta include funcționalități precum înregistrarea utilizatorilor, autentificarea, generarea și gestionarea tokenurilor JWT pentru sesiuni sigure, precum și procesele de recuperare a parolei. Auth-Service asigură securitatea și integritatea datelor utilizatorilor prin metode avansate de criptare și gestionare a tokenurilor de acces și refresh.

Arhitectura internă a microserviciului Auth-Service este similară cu cea a microserviciului SportPlanner, având o structură pe pachete bine definită, incluzând controllere, servicii, repository-uri și entități. Design-ul folosit este Facade Pattern pentru a simplifica interacțiunea cu subsistemele și pentru a organiza logică de business într-un mod eficient și modular.

**Funcționalități cheie ale Auth-Service**

1. **Înregistrare și autentificare**

* Permite utilizatorilor să se înregistreze în aplicație prin endpoint-ul /signup, unde parolele sunt criptate înainte de a fi stocate în baza de date.
* Autentificarea utilizatorilor se face prin endpoint-ul /login, unde utilizatorii primesc tokenuri JWT pentru acces și refresh.
* Parolele utilizatorilor sunt criptate folosind algoritmi de criptare avansați, asigurând protecția acestora în baza de date.

1. **Gestionarea tokenurilor JWT**

* La autentificare, utilizatorii primesc un token de acces și un token de refresh. Tokenurile JWT sunt utilizate pentru a asigura sesiuni sigure și autentificate.
* Refresh tokenurile sunt stocate în cookie-uri HTTP-only pentru a preveni atacurile de tip XSS.
* Endpoint-ul /refresh-token permite utilizatorilor să obțină un nou token de acces folosind un token de refresh valid.

1. **Recuperarea parolei**

* Endpoint-ul /forgot-password/{email} inițiază procesul de recuperare a parolei prin trimiterea unui email cu un link de resetare.
* Endpoint-ul /reset-password/{token} permite utilizatorilor să reseteze parola folosind tokenul primit pe email.

A diagram of a computer program

Description automatically generated with medium confidence

Figura 5.9 Diagrama de secvență pentru operația de recuperare a parolei

Diagrama de secvență din figura 5.9 ilustrează fluxul de acțiuni implicate în procesul de trimitere a unui e-mail de recuperare a parolei, atunci când utilizatorul își uită parola în cadrul aplicației.

Când un utilizator inițiază o cerere de recuperare a parolei, trimite o solicitare POST către endpoint-ul /forgot-password/{email}:

* Controllerul primește cererea și extrage adresa de e-mail și URL-ul de origine din header.
* Apelează metoda forgotPassword din fațada UserFacade.

Metoda forgotPassword din fațadă orchestrează procesul de recuperare a parolei:

* Verifică dacă e-mailul există folosind serviciul UserService.
* Generează un nou obiect PasswordRecovery folosind PasswordRecoveryService.
* Trimite un e-mail de recuperare a parolei utilizând MailService.

Metoda addPasswordRecovery din PasswordRecoveryService creează un token unic pentru recuperarea parolei și îl stochează în baza de date:

* reează un obiect PasswordRecovery.
* Generează un UUID unic.
* Salvează obiectul în baza de date.

Metoda sendForgotMessage trimite un e-mail către utilizator cu un link pentru resetarea parolei:

* Construiește un mesaj de e-mail care conține un link pentru resetarea parolei.
* Trimite e-mailul folosind JavaMailSender.

A screenshot of a computer

Description automatically generatedFigura 5.10 Model de email primit de utilizator

Figura 5.10 ilustrează conținutul și funcționalitatea e-mailului de resetare a parolei trimis utilizatorului în cadrul aplicației. Acest e-mail conține un link esențial, care redirecționează utilizatorul către o pagină dedicată în cadrul aplicației, unde acesta poate introduce și confirma noua parolă. Această implementare reprezintă o facilitate valoroasă, venind în ajutorul utilizatorilor care și-au uitat parola și doresc să își recupereze accesul la cont într-un mod simplu și sigur.

În microserviciul Auth-Service, securitatea parolelor este asigurată prin utilizarea unui mecanism robust de criptare a acestora. Criptarea parolelor joacă un rol crucial în protejarea datelor utilizatorilor și în prevenirea accesului neautorizat la conturi.

Algoritmul BCrypt este preferat pentru criptarea parolelor datorită următoarelor caracteristici:

* **Securitate:** BCrypt utilizează un algoritm de hashing adaptativ, ceea ce înseamnă că timpul necesar pentru a genera hash-ul poate fi crescut pe măsură ce puterea de calcul crește. Astfel, BCrypt rămâne sigur chiar și în fața atacurilor moderne cu forță brută.
* **Salt:** Fiecare hash generat de BCrypt include un salt unic, ceea ce înseamnă că două parole identice nu vor avea niciodată același hash. Acest lucru previne atacurile de tip rainbow table.

Parolele sunt criptate folosind algoritmul BCrypt, care este configurat printr-un bean PasswordEncoder în clasa ApplicationConfig, figura 5.11.

A screen shot of a computer

Description automatically generated

Figura 5.11 Clasa ApplicationConfig

Parolele utilizatorilor sunt criptate în momentul în care aceștia se înregistrează, prin intermediul metodei createUser din serviciul UserService. Parola utilizatorului este criptată folosind metoda encoder.encode(signupRequest.getPassword()). encoder este un bean de tip PasswordEncoder care utilizează Bcrypt, ca in figura 5.12.

A screen shot of a computer code

Description automatically generated

Figura 5.12 Encriptarea parolei

Criptarea parolelor în cadrul microserviciului Auth-Service este o măsură fundamentală de securitate, protejând datele sensibile ale utilizatorilor și asigurând conformitatea cu standardele de securitate în vigoare. Prin utilizarea algoritmului BCrypt, un algoritm robust și rezistent la atacuri, chiar și în cazul unei breșe de securitate a bazei de date, parolele utilizatorilor rămân protejate. Această practică este esențială pentru a preveni accesul neautorizat și a proteja împotriva atacurilor de tip brute-force, asigurând astfel un nivel înalt de securitate și încredere pentru utilizatorii platformei.



### Notification Service

Notification Service are rolul esențial de a trimite notificări către utilizatori, informându-i despre evenimente importante precum confirmările de participare la meciuri, modificările de program și alte notificări relevante. Acest microserviciu se asigură că utilizatorii sunt mereu la curent cu toate schimbările importante din cadrul aplicației SportPlanner.

Arhitectura internă a microserviciului Notification Service este similară cu cea a celorlalte microservicii din cadrul aplicației, precum SportPlanner și Auth-Service. Structura sa include pachete organizate pentru controlere, servicii și fațade, și utilizează design pattern-ul Fațadă (Facade) pentru a organiza și simplifica logica de business.

Un aspect distinct al Notification Service este faptul că acesta nu operează cu o bază de date. Spre deosebire de alte microservicii care trebuie să stocheze și să gestioneze date.

Notification Service funcționează ca un serviciu tranzacțional care procesează și livrează notificări în timp real, fără a necesita stocarea acestor date.

Pentru trimiterea notificărilor în timp real, Notification Service utilizează WebSockets. WebSockets oferă o conexiune persistentă între client și server, permițând serverului să trimită notificări către clienți imediat ce apar evenimentele relevante. Acest mecanism asigură o comunicare rapidă și eficientă, menținând utilizatorii la curent cu toate modificările și evenimentele importante.

Pentru a permite trimiterea notificărilor în timp real prin WebSockets, aplicația SportPlanner utilizează o configurare specifică pentru a stabili și gestiona conexiunile WebSocket. Configurarea WebSocket se realizează printr-o clasă anotată cu @Configuration și @EnableWebSocketMessageBroker. Aceasta definește punctele finale WebSocket și configurează broker-ul de mesaje:

* **Punctul Final WebSocket:** Metoda registerStompEndpoints adaugă un endpoint /ws care permite conexiuni WebSocket de la orice origine (setAllowedOriginPatterns("\*")). Acest endpoint suportă fallback cu SockJS pentru compatibilitate cu browsere care nu suportă WebSockets nativ.
* **Configurarea Broker-ului de Mesaje:** Metoda configureMessageBroker configurează un broker simplu de mesaje care permite trimiterea de mesaje către destinațiile /topic și /user. Prefixul /app este setat pentru destinațiile aplicației.

Pentru a gestiona evenimentele legate de conexiunile WebSocket, cum ar fi deconectarea unui utilizator, este utilizată o componentă specifică.

* Componenta de Ascultare: WebSocketEventListener este o componentă Spring (@Component) care ascultă evenimentele de deconectare WebSocket (SessionDisconnectEvent).
* Gestionarea Deconectării: Metoda handleWebSocketDisconnectListener preia detaliile despre sesiunea deconectată, inclusiv numele de utilizator, și înregistrează deconectarea în log.

Exista mai multe avantaje pentru utilizarea websocket-urilor pentru transmiterea de mesaje, precum:

* **Actualizări în Timp Real:** WebSockets permit trimiterea instantanee a notificărilor, asigurându-se că utilizatorii sunt informați imediat despre orice schimbare.
* **Eficiență:** Spre deosebire de solicitările HTTP tradiționale, WebSockets mențin o conexiune deschisă, reducând astfel latența și consumul de resurse.
* **Interactivitate Sporită:** Notificările în timp real îmbunătățesc experiența utilizatorului, oferindu-i un sentiment de interactivitate și implicare continuă în aplicație.

Procesul de trimitere a notificărilor în cadrul microserviciului Notification Service este gestionat printr-o combinație de metode REST și WebSockets. Iată o descriere detaliată a fluxului de trimitere a notificărilor:

1. Primul pas în fluxul de trimitere a notificărilor este primirea unei cereri de trimitere a notificării printr-un endpoint REST.

* Utilizatorul trimite o cerere către endpoint-ul /sendNotification, cu un NotificationDto care are numele utilizatorului destinatar mesajul notificării.
* Endpoint-ul este gestionat de metoda sendNotification din NotificationController, care preia NotificationDto și apelează metoda corespunzătoare din NotificationFacade, care apeleaza NotificationService.

1. Odată ce cererea ajunge la NotificationService, notificarea este procesată și trimisă utilizând WebSockets (figura 5.13).

* NotificationService utilizează SimpMessagingTemplate pentru a trimite mesaje prin WebSockets.
* Metoda sendNotification creează un obiect Message care conține numele destinatarului și conținutul mesajului.
* SimpMessagingTemplate trimite mesajul către utilizatorul specificat, folosind metoda convertAndSendToUser. Mesajul este trimis la destinația /notifications, unde utilizatorul destinat este ascultător.

A screen shot of a computer program

Description automatically generated Figura 5.13 Trimiterea unei notificări

1. Utilizatorul destinatar primește notificarea în timp real și este informat despre evenimentul relevant:

* Utilizatorii aplicației sunt conectați la WebSockets și ascultă mesajele pe canalul /notifications.
* Când un mesaj este trimis pe acest canal, utilizatorii relevanți îl recepționează în timp real.

Prin utilizarea WebSockets și menținerea unei arhitecturi similare cu celelalte microservicii, Notification Service oferă un sistem robust și eficient pentru gestionarea notificărilor, contribuind astfel la îmbunătățirea experienței generale a utilizatorilor în cadrul aplicației SportPlanner.

## Nivel de client

Microserviciul de frontend al aplicației SportPlanner este responsabil pentru interfața utilizatorului (UI) și interacțiunea acestuia cu backend-ul aplicației. Acesta este construit folosind framework-ul Vue.js, oferind o experiență de utilizare modernă și fluidă.

Structura proiectului frontend este organizată în mai multe directoare, fiecare având un rol specific. Iată principalele directoare:

* config: Conține configurări pentru diverse servicii, inclusiv Axios.js pentru efectuarea cererilor HTTP și websockets.js pentru gestionarea conexiunilor WebSocket.
* interceptors: Găzduiește interceptori pentru Axios, cum ar fi axios.ts, care sunt utilizați pentru a intercepta și modifica cererile și răspunsurile HTTP.
* pinia: Include configurările pentru gestionarea stării aplicației folosind Pinia, o bibliotecă modernă de gestionare a stării pentru Vue.js.
* router: Conține configurarea rutelor aplicației, asigurând navigarea între diferitele pagini și componente.
* stores: Include modulele de gestionare a stării aplicației, cum ar fi counter.ts, folosind Pinia.
* views: Găzduiește componentele Vue care reprezintă pagini întregi sau vederi ale aplicației.

#### Principalele tehnologii si biblioteci utilizate, sunt:

* **Axios**: Folosit pentru efectuarea cererilor HTTP către backend. Axios este configurat în fișierul Axios.js din directorul config și este utilizat pentru a comunica eficient cu API-urile backend-ului.
* **Pinia**: Utilizat pentru gestionarea stării aplicației. Configurarea și utilizarea Pinia se regăsesc în fișierele din directorul pinia.
* **Bootstrap**: Folosit pentru stilizarea componentelor și paginilor aplicației, asigurând o interfață de utilizator atrăgătoare și responsivă.

*A computer screen shot of a program code

Description automatically generated*

Figura 5.14 Metoda de login din frontend

Metoda loginRequest, din figura 5.14 este utilizată pentru autentificarea utilizatorilor în aplicație și realizează următoarele acțiuni:

1. **Trimiterea Cererii de Autentificare**:
   * Folosește Axios pentru a trimite o cerere POST către endpoint-ul /user/login cu emailul și parola utilizatorului. Opțiunea withCredentials: true este setată pentru a include cookie-urile necesare în cerere.
2. **Procesarea Răspunsului**: În cazul unui răspuns de succes, metoda:
   * + Stochează starea de autentificare și tokenul de acces în Pinia.
     + Stochează numele utilizatorului autentificat.
     + Setează antetul de autorizare pentru cererile ulterioare cu tokenul de acces.
     + Realizează conexiunea la WebSocket folosind numele utilizatorului.
     + Redirecționează utilizatorul către pagina de explorare (explore).
3. **Gestionarea Erorilor**:
   * În cazul unei erori, metoda captează și afișează mesajul de eroare.

Astfel, microserviciul de frontend al aplicației SportPlanner este bine structurat și utilizează tehnologii moderne pentru a asigura o interfață de utilizare performantă și intuitivă, facilitând comunicarea eficientă cu backend-ul și gestionarea notificărilor în timp real.

# Testare și validare

Testarea și validarea reprezintă etape critice în dezvoltarea aplicației SportPlanner, asigurând că funcționalitatea implementată este corectă și conformă cu cerințele specificate. În acest capitol, vom detalia metodele și tehnicile utilizate pentru testarea și validarea aplicației, incluzând testarea automată cu JUnit și testarea manuală.

Testarea automată a fost realizată utilizând framework-ul JUnit pentru a acoperi cazurile principale ale aplicației. Prin intermediul testelor unitare, am verificat corectitudinea funcționalităților implementate, asigurându-ne că fiecare componentă funcționează așa cum a fost proiectată. Testele unitare sunt esențiale pentru detectarea timpurie a defectelor și pentru facilitarea mentenanței și evoluției aplicației.

Un exemplu de test unitar este cel pentru metoda getMatch, FIGURA din cadrul MatchService. Acest test verifică dacă metoda returnează corect un meci în funcție de ID-ul său.

A screen shot of a computer program

Description automatically generated

Figura 6.1 Model de Junit test

În acest test, am creat un obiect Match de test, am configurat mock-ul matchRepository pentru a returna acest obiect atunci când este apelat cu ID-ul 1 și am verificat dacă metoda getMatch returnează meciul așteptat. De asemenea, am verificat că metoda findById din matchRepository a fost apelată o singură dată.

A screenshot of a computer

Description automatically generated

Figura 6.2 Toate serviciile pentru care s-au realizat teste

Figura 6.2 ilustrează clasele principale care au fost supuse testării pentru a evalua funcționarea corectă a operațiilor esențiale din cadrul aplicației.

Pe lângă testele unitare automate, am efectuat și testare manuală pentru a valida funcționalitatea aplicației din perspectiva utilizatorului final. Testarea manuală implică verificarea vizuală și funcțională a interfeței utilizatorului, navigarea prin diferitele secțiuni ale aplicației și executarea scenariilor de utilizare pentru a identifica problemele care nu pot fi capturate prin testele unitare automate. Testarea manuală a inclus:

* Verificarea autentificării și înregistrării utilizatorilor.
* Testarea navigării între paginile aplicației.
* Verificarea corectitudinii datelor afișate în diverse secțiuni (de exemplu, detaliile meciurilor, notificările).
* Validarea notificărilor în timp real prin intermediul WebSocket.

Combinând testarea automată cu JUnit și testarea manuală, am asigurat o acoperire completă a funcționalităților aplicației SportPlanner. Testele unitare au oferit o bază solidă pentru validarea corectitudinii codului la nivel de componentă, în timp ce testarea manuală a permis verificarea funcționalității complete și a experienței utilizatorului. Acest proces dual de testare și validare ne-a permis să livrăm o aplicație robustă și fiabilă.

# Manual de instalare si utilizare

Acest capitol prezintă resursele hardware și software necesare pentru rularea aplicației Sport Planner, precum și pașii detaliați pentru instalarea și configurarea acesteia.

## Resurse hardware

Pentru a rula aplicația Sport Planner, avem nevoie de următoarele resurse hardware:

* **Calculator:** Un desktop sau laptop cu sistem de operare Windows, macOS sau Linux. Este necesar să dispuneți de cel puțin 8 GB de memorie RAM și 256 GB de spațiu de stocare liber (ideal pe SSD).
* **Conexiune la internet**: O conexiune stabilă la internet este esențială pentru testarea și rularea aplicației în mediul de producție.

## Resurse software

Pe langa resursele hardware, utilizarea aplicatiei necesita și niște resurse software destul de importante, precum:

* Sistem de operare: Windows, macOS sau Linux
* IntelliJ IDEA Ultimate Edition: Pentru dezvoltarea backend-ului Java cu Spring Boot. Versiunea Ultimate oferă suport extins pentru framework-uri și instrumente Java, inclusiv Spring Boot și Hibernate.
* Visual Studio Code: Pentru dezvoltarea frontend-ului Vue.js. Este un editor de cod ușor și versatil, cu o gamă largă de extensii pentru Vue.js.
* JDK (Java Development Kit): Versiunea 17 sau o versiune ulterioară compatibilă cu Spring Boot.
* Server de baze de date MySQL: Versiunea 8.0 sau o versiune ulterioară pentru a stoca și gestiona datele aplicației.
* Browser web modern: Google Chrome, Mozilla Firefox sau Microsoft Edge pentru testarea și utilizarea aplicației.

## Instalarea și configurarea mediului de dezvoltare

Pentru a începe dezvoltarea și rularea aplicației Sport Planner, este necesară instalarea și configurarea corespunzătoare a următoarelor componente software:

* **Java Development Kit (JDK):** Descărcați și instalați JDK versiunea 17 sau o versiune ulterioară compatibilă cu Spring Boot de pe site-ul oficial Oracle()
* **IntelliJ IDEA Ultimate Edition:** Descărcați și instalați IntelliJ IDEA Ultimate Edition de pe site-ul JetBrains (). Această ediție oferă suport complet pentru dezvoltarea Spring Boot și integrare cu Maven. Este necesara modificarea campurilor „username” si „password” din application.properties, cu credențialele specifice fiecarui utilizator, ca in figura 7.1 .

A screen shot of a computer

Description automatically generated

Figura 7.1 application.propreties

* **MySQL WorkBench:** Descărcați și instalați MySQL WorkBench versiunea 8.0 sau ulterioară de pe site-ul oficial MySQL (). Configurați un utilizator și o parolă pentru accesul la baza de date. Dupa instalare se creeaza doua baze de date pentru microserviciu principal „SportPlanner” si o baza de date pentru microserviciu „Auth-Service”, ca in figura 7.2 .

A screenshot of a computer

Description automatically generated

Figura 7.2 crearea bazei de date

* **Node.js și npm:** Descărcați și instalați Node.js și npm de pe site-ul oficial Node.js. Acestea sunt necesare pentru dezvoltarea frontend-ului Vue.js.
* **Visual Studio Code:** Descărcați și instalați Visual Studio Code de pe site-ul oficial Microsoft (). Acesta va fi folosit ca editor de cod pentru frontend.

## Manual de utilizare

Primul pas al aplicației este înregistrarea unui nou utilizator. Această procedură poate fi efectuată așa cum se vede în figura 7.3 prin intermediul paginii de inregi înregistrare.

A screenshot of a login form

Description automatically generated

Figura 7.3 Pagina de înregistrarea utilizator

Urmatorul pas, este cel de autentificare cu noile credențialele, ca in figura 7.4. De asemenea, daca utilizatorul nu își mai aminteste parola, o poate recupera, daca introduce un email valid. Acesta va primi un mail, in care va putea sa își seteze o noua parola, ca in figura 7.5.

A screenshot of a login screen

Description automatically generatedA screenshot of a computer

Description automatically generated

Figura 7.4 Pagina de autentificare Figura 7.5 Pagina de recuperare parola

După autentificarea cu succes, utilizatorul va fi întâmpinat de pagina principală a aplicației, numită "Explore" (Figura 7.6). Această pagină prezintă o listă completă a meciurilor viitoare, permițând utilizatorului să își personalizeze experiența prin aplicarea de filtre în funcție de preferințele sale.

A screenshot of a computer

Description automatically generated

Figura 7.6 Pagina de explore

Pentru a afla mai multe detalii despre un anumit meci, utilizatorul trebuie să apese pe acesta. Această acțiune îl va redirecționa către o pagină dedicată meciului respectiv (Figura 7.7), unde va găsi toate informațiile necesare despre acel meci. De asemenea, utilizatorul va avea putea participa la meci, prin intermediul butonului “I want to play”.

A screenshot of a computer

Description automatically generated

Figura 7.7 Pagina de vizualizare meci

Aplicația oferă și funcționalitatea de a crea un meci nou. Prin apăsarea butonului "Create Match", utilizatorul este condus către pagina dedicată acestui proces. Aici, poate completa toate detaliile necesare pentru organizarea meciului, precum data, ora, locația, tipul sportului și numărul de jucători, (Figura 7.8).

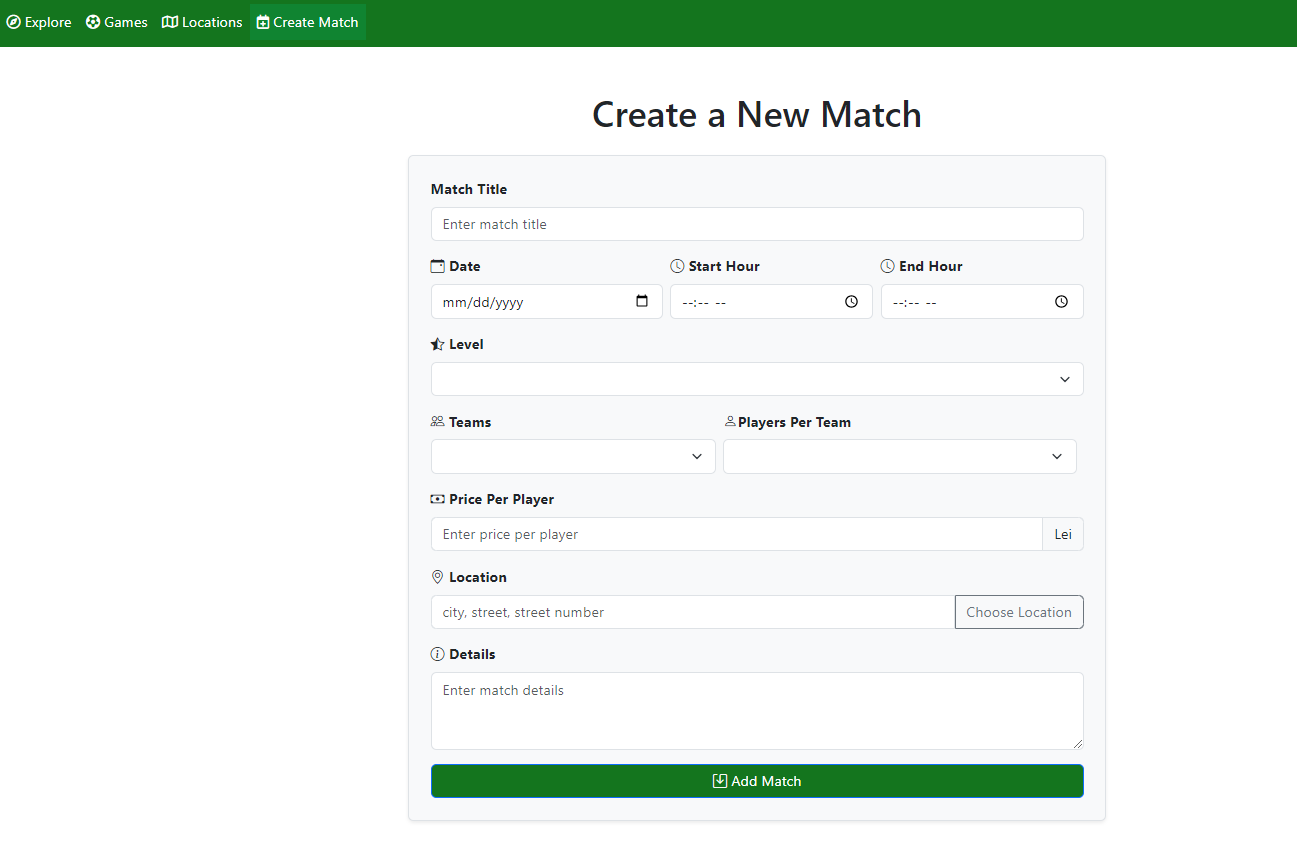


Figura 7.8 Pagina de creare meci

Odată ce meciul a fost creat și salvat în sistem, acesta va fi vizibil în secțiunea "Games". În această secțiune, utilizatorul poate urmări atât meciurile pe care le-a creat, cât și cele la care a fost invitat sau la care a participat deja, având astfel o imagine de ansamblu asupra activității sale sportive în cadrul platformei, (Figura 7.9).

A screenshot of a match list

Description automatically generated

Figura 7.9 Pagina de vizualizare a meciurilor la care participa utilizatorul

În calitate de organizator al unui meci, ai posibilitatea de a invita și alți utilizatori să participe. Aceasta se realizează prin intermediul butonului "Invite", care te va ghida către o interfață intuitivă de căutare și selecție a jucătorilor. Poți căuta utilizatori după nume sau poți aplica filtre pentru a găsi jucători care se potrivesc nevoilor tale, de exemplu, în funcție de poziția pe care o ocupă în teren (Figura 7.10).

A screenshot of a computer

Description automatically generated

Figura 7.10 Modal de invitare utilizatori la meci

Secțiunea "Locations" îți oferă o perspectivă completă asupra tuturor terenurilor de sport disponibile în aplicație. Cu un click pe o locație, vei avea acces la detalii suplimentare, programul de funcționare și, cel mai important, poziționarea exactă pe hartă, pentru a te ajuta să ajungi acolo fără probleme (Figura 7.11).

A screenshot of a sports field

Description automatically generated

Figura 7.10 Vizuallizare locații disponibile

A screenshot of a sports field

Description automatically generated

Figura 7.11 Vizualizare detalii locație

Astfel, am prezentat un ghid al funcționalităților cheie oferite de aplicație, ilustrând modul în care utilizatorul poate naviga și interacționa cu acestea pentru a-și optimiza experiența sportivă.

# Concluzii



## Contribuțiile personale

Contribuțiile personale aduse acestui sistem includ furnizarea de soluții de proiectare, dezvoltare, testare și extensibilitate care satisfac cerințele funcționale și nefuncționale. Cele mai importante aspecte realizate sunt:

* Funcționalități cheie: Am oferit o soluție completă pentru dezvoltarea unei platforme digitale care oferă posibilitatea de a crea conturi, de a gestiona profilurile, de a organiza și de a participa la meciuri sportive. Utilizatorii pot căuta și vizualiza meciuri, pot trimite invitații și pot participa la alte meciuri organizate de către alți utilizatori. Aceste funcționalități asigură o experiență de utilizare intuitivă și eficientă.
* **Gestionarea locațiilor:** Aplicația include un sistem de gestionare a locațiilor, permițând utilizatorilor să vizualizeze terenurile de joc pe o hartă integrată. Locațiile sunt afișate cu detalii relevante, cum ar fi adresa exactă, coordonatele geografice și facilitățile disponibile.
* Comunicare eficientă: Platforma facilitează comunicarea între utilizatori, oferind funcționalități de trimitere a invitațiilor și de notificare. Astfel, sportivii și organizatorii pot coordona eficient detaliile meciurilor și antrenamentelor.
* Interfață de utilizator prietenoasă: Am dezvoltat o interfață de utilizator modernă și ușor de utilizat, utilizând framework-ul Vue.js. Aceasta permite utilizatorilor să navigheze rapid prin platformă și să acceseze toate funcționalitățile necesare cu minim de efort.
* Securitatea și integritatea datelor: Am implementat măsuri de securitate riguroase pentru a proteja datele utilizatorilor și pentru a asigura integritatea și confidențialitatea informațiilor. Aceste măsuri includ autentificarea utilizatorilor, criptarea datelor și protecția împotriva atacurilor cibernetice.
* Susținerea fotbalului la nivel local: Proiectul poate susține dezvoltarea fotbalului la nivel local prin promovarea activităților sportive și prin facilitarea organizării de evenimente. Astfel, platforma poate contribui la creșterea interesului pentru sport și la îmbunătățirea sănătății și bunăstării comunității.

## Analiza critică

Analiza critică a aplicației evidențiază câteva limitări semnificative, ce ar putea afecta puțin imaginea aplicația și experiența utilizatorilor, acestea fiind:

* Lipsa unui sistem de raportare a problemelor: Aplicația nu oferă utilizatorilor posibilitatea de a raporta probleme precum locații indisponibile, jucători care nu se prezintă la meci sau comportamente neadecvate. Un astfel de sistem ar permite o mai bună gestionare a incidentelor și ar contribui la îmbunătățirea experienței utilizatorilor.
* Lipsa mecanismului de verificare a autenticității informațiilor: Aplicația nu dispune de un mecanism de verificare a autenticității informațiilor oferite de organizatori, cum ar fi confirmarea închirierii terenului de sport. Această lipsă poate duce la situații în care utilizatorii se prezintă la un teren care nu este disponibil sau rezervat pentru ei, generând frustrare și neîncredere în platformă.
* Acurateței profilurilor utilizatorilor: Aplicația nu poate garanta că profilul unui utilizator reflectă cu acuratețe realitatea. Nu există nicio modalitate de a verifica dacă un utilizator joacă într-adevăr pe poziția specificată, dacă nivelul său de joc este cel declarat sau dacă profilul este creat și gestionat chiar de către jucătorul respectiv. Această lipsă de verificare poate conduce la formarea de echipe dezechilibrate și la experiențe de joc nesatisfăcătoare pentru utilizatori.

## Posibile îmbunătățiri

Aplicația a reușit să îndeplinească principalele obiective, oferind o platformă eficientă pentru organizarea și gestionarea meciurilor sportive. Totuși, există mai multe posibilitați de a îmbunătăți și a extinde funcționalitatea aplicației, precum:

* Rezervarea terenurilor: Dezvoltarea unor sisteme automate de rezervare și confirmare a terenurilor și altor resurse necesare pentru meciuri, ar putea reduce timpul și efortul necesar pentru organizare, făcând procesul mai eficient.
* Integrarea cu sisteme de plată: Integrarea cu diferite sisteme de plată pentru a permite plata online a taxelor de participare, închirieri de terenuri și alte servicii ar simplifica procesele administrative și ar oferi o experiență completă utilizatorilor.
* Implementarea funcționalității de chat: Îmbunătățirea funcționalităților de chat prin adăugarea de caracteristici avansate, cum ar fi chat de grup, mesaje vocale și video, și integrarea cu aplicații de mesagerie existente ar facilita o comunicare mai eficientă și diversificată.
* Implementarea unui sistem de evaluare și feedback: Introducerea unui sistem de evaluare și feedback pentru utilizatori, astfel încât organizatorii să poată primi comentarii și sugestii de îmbunătățire, ar putea contribui la creșterea calității serviciilor oferite.
* Organizarea de turnee și campionate: Dezvoltarea funcționalităților pentru organizarea și gestionarea de turnee sportive și campionate ar adăuga o nouă dimensiune platformei și ar putea aduce mai mulți utilizatori ai aplicației.

# Bibliografie

|  |  |
| --- | --- |
| [1] | „TeamSnap,” [Interactiv]. Available: https://www.teamsnap.com/. [Accesat 14 09 2022]. |
| [2] | „Heja,” [Interactiv]. Available: https://heja.io/. [Accesat 20 02 2024]. |
| [3] | „TacticalPad,” [Interactiv]. Available: https://www.tacticalpad.com/new/index.php. [Accesat 24 02 2024]. |
| [4] | „Calendis,” [Interactiv]. Available: https://www.calendis.ro/. [Accesat 05 10 2020]. |
| [5] | „Footy Addicts,” [Interactiv]. Available: https://footyaddicts.com/. [Accesat 07 04 2022]. |
| [6] | „Fubles,” [Interactiv]. Available: https://www.fubles.com/en/. [Accesat 08 03 2024]. |
| [7] | „Java,” [Interactiv]. Available: https://www.java.com/en/. [Accesat 05 01 2024]. |
| [8] | „Spring Boot,” [Interactiv]. Available: https://spring.io/. [Accesat 05 01 2024]. |
| [9] | „Maven Introduction,” [Interactiv]. Available: https://maven.apache.org/what-is-maven.html. [Accesat 05 01 2024]. |
| [10] | „MySQL,” [Interactiv]. Available: https://www.mysql.com/. [Accesat 06 01 2024]. |
| [11] | „VueJs Introduction,” [Interactiv]. Available: https://vuejs.org/guide/introduction.html. [Accesat 06 01 2024]. |
| [12] | „TypeScript Introduction,” [Interactiv]. Available: https://www.typescriptlang.org/docs/. [Accesat 06 01 2024]. |
| [13] | „Boostrap,” [Interactiv]. Available: https://getbootstrap.com/. [Accesat 06 01 2024]. |
| [14] | „CSS Tutorial,” [Interactiv]. Available: https://www.w3schools.com/css/. [Accesat 10 01 2024]. |
| [15] | „JetBrains,” [Interactiv]. Available: https://www.jetbrains.com/idea/. [Accesat 01 02 2024]. |
| [16] | „MySQL Workbench,” [Interactiv]. Available: https://www.mysql.com/products/workbench/. [Accesat 01 02 2024]. |
| [17] | „Visual Studio Code,” [Interactiv]. Available: https://code.visualstudio.com/. [Accesat 01 02 2024]. |
| [18] | J. G. J. M. H. F. L. M. P. L. a. T. B.-L. R. Fielding, „Hypertext Transfer Protocol,” *RFC Editor,* 01 June 1999. |
| [19] | R. Sureswaran, H. A. Bazar, O. Abouabdalla, A. M. Manasrah și H. El-Taj, „Active e-mail system SMTP protocol monitoring algorithm,” în *IEEE International Conference on Broadband Network & Multimedia Technology*, Beijing, China, 2009. |
| [20] | V. Pimentel și B. G. Nickerson, „Communicating and Displaying Real-Time Data with WebSocket,” *IEEE Internet Computing,* 04 Aug 2012. |
| [21] | J. B. a. N. S. M. Jones, „JSON Web Token (JWT),” *RFC Editor,* 01 May 2015. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Anexa 1

…

Secțiuni de cod relevante

# Anexa 2

Alte informații relevante (demonstrații etc.)

…

# Anexa 3

Lucrări publicate (dacă există)

etc.