



**UNIVERSITATEA
TEHNICĂ
DIN CLUJ-NAPOCA**

CATEDRA DE CALCULATOARE

FC Barcelona Website

INGINERIE SOFTWARE

Autor:

Nica Razvan-Emil

Grupa:

30237

Îndrumător proiect:

Adrian Burzo

Ianuarie
2026

Cuprins

1	Introducere	2
1.1	Scurtă descriere a proiectului	2
1.2	Cerințe Funcționale	2
1.3	Cerințe Non-Funcționale	2
2	Tehnologii Utilizate	4
2.1	Frontend - React.js	4
2.2	Backend - ASP.NET Core	4
2.3	Baza de Date	4
3	Arhitectura Sistemului și Diagrame UML	5
3.1	Diagrama Use Case	5
3.2	Diagrama de Activitate	6
3.3	Diagrama de Secvență	8
3.4	Diagrama de Stări	9
3.5	Design Patterns - Observer Pattern în React	9
4	Implementare și Funcționalități	11
4.1	Modulul Players - Gestiune și Interacțiune	11
4.2	Modulul Legends - Vizualizare Tacticală	11
4.3	Cinema Mode - Experiență Multimedia	12
4.4	Baza de Date și Securitatea Datelor	12
4.5	Resurse Online	12
5	Manual de utilizare (ReadMe)	13
5.1	Pornirea Backend-ului (.NET)	13
5.2	Pornirea Frontend-ului (React)	13
5.3	Utilizarea Bazei de Date	13

1. Introducere

1.1 Scurtă descriere a proiectului

Proiectul „FC Barcelona - Més que un club” reprezintă o platformă web dedicată istoriei și performanței unuia dintre cele mai mari cluburi de fotbal din lume. Aplicația oferă fanilor o experiență interactivă prin vizualizarea lotului de jucători, a legendelor clubului și a momentelor istorice.

1.2 Cerințe Funcționale

Cerințele funcționale definesc serviciile pe care aplicația trebuie să le ofere utilizatorului final:

- **Gestiunea lotului:** Sistemul trebuie să permită vizualizarea și căutarea jucătorilor din lotul actual.
- **Filtrare și Sortare:** Utilizatorul trebuie să poată filtra jucătorii după poziție sau naționalitate și să îi sorteze după metrici de performanță (goluri).
- **Comparație Jucători:** Posibilitatea de a selecta doi jucători pentru a le compara statisticile side-by-side într-o fereastră modală.
- **Modul Legende:** Afișarea interactivă a jucătorilor istorici într-o formație tactică specifică pe un teren de fotbal virtual.
- **Sistem de Vot (POTM):** Permitea utilizatorilor de a vota jucătorul lunii, cu salvarea stării pentru interactivitate.
- **Redare Multimedia:** Integrarea unui modul Cinema pentru vizionarea materialelor video despre istoria clubului.

1.3 Cerințe Non-Funcționale

Cerințele non-funcționale definesc constrângerile tehnice și parametrii de funcționare ai sistemului:

- **Integritatea Datelor:** Sistemul trebuie să asigure că datele despre jucători și voturi sunt persistente și nu se pierd la repornirea aplicației, prin utilizarea motorului SQLite.

- **Performanța API-ului:** Timpul de răspuns al serverului ASP.NET Core pentru cererile de tip GET trebuie să fie sub pragul de 300ms pentru a asigura o experiență fluidă.
- **Scalabilitate Orizontală:** Design-ul API-ului trebuie să permită adăugarea de noi endpoint-uri fără a modifica structura existentă a controlerelor de jucători.
- **Portabilitatea Bazei de Date:** Baza de date trebuie să fie de tip "file-based" (SQLite) pentru a permite mutarea întregului ecosistem între diferite medii de dezvoltare fără migrații complexe de SQL Server.

2. Tehnologii Utilizate

2.1 Frontend - React.js

Interfața utilizatorului este construită folosind biblioteca **React.js**, care permite crearea unei aplicații rapide de tip Single Page Application (SPA). Aceasta asigură o navigare fluidă între pagini, fără a reîncărca tot site-ul. Dezvoltarea s-a bazat pe utilizarea componentelor și a funcțiilor de bază:

- **useState**: utilizat pentru a reține și a actualiza datele pe ecran în timp real (de exemplu, când bifezi un filtru sau selectezi un jucător).
- **useEffect**: folosit pentru a aduce automat datele din baza de date imediat ce se deschide pagina.

Pentru scrierea și rularea codului am utilizat **Vite**, un instrument modern care ajută la pornirea rapidă a proiectului și la vizualizarea instantanee a modificărilor făcute în cod.

2.2 Backend - ASP.NET Core

Logica de server (server-side) este implementată folosind framework-ul **ASP.NET Core**, bazat pe limbajul C#. Această alegere asigură o arhitectură robustă și performantă, organizată sub forma unui **REST API**.

- **Entity Framework Core**: utilizat ca Object-Relational Mapper (ORM) pentru a facilita interacțiunea cu baza de date prin intermediul obiectelor C#, eliminând necesitatea scrierii manuale a interogărilor SQL complexe.
- **Routing**: sistemul de rutare gestionează cererile HTTP (GET, POST), mapându-le către metodele specifice din Controller.

2.3 Baza de Date

Pentru stocarea persistentă a informațiilor, sistemul utilizează **SQLite**, un motor de bază de date relațional. Structura bazei de date este normalizată pentru a asigura integritatea datelor și cuprinde tabele dedicate pentru:

- Jucătorii din lotul actual și statisticile lor de performanță.
- Legendele clubului, incluzând date despre poziționarea tactică pe teren și palmaresul de trofee.
- Înregistrările voturilor utilizatorilor pentru modulul interactiv.

3. Arhitectura Sistemului și Diagrame UML

Acest capitol detaliază structura software și comportamentul logic al aplicației prin intermediul diagramelor UML, evidențiind interacțiunile dintre componentele frontend și backend.

3.1 Diagrama Use Case

Diagrama Use Case descrie interacțiunile dintre utilizator și funcționalitățile principale oferite de sistem.

Descriere: Utilizatorul poate naviga prin aplicație pentru a vizualiza, filtra, căuta și sorta jucătorii din lotul actual.

- **Vizualizare detalii:** Acțiunea de vizualizare include automat afișarea statisticilor detaliate.
- **Interactivitate:** Utilizatorul are posibilitatea de a compara doi jucători (proces ce implică selectarea acestora și afișarea metricilor comparate) sau de a vota pentru "Player of the Month" (POTM).
- **Multimedia și Istorie:** Aplicația oferă acces la secțiunea Cinema Barcelona și la vizualizarea primului 11 istoric (Best 11).

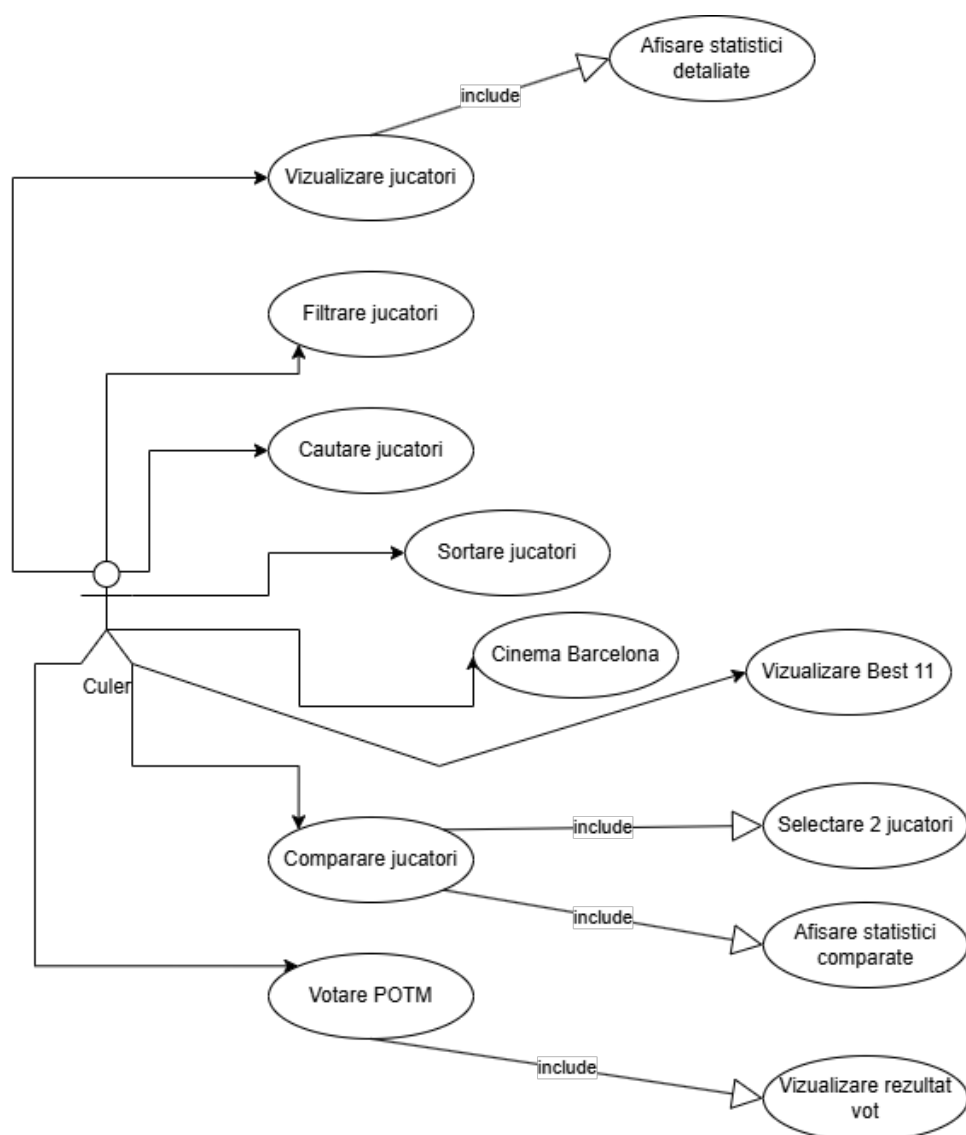


Figura 3.1: Diagrama Use Case - Interacțiunile utilizatorului cu sistemul

3.2 Diagrama de Activitate

Diagrama de activitate redă fluxul proceselor din perspectiva utilizatorului, de la accesarea paginii până la filtrarea avansată a datelor.

Descriere: Fluxul începe cu încărcarea listei de jucători. Utilizatorul poate aplica filtre succesive pentru poziție, naționalitate sau performanță. Sistemul evaluează aceste criterii și, în funcție de rezultate, permite vizualizarea detaliilor individuale sau compararea a doi jucători în mod simultan.

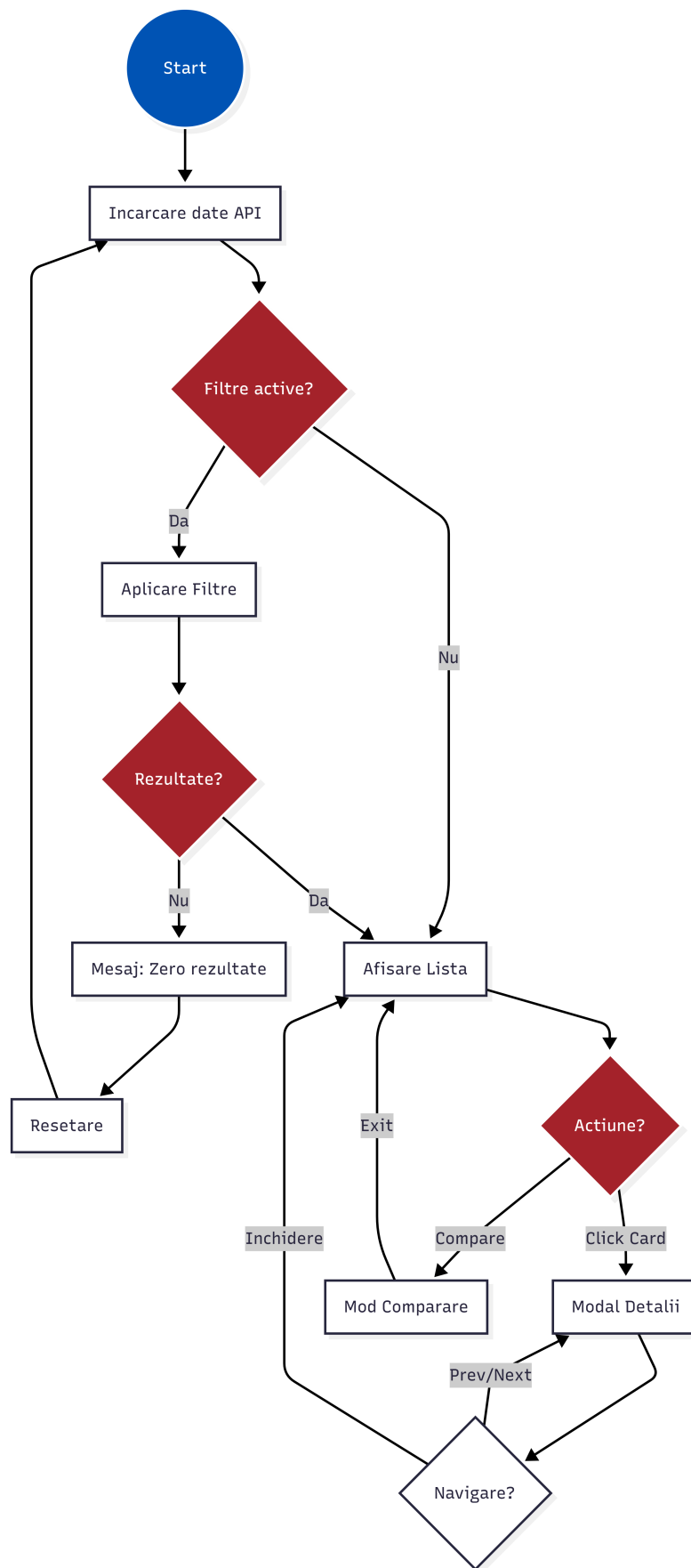


Figura 3.2: Diagrama de Activitate - Fluxul de navigare și procesare

3.3 Diagrama de Secvență

Această diagramă ilustrează comunicarea asincronă între componente pentru obținerea detaliilor unui jucător prin intermediul API-ului REST.

Descriere: La declanșarea evenimentului „Click pe card”, componenta React solicită datele prin `GET /api/players/{id}`. Serverul interoghează baza de date SQLite, iar obiectul JSON returnat este procesat local (ex: calculul vârstei) înainte de a fi afișat în interfață.

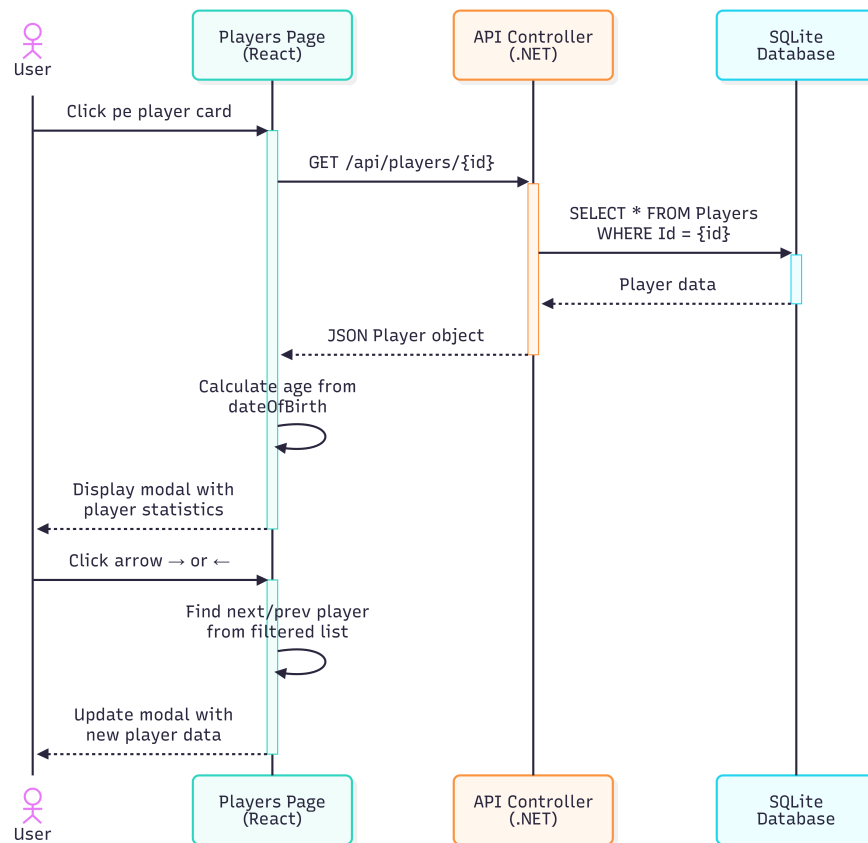


Figura 3.3: Diagrama de Secvență - Interacțiunea client-server

3.4 Diagrama de Stări

Diagrama de stări definește ciclul de viață al ferestrelor modale și modul în care interfața reacționează la input-ul utilizatorului.

Descriere: Fereastra de detalii tranzitează între stările „Închis”, „Deschidere” (animație) și „Afișat”. Din starea activă, utilizatorul poate naviga către alți jucători (starea „Navigare”) sau poate declanșa închiderea prin tastatură (Esc) sau click pe fundal, readucând sistemul în starea inițială.

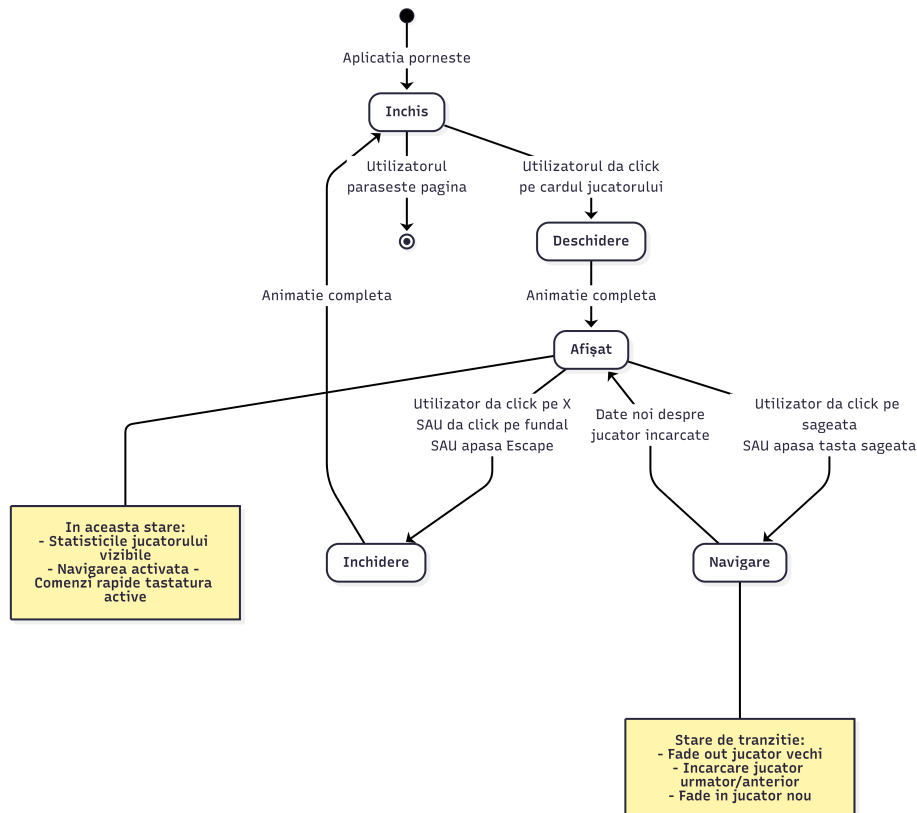


Figura 3.4: Diagrama de Stări - Fereastra Modală

3.5 Design Patterns - Observer Pattern în React

Aplicația utilizează design pattern-ul **Observer** pentru a asigura o interfață reactivă și fluidă. În dezvoltarea frontend cu React, acest pattern este implementat nativ prin intermediul sistemului de gestionare a stării (*State Management*).

Explicație și Implementare:

- **Mecanism:** Atunci când starea unei componente (ex: `selectedLegend` sau lista de `players`) se modifică, toate elementele de interfață care „observă” acea stare sunt notificate automat și se re-renderează pentru a reflecta noile date.
- **Hooks:** Acest proces este realizat prin utilizarea hook-urilor `useState` (care definește subiectul observat) și `useEffect` (care reacționează la schimbările acestuia).

- **Exemplu în proiect:** În pagina *Legends*, modalul este un „observator”. Imediat ce utilizatorul dă click pe un card, starea se schimbă, iar modalul reacționează instantaneu afișând detaliile corecte, fără a fi nevoie de o reîncărcare manuală a paginii.

4. Implementare și Funcționalități

Această secțiune descrie în detaliu modulele principale ale aplicației și modul în care tehnologiile frontend și backend interacționează pentru a oferi o experiență utilizator completă.

4.1 Modulul Players - Gestiune și Interacțiune

Acesta este cel mai complex modul al aplicației, oferind utilizatorului instrumente avansate de explorare a lotului actual de jucători:

- **Sistemul de Filtrare și Căutare:** Utilizatorul poate filtra jucătorii în timp real după poziție (GK, DF, MF, FW) sau naționalitate. Bara de căutare permite filtrarea instantanee după numele jucătorului, utilizând funcții de procesare a string-urilor în JavaScript.
- **Sortare Dinamică:** Datele pot fi ordonate crescător sau descrescător în funcție de nume, numărul de pe tricou sau performanțe (goluri marcate).
- **Player Card și Modal:** Fiecare jucător este prezentat printr-un card stilizat. La selecție, se deschide un modal (fereastră suprapusă) care încarcă asincron statistici detaliate.
- **Player Comparison:** O funcționalitate avansată care permite selectarea a doi jucători și afișarea lor în paralel pentru a compara metricile de performanță.
- **Sistemul de Vot (POTM):** Utilizatorii pot vota pentru „Player of the Month”, datele fiind transmise prin API și stocate permanent în tabelele SQLite; sistemul verifică un identificator unic al vizitatorului (*VoterIdentifier*) pentru a preveni votul multiplu și pentru a asigura corectitudinea clasamentului.

4.2 Modulul Legends - Vizualizare Tacticala

Secțiunea de Legende transpune datele istorice într-un format vizual inspirat din simulațiile sportive (FIFA Ultimate Team):

- **Afișare pe Teren:** Jucătorii nu sunt listați simplu, ci sunt poziționați pe un teren de fotbal virtual. Coordonatele acestora sunt preluate din baza de date (*FormationX*, *FormationY*) și aplicate dinamic prin CSS.
- **Design Premium:** Cardurile legendelor utilizează un gradient auriu metalic și efecte de strălucire la trecerea cursorului (hover), subliniind statutul lor special.

- **Gestiunea Managerului:** O zonă dedicată antrenorului, situată sub teren, oferă detalii despre perioada petrecută la club și numărul total de trofee câștigate.

4.3 Cinema Mode - Experiență Multimedia

Această pagină oferă o experiență de vizionare relaxantă, dedicată istoriei clubului.

- **Design de Cinema:** Pagina folosește un fundal negru complet pentru a scoate în evidență videoclipul și pentru a crea atmosfera unei săli de cinema.
- **Integrare YouTube:** Videoclipul este rulat direct prin YouTube, permițând utilizatorului să schimbe rezoluția sau să îl vadă pe tot ecranul.
- **Animatie de Start:** La intrarea pe pagină, apare o animație cu o rolă de film care rulează până când ecranul de cinema este gata de vizionare.

4.4 Baza de Date și Securitatea Datelor

Toate aceste funcționalități sunt susținute de un backend în C# care asigură integritatea datelor prin Entity Framework. Utilizarea SQLite permite portabilitatea proiectului, baza de date fiind livrată direct cu aplicația, facilitând instalarea imediată pe orice mediu de testare.

4.5 Resurse Online

Repository GitHub: https://github.com/zRazvaN/FC_BARCELONA

5. Manual de utilizare (ReadMe)

Pentru a rula aplicația local, trebuie urmați următorii pași:

5.1 Pornirea Backend-ului (.NET)

1. Se deschide folderul `FCBarcelona.Server` într-un terminal sau în Visual Studio.
2. Se execută comanda `dotnet run` pentru a porni serverul API și baza de date SQLite.
3. Serverul va fi disponibil implicit la adresa `https://localhost:5173`.

5.2 Pornirea Frontend-ului (React)

1. Se deschide folderul `fcbarcelona.client` în terminal.
2. Se execută `npm install` pentru instalarea bibliotecilor necesare.
3. Se execută `npm run dev` pentru a lansa interfața în browser.

5.3 Utilizarea Bazei de Date

Baza de date este inclusă sub forma fișierului `BarcaData.db`, deci nu este necesară nicio configurare suplimentară de server SQL.

Bibliografie

- [1] React Documentation, <https://react.dev/>
- [2] ASP.NET Core Documentation, <https://learn.microsoft.com/en-us/aspnet/core/>
- [3] Entity Framework Core Docs, <https://learn.microsoft.com/en-us/ef/core/>