Formula 1 Simulation

Decean Alexandru

Cuprins

1. Scopul proiectului

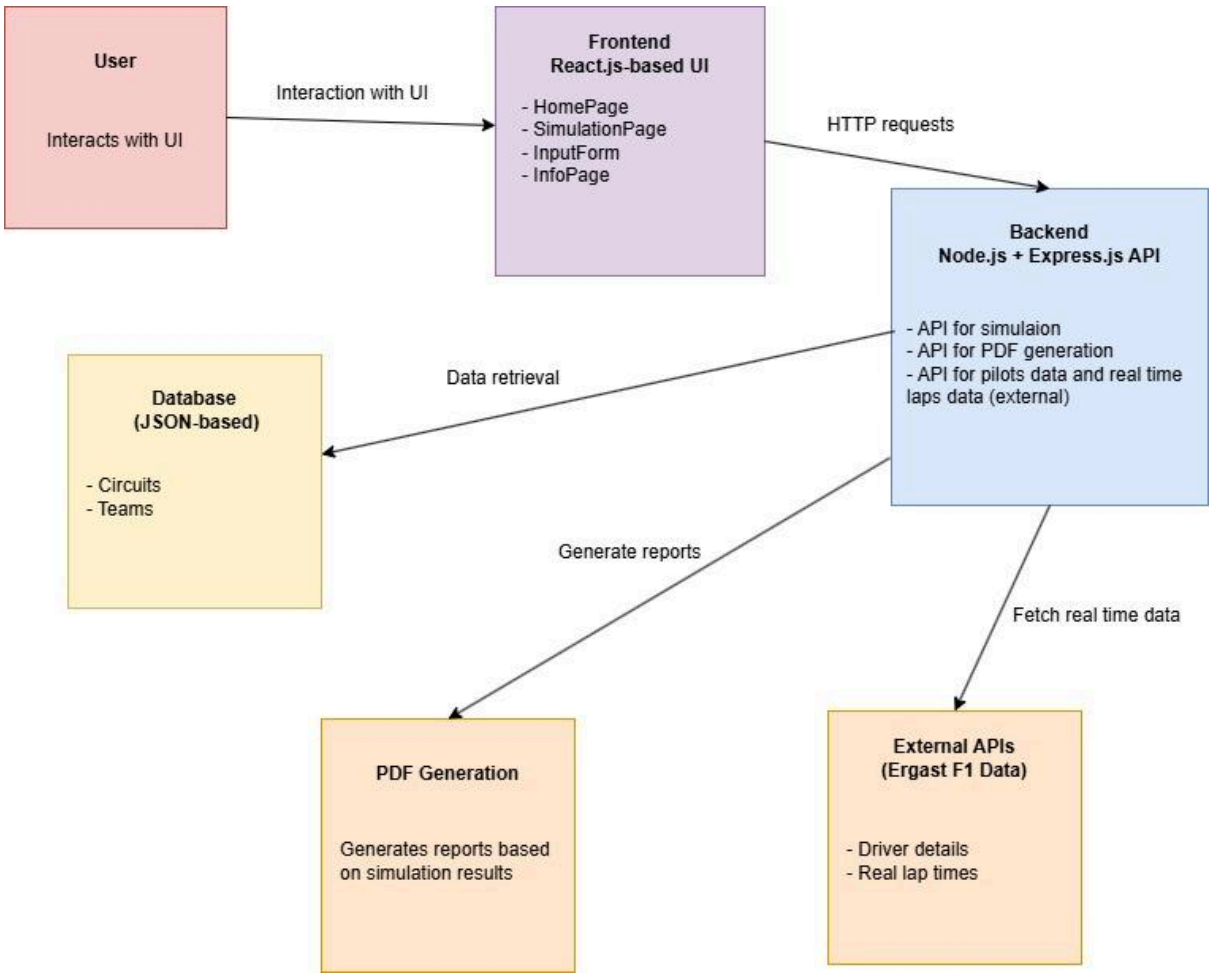
Acest proiect reprezintă o aplicație web interactivă care permite utilizatorilor să simuleze performanțele monoposturilor de Formula 1 pe diverse circuite. Proiectul a fost conceput cu scopul de a oferi o experiență educativă și practică, permițând utilizatorilor să înțeleagă cum factori precum echipa, modelul, condițiile meteo și pneurile influențează performanțele unei mașini de Formula 1.

De asemenea, aplicația oferă informații detaliate despre piloți, circuite și pneuri, utilizând date preluate dintr-un API extern (Ergast Developer API).

2. Caracteristici principale

- **Simulare realistă a performanțelor monoposturilor:**
  - Timp pe tur (simulat și real).
  - Viteză maximă.
- **Informații actualizate:**
  - Piloții Formula 1 din ultimii 3 ani (ultimele 3 sezoane).
  - Circuite și pneuri utilizate în Formula 1.
- **Generare rapoarte PDF:**
  - Exportarea rezultatelor simulării într-un format PDF.
- **Validare avansată:**
  - Formulare robuste pentru introducerea datelor utilizatorului.

3. Diagrama Arhitecturala



4. Instructiuni de instala si rulare

Cerințe preliminare

1. **Software necesar:**
    - Node.js (versiune 14 sau mai mare).
    - Git.
    - Git LFS pentru fișiere mari (opțional).
2. **Clonarea repository-ului:**
    - git clone https://github.com/alexandrudecean/Formula-1-Simulation.git
    - cd Formula-1-Simulation

Instalare Backend

1. Navighează în directorul backend: cd backend
2. Instalează dependențele: npm install
3. Pornește serverul: npm start

   Serverul va rula pe http://localhost:5000.

Instalare Frontend

1. Navighează în directorul frontend: cd frontend
2. Instalează dependențele: npm install
3. Pornește aplicația: npm start

   Aplicația va fi disponibilă pe http://localhost:3000.

5. API-uri Implementate

Rute backend principale:

1. GET /api/drivers
   - Returneaza o lista cu pilotii activi din sezonul curent.
   - Raspuns


```
[
  {
    "givenName": "Lewis",
    "familyName": "Hamilton",
    "nationality": "British",
    "dateOfBirth": "1985-01-07",
    "url":
"https://en.wikipedia.org/wiki/Lewis_Hamilton"
  }
]
```

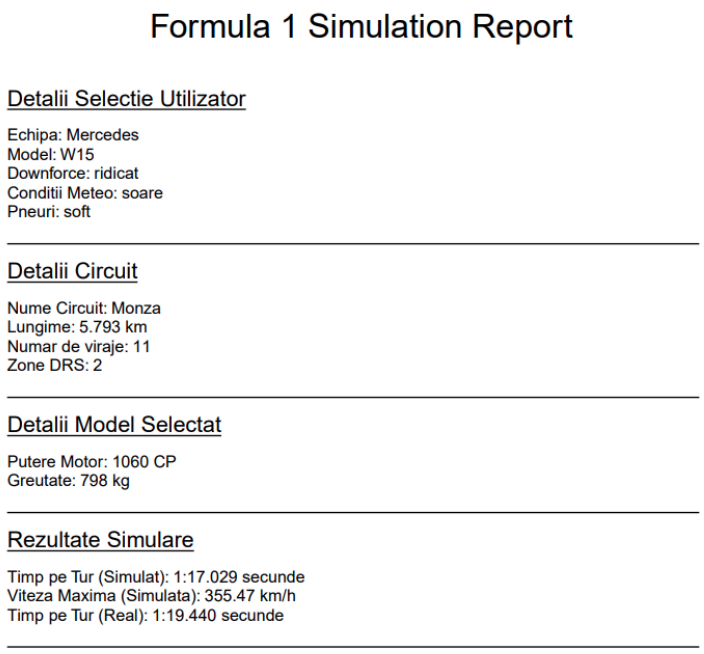

2. POST /api/simulation
   - Primeste datele din formular si returneaza datele simularii
   - Request:


```
{ "team": "Mercedes",
 "model": "W15",
 "circuit": "Monaco",
 "weather": "soare",
 "tires": "soft",
 "downforce": "ridicat" }
```


   - Raspuns:


```
{ "lapTime": "1:00.560",
 "maxSpeed": "337.43" }
```

3. POST /api/pdf
    - Genereaza un raport PDF pe baza datelor introduse de utilizator si a rezultatelor simularii.
    - Request-ul este cel din api/simulation la care se adauga si raspunsul cu lapTime si maxSpeed.
    - Raspuns



Formula 1 Simulation Report

Detalii Selectie Utilizator

Echipa: Mercedes
Model: W15
Downforce: ridicat
Conditii Meteo: soare
Pneuri: soft

Detalii Circuit

Nume Circuit: Monza
Lungime: 5.793 km
Numar de viraje: 11
Zone DRS: 2

Detalii Model Selectat

Putere Motor: 1060 CP
Greutate: 798 kg

Rezultate Simulare

Timp pe Tur (Simulat): 1:17.029 secunde
Viteza Maxima (Simulata): 355.47 km/h
Timp pe Tur (Real): 1:19.440 secunde

4. GET api/teams/lap-time
    - Returneaza timpul real pe tur pentru o anumita echipa, model si circuit.
    - Request: ?team=Mercedes&model=W15&circuit=Monaco
    - Raspuns: "bestLapTime": "1:10.543"

6. Design Patterns Utilizat

Factory Pattern este un design pattern utilizat pentru a crea obiecte fără a expune logica de creare directă în codul care le folosește. Acest pattern se bazează pe o metodă comună (factory method) care permite instanțierea unor obiecte bazate pe anumite condiții sau parametri.

Factory Pattern este folosit frecvent în situații în care:

- Sunt necesare mai multe tipuri de obiecte care împărtășesc o interfață comună.
- Se dorește separarea logicii de creare a obiectelor de logica de utilizare.
- Se dorește centralizarea și simplificarea instanțierii obiectelor.

Implementare în proiect

În acest proiect, Factory Pattern este implementat prin clasa TeamFactory, care centralizează logica de creare a obiectelor de tip echipă și model. Aceasta asigură flexibilitate, permite adăugarea rapidă de noi echipe și modele, și păstrează codul curat și modular.

Beneficii ale Factory Pattern în proiect

Flexibilitate: Poți adăuga rapid noi echipe și modele fără a modifica codul din alte părți ale proiectului.

Separarea responsabilităților: Logica de creare a echipelor este izolată, reducând complexitatea din alte părți ale codului.

Scalabilitate: Permite extinderea ușoară a proiectului, cum ar fi adăugarea unor echipe noi cu modele suplimentare.

7. Testare

Proiectul include mai multe tipuri de teste care pot fi rulate.

- Teste Unitare
    - Este verificata functionalitatea unor componente individuale precum InputForm si TeamFactory
    - Se pot rula cu comanda npm test
- Teste de integrare
    - Testeaza API-urile backend, cum ar fi /api/simulation si /api/pdf
    - Verifica raspunsurile API-ului la cereri valide si situatii de eroare.
    - Se pot rula cu comanda npm test
- Test de performanta
    - Evalueaza cat de bine gestioneaza serverul backend o sarcina mare (cereri multiple)
    - Pentru rulare se foloseste comanda npm run performance

8. Proces de dezvoltare

Am început dezvoltarea proiectului prin crearea structurii și organizarea folderelor. Inițial, am configurat fișierele JSON, am implementat serverul și am dezvoltat algoritmul principal. Ulterior, am implementat design pattern-ul TeamFactory pentru a gestiona eficient datele despre echipe și modele.

În partea de frontend, am început prin dezvoltarea paginii de simulare, împreună cu stilurile CSS corespunzătoare. Am extins funcționalitatea prin adăugarea opțiunii de descărcare a unui raport PDF generat din datele simularii. După mai multe ajustări și completări ale algoritmului pentru calcule precise, am adăugat și paginile Home și Info, asigurând separarea logicii CSS și JavaScript pentru fiecare pagină în parte.

Ulterior, am integrat un API extern pentru a prelua informații despre piloți activi, pe care le-am afișat în pagina Info. Am finisat design-ul CSS pentru paginile Home și Info, oferindu-le un aspect profesionist și intuitiv pentru utilizatori.

Am continuat dezvoltarea prin adăugarea unui al doilea API extern, utilizat pentru afișarea rezultatelor reale ale timpilor pe tur, în funcție de selecțiile utilizatorului din formularul de simulare. Aceste informații au fost, de asemenea, incluse în raportul PDF generat.

În etapa de testare, am implementat teste unitare, de integrare și de performanță pentru a asigura funcționalitatea corectă și stabilitatea aplicației.

În final, am actualizat fișierul README pentru a reflecta noile funcționalități adăugate și pentru a furniza instrucțiuni clare privind instalarea și utilizarea aplicației.

9. Extinderi

In cadrul aplicatiei pot fi adaugate sau implementate urmatoarele:

Implementarea unui Strategy Pattern pentru InputForm

Separarea logicii de validare și prelucrare a datelor din formular folosind Strategy Pattern. Această implementare ar permite adaptarea ușoară a validării pentru diferite tipuri de input sau scenarii viitoare.

Adăugarea dificultății circuitului în algoritm

Introducerea unui parametru suplimentar, precum dificultatea circuitului, care să influențeze timpul pe tur și performanța simulării. Acest lucru ar face algoritmul mai realist și ar adăuga complexitate simulării.

Extinderea informațiilor din InfoPage

Adăugarea unor informații generale despre Formula 1, cum ar fi istoria acestui sport, reguli de bază, recorduri sau tehnologii avansate utilizate. Aceasta ar îmbogăți experiența utilizatorului și ar face aplicația mai educativă.

Integrarea unui modul de feedback pentru utilizatori

Adăugarea unei funcționalități care să permită utilizatorilor să ofere feedback despre aplicație, astfel încât dezvoltatorii să poată adresa îmbunătățiri și sugestii.

Optimizarea interacțiunii cu API-ul extern

Reducerea timpilor de încărcare prin cache-ul datelor despre piloți și circuite, evitând astfel solicitările repetitive către API-ul extern.