SQUAD MANAGER

★ SPORT TEAM MANAGEMENT APP ★

- Architectural Report -

Developed by

↳ Avian Silviu

↳ Dijmărescu Cristina

↳ Haiducu Ștefan

↳ Ioan Tudor

↳ Ionescu Alexandru

↳ Stan Ana-Maria

❖ Identify the main quality attributes of your application

Squad Manager este o aplicație web dezvoltată in framework-ul React, utilizand JavaScript ca tehnologie principala.

“Squad Manager” este o aplicație web care ajută în gestionarea unui club sportiv de fotbal. Aplicația a fost dezvoltată special pentru a ușura munca cluburilor sportive, ajutându-i pe oamenii ce activează în industria sportivă să gestioneze mai facil clubul, să aibă acces oricând la toate informațiile echipei. Acest software preia din atribuțiile managerului clubului, memorând toate datele importante de la nivelul clubului. Astfel, aplicația este mereu pregătită să îți ofere orice informație ai nevoie referitoare la echipa ta.

Aplicația are mai multe roluri pentru utilizator, inclusiv administrator, membru al staffului și jucător, fiecare cu acces diferit la functionalitatile aplicatiei.

In Squad Manager sunt integrate si functionalitati ale altor aplicatii, cum ar fi Excel, PDF pentru a exporta usor date și API-urile meteo pentru a vizualiza datele meteorologice corespunzatoare meciurilor viitoare. Datele aplicatiei sunt stocate cu ajutorul platformei Firestore, astfel Firebase este folosita ca baza de date principală pentru a salva datele fiecarei sectiuni din club.

Scopul principal al aplicației este de a îmbunătăți organizarea și performanța clubului, oferind o modalitate ușoară pentru membrii clubului de a accesa și gestiona datele și activitățile clubului.

Atributele principale ale aplicației Squad Manager sunt:

- Simplitate in utilizare - aplicația este ușor de utilizat de catre utilizatori cu diferite niveluri de experiență tehnică
- Performanța - aplicația poate gestiona numarul de utilizatori prezenti in cadrul unui club sportiv și poate raspunde rapid cerintelor acestora.
- Scalabilitatea - aplicația are posibilitatea de a fi imbunatatita pentru a gestiona un numar ridicat de utilizatori și date in viitor, daca clubul isi mareste echipa.
- Accesibilitatea - aplicația este accesibilă utilizatorilor de pe diferite dispozitive și din orice locatie.
- Managementul datelor - aplicația ofera o modalitate eficientă de a stoca, gestiona și accesa datele importante pentru club, utilizand platforma Firebase.

- Roluri și permisiuni specifice utilizatorului - aplicația are roluri și permisiuni diferite pentru utilizatorii din club (jucator, staff sau admin)

- Integrarea - aplicația este integrată cu servicii și platforme externe pentru a oferi mai multe functionalitati utilizatorilor, cum ar fi exportul datelor din pagini in Excel/PDF si afisare vreme pentru meciurile viitoare folosind API-uri.

- Generare statistici - aplicația ofera funcționalitatea de vizualizare si exportare statistici pentru a ajuta clubul să analizeze și să ia decizii bazate pe date concrete.

- Îmbunătățirea continuă - aplicația poate fi actualizată și îmbunătățită continuu pentru a se adapta nevoilor dinamice ale clubului.

❖ Reason why are those the most important attributes

Atributele prezentate anterior sunt importante pentru arhitectura aplicatiei deorece ating toate cerintele clubului și oferta utilizatorilor săi posibilitatea de a gestiona eficient datele de la nivelul echipei.

Stakeholderii care necesită aceste atribute sunt chiar cei ce vor utiliza aplicatia, adica managerii clubului, membrii staff-ului și jucătorii. Ei au nevoie de o aplicație ușor de utilizat și intuitivă, care poate gestiona un numar de utilizatori egal cu numarul angajatilor din cadrul echipei si care le poate oferi acestora datele și functionalitatile de care au nevoie pentru a manageria clubul și a îmbunătăți performanța in competitiile sportive.

Stakeholderii au nevoie de o aplicatie sigura, in care doar utilizatorii logati cu un anumit rol sa poata efectua actiuni de modificare a bazei cu datele clubului. O aplicatie care nu ofera securitate utilizatorilor sai pune in pericol coerenta si corectitudinea datelor afisate, astfel incat increderea in statisticile furnizate scade considerabil. De aceea, este un atribut important al aplicatiei optiunea de autentificare.

❖ Exemplify tactics used to satisfy QAs

- Realizarea de teste manuale pentru a verifica performanța aplicației și pentru a detecta eventualele erori.
- Monitorizarea traficului utilizatorilor in Dashboard-ul din Firebase pentru a analiza modul în care se utilizează aplicația și pentru a identifica zonele care necesită îmbunătățiri.
- Utilizarea platformei GitHub pentru a ține evidența modificărilor făcute de fiecare membru al echipei noastre și pentru a permite oricand întoarcerea la versiuni anterioare dacă este necesar.
- Implementarea functionalitatii de autentificare pentru a mentine securitatea aplicatiei la standardele impuse de stakeholderi.
- Utilizarea unui sistem sigur de management al bazelor de date pentru a gestiona și a proteja datele clubului cum ar fi Firebase, o platforma implementata de Google.
- Utilizarea unui sistem de suport prin email pentru a răspunde rapid la cererile utilizatorilor și pentru a rezolva problemele raportate.
- Participarea la cursuri și evenimente din cadrul cursurilor pentru a fi mereu la curent cu cele mai noi tehnologii software și pentru a îmbunătăți aplicația cu ideile descoperite prin aceste prezentari.

❖ Design Patterns used in the application

Motivele pentru care am folosit Design Pattern-urile urmatoare:

- De ce am folosit Functional Component? - Exemplu: Pagina Home

  Ele sunt mai simple și sunt folosite pentru componente care trebuie să fie randate, neavand nevoie de state cum ar fi pagina de Home.

- De ce am folosit Container Component? - Exemplu: Weather API

  Componentele containerului sunt adesea folosite pentru a gestiona logica complexă cu ajutorul state-urilor, cum ar fi preluarea datelor din Weather API sau gestionarea interacțiunilor utilizatorului.

- De ce am folosit Compound Components? - Exemplu: Weather API

  Rolul principal al unei componente de container compus este de a acționa ca un container de nivel superior care gestionează starea și logica pentru mai multe componente ale containerului copil. In cazul nostru, widget-ul de vreme primeste state-ul de la parintele sau.

- De ce am folosit Conditional Rendering? - Exemplu: NavBar-ul se modifica in functie de rol

  Randarea condiționată este adesea folosită de noi pentru a crea interfețe dinamice și receptive, unde conținutul afișat pe ecran se poate modifica în funcție de acțiunile utilizatorului sau de alte evenimente.

- De ce am folosit Controlled Components? - Exemplu: Formulare de Add si Update

  Rolul principal al componentelor controlate este de a se asigura că starea componentei este întotdeauna sincronizată cu starea componentei părinte. In aplicatie noastra comunica formularul cu field-urile din cadrul acestuia.

- De ce am folosit React Hooks? - Exemplu: La afisarea jucatorilor

  React Hooks sunt funcții care ne-au permis să utilizam state-uri și alte caracteristici React în componente funcționale, făcându-le mai puternice și versatile. Am folosit UseEffect pentru a afisa lista jucatorilor la incarcarea paginii.

Benefits and Tradeoffs

➔ Functional Component

Beneficii:
-simplitate
-performanta
-posibilitatea de refolosire
-usurinta testarii

Compromisuri:
-functionalitate limitata

➔ Container Component

Beneficii:
-performanta
-posibilitatea de refolosire
-usurinta testarii

Compromisuri:
-complexitate

➔ Compound Components

Beneficii:
-performanta

Compromisuri:
-complexitate

➔ Conditional Rendering

Beneficii:
-interfata dinamica

Compromisuri:
-complexitate
-posibilitatea de refolosire limitata

➔ Controlled Components

Beneficii:
-comportament predictibil

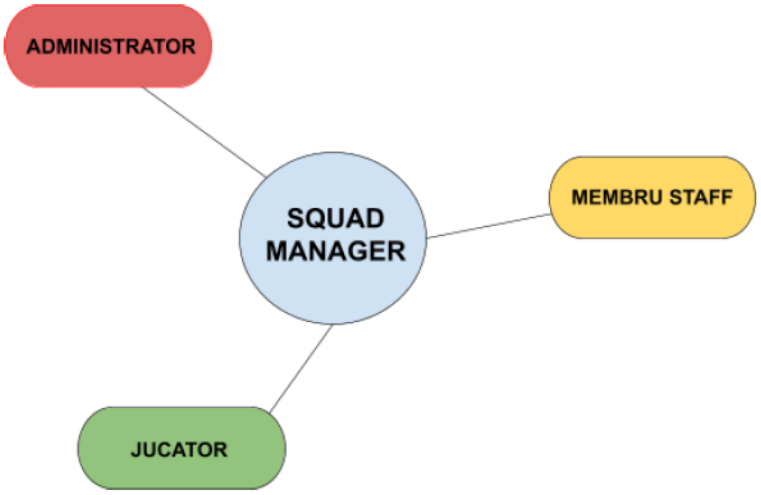
Compromisuri:
-complexitate
-posibilitatea de refolosire limitata

➔ React Hooks

Beneficii:
-cod mai scurt

Compromisuri:
-complexitate

❖ Users and their role



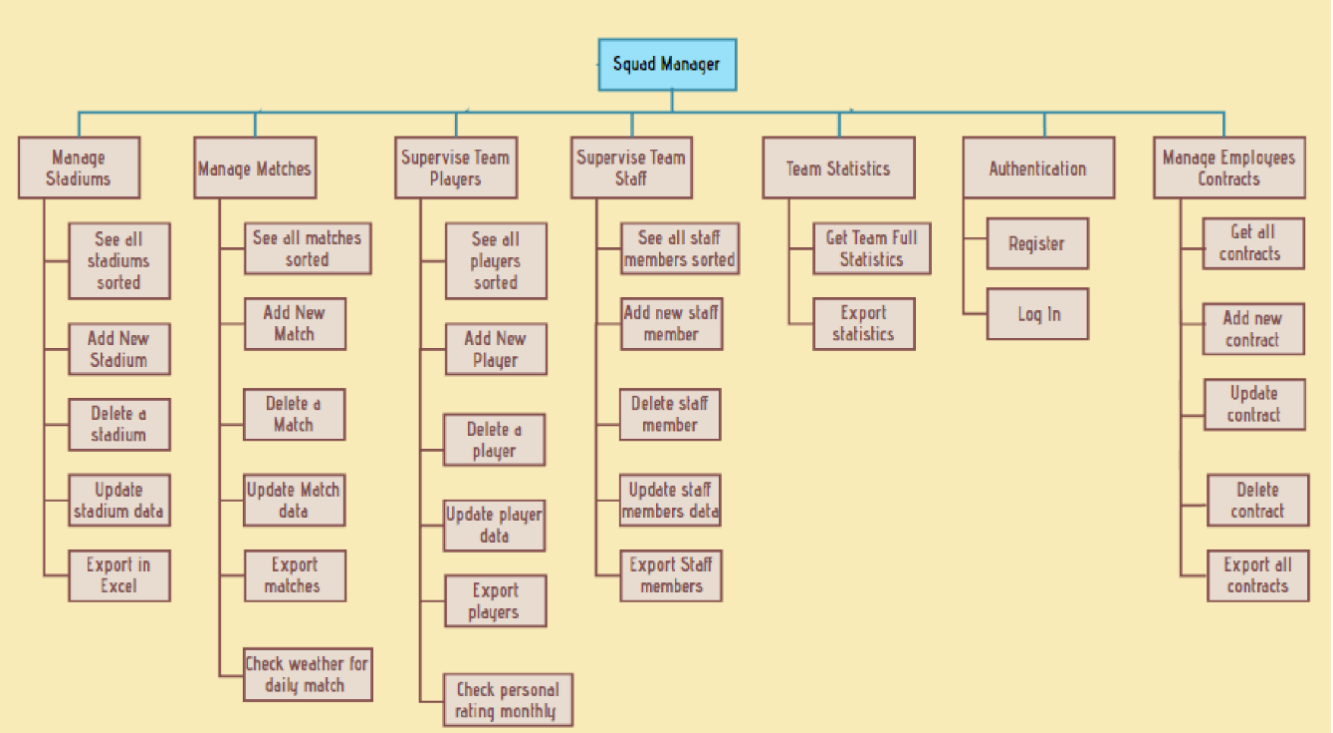
- ADMINISTRATOR:
  - are acces deplin la toate functionalitatile din cadrul aplicatiei
  - in pagina Profil are acces la datele sale personale

- MEMBRU STAFF:
  - in pagina Stadioane are acces la toate functionalitatile
  - in pagina Meciuri are acces la toate functionalitatile
  - in pagina Jucatori are acces la toate functionalitatile
  - in pagina Staff are acces doar in a vizualiza datele
  - in pagina Contracte nu are acces
  - in pagina Profil are acces la datele sale personale

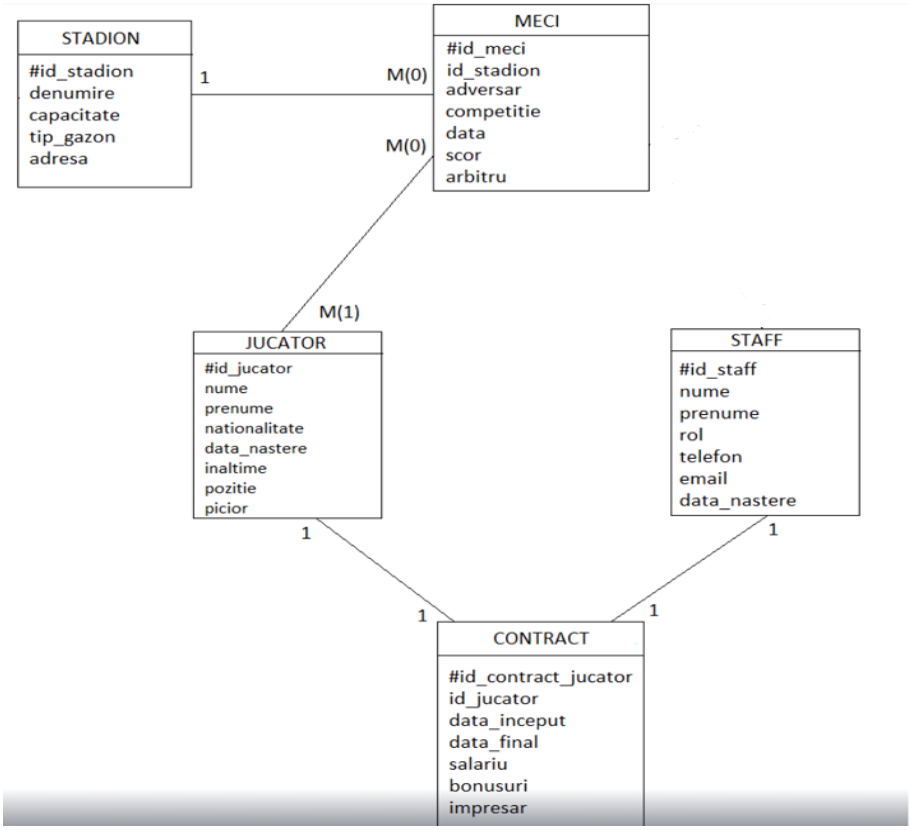
- JUCATOR:
  - in pagina Stadioane are acces doar in a vizualiza datele
  - in pagina Meciuri are acces doar in a vizualiza datele
  - in pagina Jucatori are acces doar in a vizualiza datele
  - in pagina Staff are acces doar in a vizualiza datele
  - in pagina Contracte nu are acces
  - in pagina Profil are acces la datele sale personale

- UTILIZATOR NELOGAT:
  - in pagina Stadioane are acces doar in a vizualiza datele
  - in pagina Meciuri are acces doar in a vizualiza datele
  - in pagina Jucatori are acces doar in a vizualiza datele
  - in pagina Staff are acces doar in a vizualiza datele
  - in pagina Contracte nu are acces

❖ Functional decomposition diagrama

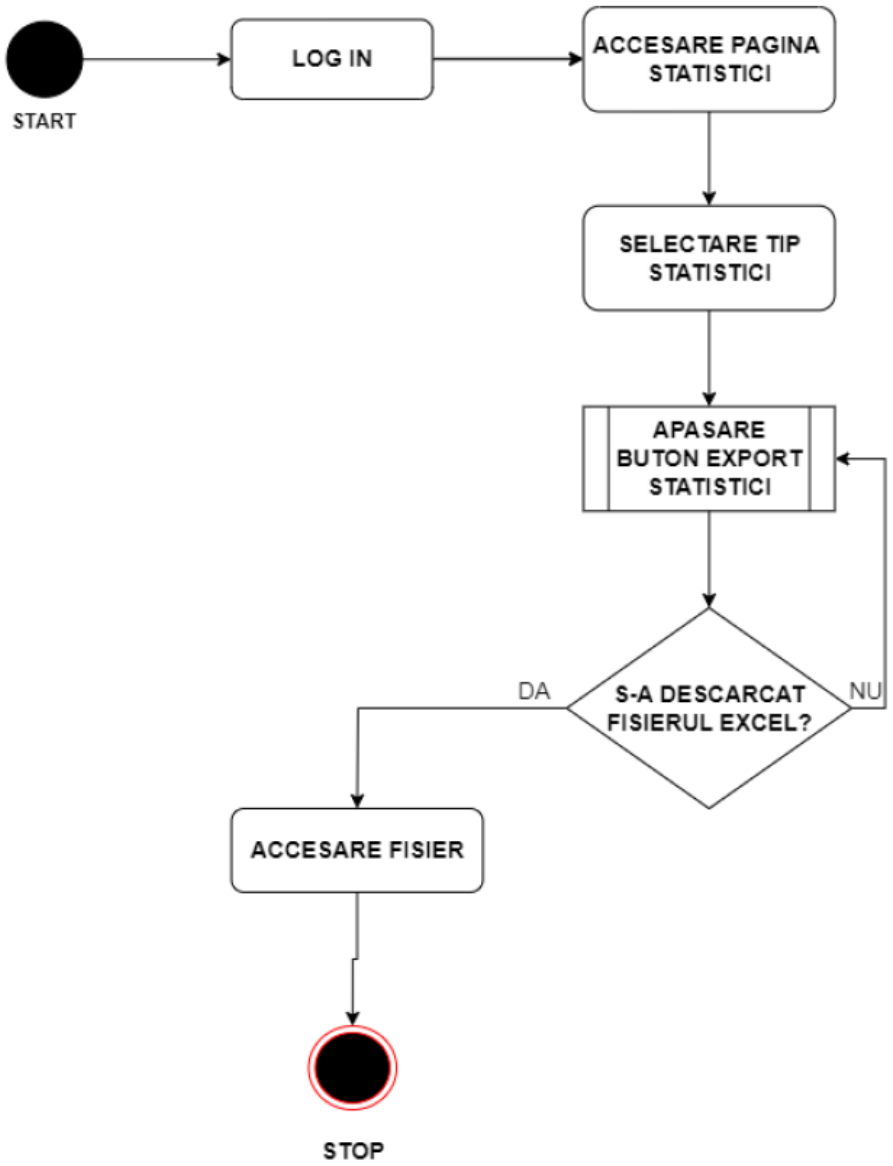


❖ Tables diagram



❖ Activity diagrams

- **Exportare statistici in Excel**



- **Adaugare meci nou in pagina dedicata meciurilor**