Aplicatie de inchiriere a stadioanelor de fotbal

Gradinariu Robert-Iulian

18 ianuarie 2024

1 Introducere

Aplicatia de inchiriere a stadioanelor de fotbal este conceputa pentru a satisface nevoile specifice pasionatilor de fotbal care isi doresc sa experimenteze fotbalul la un alt nivel.

Ideea centrala a aplicatiei e creearea unei platforme care permite utilizatorului sa-si creeze cont si sa aiba in final posibilitatea de a juca pe un stadion, ne mai fiind nevoit sa priveasca totul printr un ecran mic de TV sau limitat in tribunele echipei sale preferate.

Aplicația servește, de asemenea, ca un instrument pentru administratori pentru a efectua operațiuni CRUD (Create, Read, Update, Delete) pe datele utilizatorilor, stadioanelor și echipamentelor.

2 Tehnologii folosite

2.1 Backend

Python: Alegerea limbajului Python pentru backend se datorează ușurinței sale de citire și scriere, ceea ce face dezvoltarea rapidă și eficientă. Python este renumit pentru comunitatea sa mare și bibliotecile sale bogate, ceea ce îl face ideal pentru o gamă largă de aplicații web.

Django:Django este un framework web de nivel înalt, scris în Python, care promovează dezvoltarea rapidă și designul curat, pragmatic. Acesta este ales pentru caracteristicile sale robuste, inclusiv:

ORM (Object-Relational Mapping): ORM-ul Django permite lucrul cu baza de date printr-o interfață Pythonică, evitând necesitatea scrierii directe de SQL. Aceasta face manipularea datelor mai intuitivă și reduce riscul de erori.

Model-View-Template Architecture: Separarea logică a aplicației în modele (structura datelor), views (logica de afișare) și templates (prezentarea datelor), ceea ce face codul mai organizat și mai ușor de întreținut.

Sistemul de autentificare: Django vine cu un sistem de autentificare încorporat, care facilitează gestionarea utilizatorilor, permisiunilor și sesiunilor, fiind vital pentru aplicația mea, unde gestionarea utilizatorilor este un aspect cheie.

Panou de administrare: Oferă un panou de administrare gata de utilizat pentru a gestiona aplicația, permițând administratorului să efectueze operațiuni CRUD asupra stadioanelor, echipamentelor și utilizatorilor.

Securitate: Django este proiectat cu gândul la securitate, oferind protecție împotriva unor vulnerabilități comune cum ar fi SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF) și altele.

2.2 Frontend

HTML (HyperText Markup Language): Este fundamentul oricărei pagini web, utilizat pentru a structura conținutul. În aplicația ta, HTML este folosit pentru a crea scheletul paginilor web, inclusiv structura pentru afișarea stadioanelor, formularele pentru închiriere și secțiunile de utilizator.

CSS (Cascading Style Sheets): CSS este utilizat pentru a stiliza elementele HTML. În aplicația ta, CSS adaugă estetică și design paginilor, oferind o experiență vizuală plăcută și o interfață utilizator intuitivă.

Bootstrap: Este un framework de frontend pentru dezvoltarea de site-uri și aplicații web responsive și mobile-first. În proiectul meu, Bootstrap este folosit pentru a crea un design responsiv care se adaptează la diferite dimensiuni de ecran și dispozitive, asigurând că aplicația este ușor accesibilă și utilizabilă pe smartphone-uri, tablete și desktopuri.

jQuery: O bibliotecă JavaScript rapidă, mică și bogată în funcții. jQuery simplifică lucruri precum manipularea documentelor HTML, gestionarea evenimentelor, animația și Ajax, facilitând dezvoltarea frontend-ului interactiv. În aplicația mea, jQuery este folosit pentru a adăuga dinamism paginilor, permițând interacțiuni asincrone cu backend-ul fără a reîncărca întreaga pagină, cum ar fi actualizarea datelor utilizatorului și afișarea stadioanelor disponibile.

Această combinație de tehnologii backend și frontend oferă un echilibru între un backend puternic și flexibil și o interfață utilizator atrăgătoare și receptivă, esențială pentru succesul aplicației mele de închiriere a stadioanelor de fotbal.

2.3 Baza de Date

Tabele și Relații: Am modelat tabele pentru utilizatori, stadioane, rezervări, echipamente, comenzi. Relațiile dintre aceste tabele (cum ar fi utilizatori către rezervări, stadioane către rezervări) trebuie gestionate pentru a reflecta cum interacționează entitățile în aplicație.

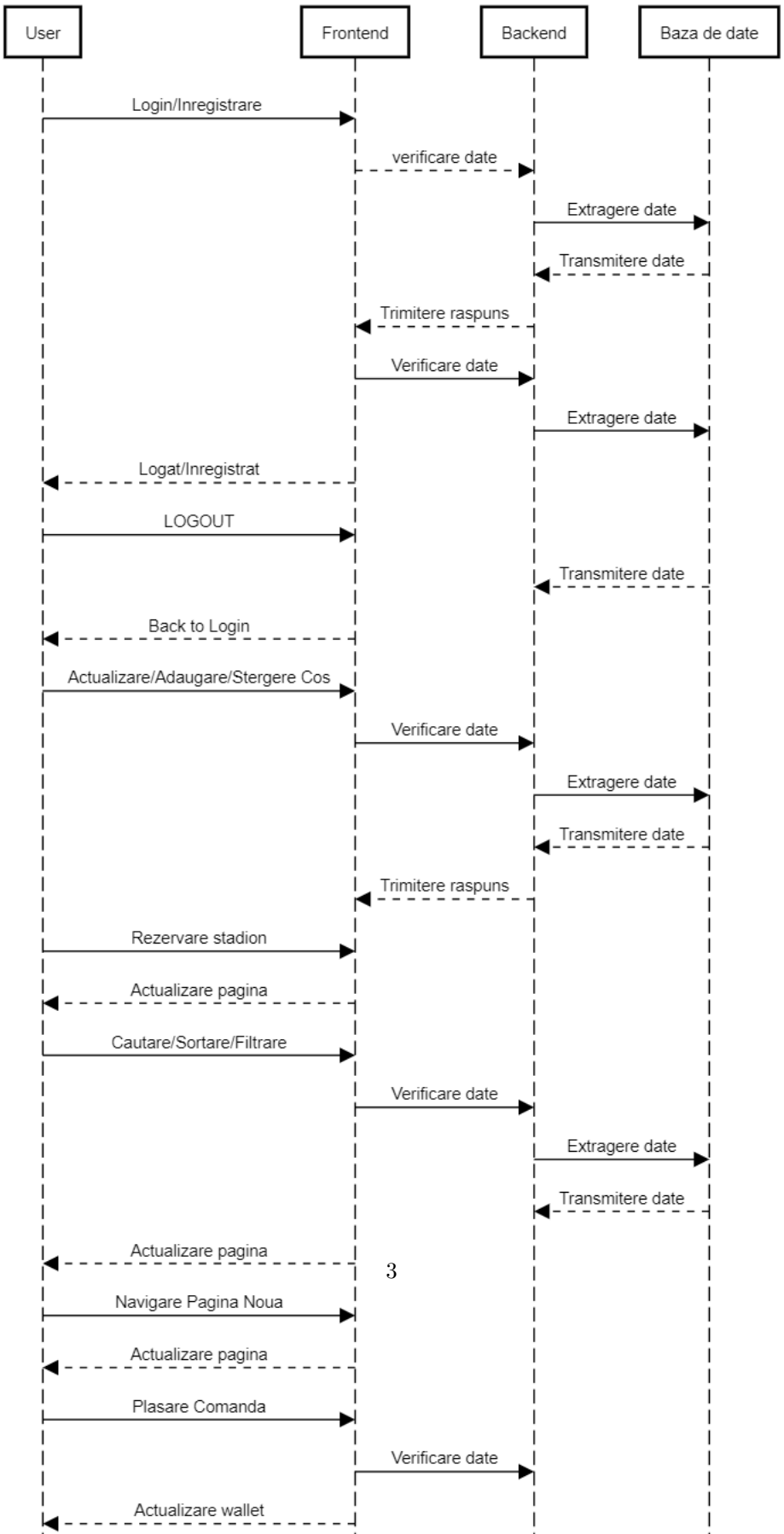
Normalizarea: Este important să mormalizez baza de date pentru a reduce redundanța și a îmbunătăți integritatea datelor. Acest lucru înseamnă structurarea tabelelor într-un mod care minimizează duplicarea datelor și facilitează întreținerea.

Tranzacții: Într-o aplicație care implică tranzacții financiare, cum ar fi încărcarea portofelului virtual sau plata pentru închirierea unui stadion, gestionarea tranzacțiilor în mod corespunzător este crucială pentru a asigura integritatea și consistența datelor. Interogări Eficace: Pentru a oferi o experiență de utilizator rapidă și eficientă, interogările bazei de date trebuie să fie optimizate. Acest lucru include utilizarea indexurilor pentru a accelera căutările și evitarea interogărilor ineficiente.

2.4 Diagrama de Secvențe

Descrierea și eventual o imagine cu diagrama de secvențe.

Diagrama de secvente



3 Design Pattern

In modelul Singleton, constructorul clasei este făcut privat și o metodă statică este folosită pentru a crea și/sau a returna instanța. Deoarece constructorul este privat, nu se pot crea noi instanțe ale clasei din exterior. În schimb, se utilizează metoda statică, care creează o instanță dacă aceasta nu există deja și returnează instanța existentă dacă aceasta a fost deja creată. Sistemul creează conexiuni la baza de date, un Singleton poate fi folosit pentru a gestiona această conexiune, asigurându-te că nu se creează conexiuni multiple inutil sau costisitoare.

Listing 1: exemplul de utilizare a SingleTone-ului

```
class DatabaseConnection:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super(DatabaseConnection, cls).__new__(cls)
            # Ini ializeaz  aici conexiunea la baza de date
        return cls._instance

# Accesarea instan ei
db_connection = DatabaseConnection()
```

4 Implementare

4.1 Backend

Structurarea Modelelor de Date Definirea Modelelor: Modelele în Django reprezintă structura tabelului în baza de date. Pentru proiectul meu, am avut nevoie de modele precum Customer (care extinde functionalitatile Userului prestabilit), Stadium, Equipment, Feedback si Order. Fiecare model este definit ca o clasă în Django și conține câmpuri care reprezintă coloanele tabelului.

Logică de Afaceri: View-urile în Django sunt responsabile de logica de afaceri. Ele preiau datele trimise de utilizator, procesează acele date (de exemplu, crearea unei noi rezervări), interacționează cu modelul pentru a citi sau scrie în baza de date și trimit un răspuns înapoi clientului.

Template-uri: View-urile sunt adesea legate de template-uri, care sunt fișiere HTML cu markup special pentru a permite inserarea de date dinamic. Template-urile sunt folosite pentru a genera pagini web care vor fi afișate utilizatorilor.

URL Routing: Django folosește un sistem de URL routing pentru a direcționa cererile HTTP către view-ul corespunzător. Definiești un set de reguli de URL în fișierul urls.py al proiectului, care indică ce view trebuie să gestioneze fiecare cale URL.

Listing 2: exemplul de utilizare a URL-urilor

```
from django.urls import path
from . import views

urlpatterns = [
```

```
    path('stadiums/', views.list_stadiums, name='list_stadiums'),
]
```

Interacțiunea cu Baza de Date ORM (Object-Relational Mapping): Django include un ORM puternic care te permite să interacționezi cu baza de date într-un mod mai intuitiv și sigur. ORM-ul transformă interogările și operațiunile asupra datelor în metode și proprietăți ale obiectelor Python.

CRUD Operations: Operațiunile CRUD (Create, Read, Update, Delete) sunt fundamentale în orice aplicație web și sunt implementate în Django prin metodele ORM. De exemplu, pentru a adăuga un nou stadion, ai crea un obiect Stadium și l-ai salva.

4.2 Frontend

Structura de Bază: HTML este folosit pentru a crea structura de bază a paginilor tale web. Acest lucru include elemente precum header-e, footer-e, form-uri pentru închirierea stadioanelor, listele de stadioane, secțiunile de utilizator și orice alt conținut specific paginii. Semantica: Folosirea elementelor HTML semantice ajută la structurarea paginilor într-un mod clar și logic, care este benefic pentru accesibilitate și SEO.

Stilizare și Design: CSS este utilizat pentru a defini stilul și aspectul paginilor tale web. Acest lucru include layout-uri, culori, fonturi, margini, padding-uri și alte elemente de design.

Responsive Design: Prin utilizarea CSS, în special cu ajutorul media queries, poți asigura că aplicația arată bine și funcționează pe o varietate de dispozitive și dimensiuni de ecran.

Componente UI Predefinite: Bootstrap oferă o colecție de componente UI predefinite, cum ar fi butoane, carduri, navbars, dropdowns, care pot fi folosite pentru a construi rapid interfața utilizator.

Grid System: Sistemul de grid din Bootstrap este esențial pentru crearea unor layout-uri responsive și aliniate. Acesta permite o aranjare ușoară și flexibilă a conținutului în diferite configurații și dimensiuni.

Interactivitate: JavaScript este folosit pentru a adăuga interactivitate paginilor web, cum ar fi răspunsul la acțiunile utilizatorilor, validarea formelor înainte de trimitere, și orice alte efecte sau funcționalități dinamice.

Manipularea DOM-ului: jQuery, o bibliotecă JavaScript, face manipularea DOM-ului (Document Object Model) mult mai ușoară și mai intuitivă. Acest lucru permite actualizări ale conținutului paginii fără a reîncărca întreaga pagină.

Cereri AJAX: jQuery simplifică, de asemenea, efectuarea de cereri AJAX, permitând aplicației tale să comunice asincron cu serverul pentru încărcarea datelor fără a reîncărca pagina, cum ar fi afișarea dinamică a disponibilității stadioanelor sau actualizarea datelor utilizatorului.

5 Concluzie

În final, succesul aplicației depinde de experiența utilizatorului final. Prin combinarea unui backend robust cu un frontend atrăgător și ușor de utilizat, aplicația este bine poziționată pentru a oferi o experiență satisfăcătoare utilizatorilor, fie că sunt clienți care caută să închirieze un stadion sau administratori care gestionează operațiunile zilnice.

În concluzie, proiectul de închiriere a stadioanelor de fotbal este un exemplu excelent de utilizare a tehnologiilor moderne pentru a crea o soluție complexă, care să răspundă nevoilor reale ale pieței și să ofere o experiență de utilizare de înaltă calitate. Acesta demonstrează puterea integrării

eficiente a diferitelor tehnologii și importanța unei abordări centrate pe utilizator în dezvoltarea aplicațiilor web.