**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

****

LUCRARE DE LICENȚĂ

**Playing Fields**

**propusă de**

**Ilie-Cătălin Prisacariu**

* + 1. **Sesiunea:** februarie, 2020

**Coordonator științific**

Colab. Florin Olariu

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

Playing Fields

Ilie-Cătălin Prisacariu

**Sesiunea:** februarie, 2020

**Coordonator științific**

Colab. Florin Olariu

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ………………………. specializarea …………………………………………………………, promoția …………………………., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Prenume Nume*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

**Cuprins**

[Introducere 7](#_Toc31220297)

[Capitolul 1: Aplicații similare 9](#_Toc31220298)

[1.1 SportBooking 9](#_Toc31220299)

[1.2 Playfinder 9](#_Toc31220300)

[Capitolul 2: Tehnologii folosite 11](#_Toc31220301)

[2.1 React 11](#_Toc31220302)

[2.2 Flask 11](#_Toc31220303)

[2.3 SQLAlchemy 12](#_Toc31220304)

[Capitolul 3: Funcționalitățile aplicației 13](#_Toc31220305)

[3.1 Înregistrare 13](#_Toc31220306)

[3.2 Autentificare 14](#_Toc31220307)

[3.3 Adăugarea unui teren de sport 15](#_Toc31220308)

[3.4 Căutare 16](#_Toc31220309)

[Capitolul 4: Structura aplicației 19](#_Toc31220310)

[4.1 Front-end 20](#_Toc31220311)

[Bibliography 23](#_Toc31220312)

# Introducere

Tema lucrării de licență a fost aleasă datorită faptului că nu am găsit o aplicație care să ofere informații despre terenurile de sport din oraș atunci când am avut nevoie de ele. Dacă există aplicații web care care ne oferă acces la diferite tipuri de anunțuri, precum cele despre domeniul auto, domeniul imobiliar sau obiecte second-hand, de ce nu ar exista și aplicații web care sa ofere informații despre terenuri de sport. Informații precum: adresa, date de contact, program, facilități, etc..

După alegerea temei de licență am făcut o cercetare amănunțită și am aflat că există o singură astfel de aplicație în România. Aplicația respectivă se numește SportBooking și oferă posibilitatea utilizatorilor de a vedea anumite informații despre un teren de sport și de a face o programare, însă pentru proprietarii terenurilor, aplicația nu este gratuită, motiv pentru care cred că această nu este populară și nu a fost adoptată de proprietarii terenurilor de sport din Iași, unii dintre ei preferând să facă anunțuri pe platforme de socializare sau neavând infomații despre ele nicăieri.

Sportul este o activitate în general recomandată oricărei persoane, indiferent de vârstă. Însă una dintre categoriile de vârstă pentru care ar trebui să fie un interes maxim este categoria tinerilor. Sportul reprezintă o activitate care poate influența stilul de viața, sănătatea sau personalitatea unui om. Este cunoscut faptul că tinerii care practică un sport înțeleg mult mai ușor ce înseamnă să câștigi sau să pierzi și faptul că pentru a câștiga trebuie să muncești din greu alături de membrii echipei. Fapt pentru care, mai târziu în viață, se vor integra mult mai ușor și vor face tot posibilul pentru a fi cei mai buni.

Pentru practicarea unui sport, fie la un nivel profesionist, fie la un nivel începător, este nevoie de un teren care să permită punerea în practică a acestuia. La nivelul actual tehnologie este destul de avansată, în România fiecare om poate avea acces la o rețea de internet pentru un preț foarte mic, rețea care să permită o conexiune bună.

Având aceste lucruri în considerare ar trebui ca o aplicație în care există detalii despre terenurile pe care se poate practica un sport să existe și să fie gratuită, atât pentru proprietarii acestora, cât și pentru oamenii care doresc să închirieze așa ceva.

Aplicația creată de către mine oferă posibilitate proprietarilor de terenuri de sport, baze de sport sau orice alt tip de unitate de sport să își creeze un cont în cadrul aplicației, iar ulterior să poată adăuga terenuri și detalii despre acestea. Iar pentru utilizatorii normali oferă posibilitatea să caute un anumit teren de sport, într-un anumit oraș și cu un anumit număr maxim de jucători. Pentru utilizatorii care doresc să caute un anumit teren de sport nefiind necesară crearea unui cont.

În Capitolul 1: am descris puțin aplicațiile similare, unde se află acestea și ce funcționalități oferă atât pentru proprietarii unor terenuri de sport cât și pentru un om care dorește să închirieze un teren de sport.

În Capitolul 2: am descris puțin tehnologiile folosite pentru realizarea aplicației, atât cele folosite pentru partea de front-end cât și cele folosite pentru partea de back-end.

În Capitolul 3: am descris funcționalitățile oferite de aplicație, atât pentru utilizator care are un cont în cadrul aplicației, aici mă refer la un proprietar al unui teren de sport, cât și funcționalitățile oferite unui utilizator care nu are un cont.

În Capitolul 4: am descris structura aplicației. Detalii despre partea de back-end cât și despre partea de front-end.

# Aplicații similare

## SportBooking

Singura aplicație similară din România este SportBooking (Fig. 1), aplicație creată de către un grup de cinci antreprenori din Cluj. Aplicația dezvoltată de către cei cinci antreprenori propune o prezentare a unităților de sport existente în România, oferind pentru fiecare în parte informații privind tarifele practicate precum și condițiile oferite. Însă în momentul actual aplicația are detalii despre unitățile de sport din doar treisprezece orașe din țară.

Aplicația nu oferă posibilitatea unui proprietar al unei unități de sport să-și creeze un cont pe site și să își adauge terenul respectiv. Pentru adăugarea unei unități de sport proprietarul trebui să îi contacteze prin email. În încercarea de a obține mai multe detalii despre procesul de adăugare a unui teren de sport i-am contactat la adresa pe care aceștia o au pe site, însă fără niciun rezultat, nu mi-au răspuns.



Fig. 1 Aplicația SpoortBooking

## Playfinder

Este o platformă disponibilă doar în Marea Britanie care oferă un portal de căutare și rezervare pentru facilitățile sportive. Playfinder (Fig. 2) operează alături de locurile de sport pentru a-și promova mai bine terenurile și sălile de sport și pentru a simplifica procesul de găsire al acestora. Playfinder lucrează cu operatori și săli de sport de toate tipurile și dimensiunile, de la centre și școli comunitare locale până la operatori de agrement, consilii și organe de conducere.

Dacă ești proprietar al unei unități de sport și dorești să fii promovat pe platforma Playfinder, atunci trebuie să iei legătura cu ei. Spre deosebire de SportBooking, Playfinder pune la dispoziție și un număr de telefon pentru contact, pe lângă o adresă de email. La fel ca și în cazul aplicației SportBooking, proprietarul unei unități sportive trebuie sa plătească pentru a fi promovați pe platforma Playfinder. Nu se menționează un preț fix pe platformă, însă este menționat faptul că plata este sigură făcându-se prin intermediul Stripe[[1]](#footnote-1).

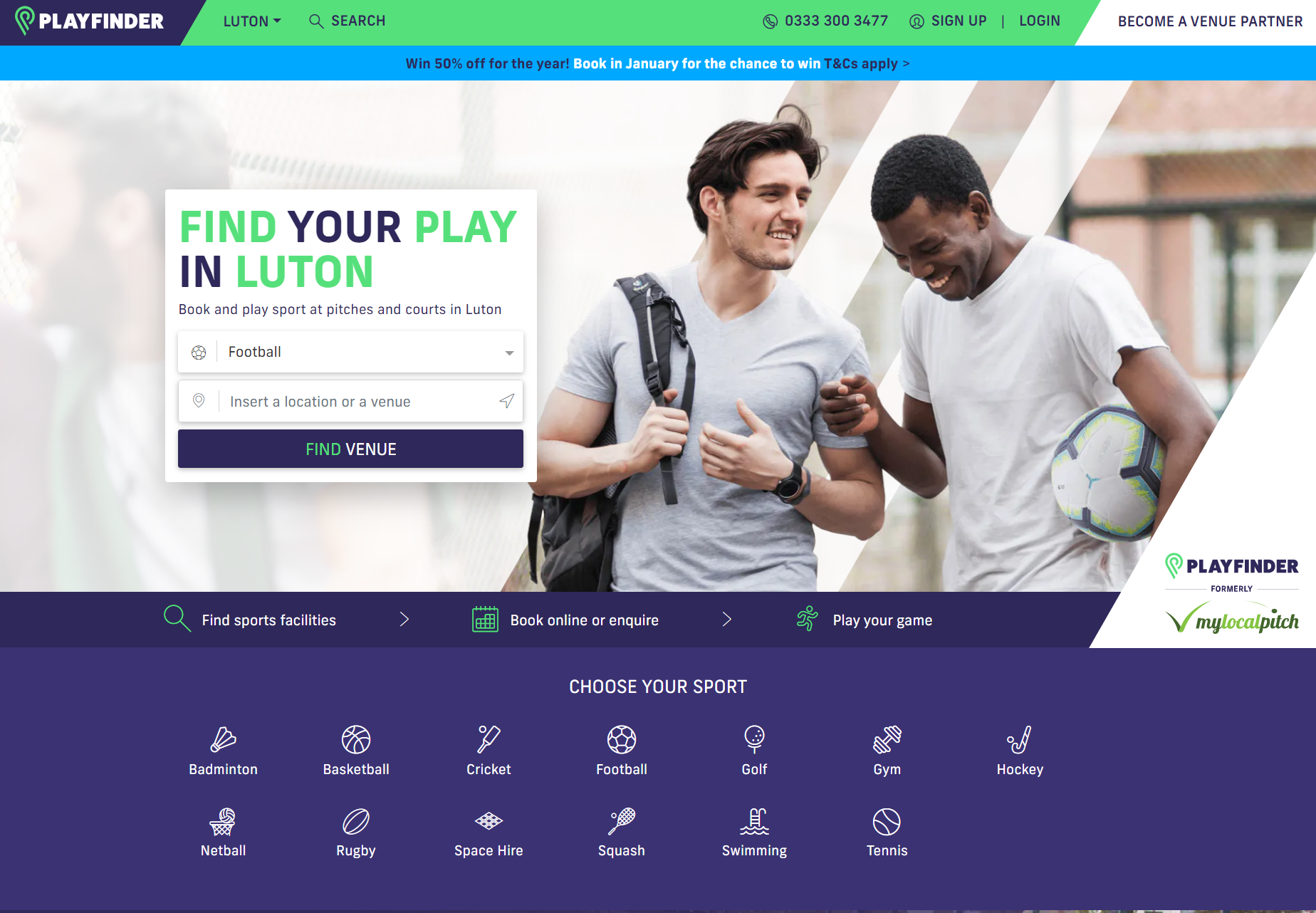


Fig. 2 Aplicația Playfinder

# Tehnologii folosite

Aplicația este alcătuită din două părți. Partea de *front-end* si partea de *back-end*. Partea de *front-end* a fost scrisă folosind React, o bibliotecă JavaScript folosită pentru crearea de interfețe de utilizare.

Partea de *back-end* a fost scrisă folosind Flask, un *micro-framework* scris în limbajul de programare Python folosit pentru crearea de servicii.

Pentru crearea și tranzacționarea asupra bazei de date am folosit SQLAlchemy.

## React

React este o "bibliotecă" JavaScript. Nu este exact un "*framework"*. Nu este o soluție completă, iar de obicei nu oferă o soluție completă pentru rezolvarea unei probleme, fapt pentru care trebuie să folosim mai multe alte biblioteci pentru formarea unei soluții. React nu își asumă nimic despre celelalte părți în nicio soluție.

*Framework-ul* servește pentru un scop mai mare, în special pentru echipele tinere si pentru startup-uri. Când lucrezi cu un *framework*, multe decizii de proiectare sunt deja făcute pentru noi, ceea ce ne oferă o cale liberă în a ne concentra pe scrierea nivelului logic al aplicației. Cu toate acestea  *framework-urile* vin cu niște dezavantaje. Pentru dezvoltatorii cu experiență care lucrează cu programe mari, aceste dezavantaje sunt câteodată foarte supărătoare.

Crearea unei pagini web se face folosind componente. Acestea pot fi componente clasă sau componente funcții. O componentă returnează un cod HTML.

În ceea ce privește componentele clasă există niște metode care fac parte din ciclul de viață al acesteia. O clasă trebuie să extindă obiectul *Component* din React și trebuie în mod obligatoriu să conțină următoarele două lucruri:

* un constructor care primește ca și argumet *props*, primul lucru care trebuie făcut în cadrul acestuia fiind următorul apel *super(props)*, cu ajutorul acestuia se apelează constructorul din clasa *Component* din React. Fără acest apel nu se va putea folosi operatorul *this*.
* metoda *render( )*, această metodă returnează un cod HTML care va fi afișat acolo unde este apelată componenta.

React vine cu o noutate în ceea ce privește componentele funcții. La o conferință din 2018 aceștia au prezentat ceea ce ei consideră că este viitorul în ceea ce ține de React și anume *hooks*. Cu ajutorul acestui concept nou, dezvoltatorii pot folosi *state-ul* cât și alte funcționalități oferite de React fără a scrie o nouă clasă. Printre beneficiile pe care ale aduc *hooks* se află urmatoatele:

* este ușor de lucrat cu ajutorul lor și totodată sunt ușor de testat, iar în ceea ce privește codul, acesta arată mai curat folosind *hooks*
* codul scris cu ajutorul *hooks* este mai ușor de înțeles și conține mai puține linii de cod.
* datorită lor este mai ușoară reutilizarea codului, deasemenea nu se crează allt elemnt în DOM ca în cazul HOCs[[2]](#footnote-2). Cu HOC se separă logica unei stări care nu are legătură cu anumite funcții și se injectează în componenta principală ca și *props*. Deși cu *hooks* se poate rezolva acest lucru ca și cu HOC, acest lucru se face fără haosul creat în urma folosirii unui *wrapper.*

## Flask

Este considerat *micro-framework* deoarece nu necesită unelte sau biblioteci suplimentare. Nu are nici un strat de abstractizare peste baza de date, validator de formulare, sau orice alte componente care oferă funcționalități comune prin existența unei părți terțe. Totuși, Flask suportă extensii care pot adăuga funcționalități ca și cum ar fi implementate în Flask însuși. Există extensii pentru mapări relaționare între obiecte, validare de formulare, diverse tehnologii de autentificare deschisă și numeroase instrumente care au în legătură framework-uri comune. Extensiile sunt actualizate mult mai frecvent decât nucleul programului Flask.

Am ales Flask pentru a scrie partea de *back-end* datorită libertății pe care o oferă acesta. Totodată am ales Python datorită creșterii folosirii acestuia pe piață.

## SQLAlchemy

Este o bibliotecă care facilitează comunicarea dintre programe scrise cu ajutorul limbajului de programare Python și bazele de date. În cele mai multe cazuri biblioteca este folosită ca și instrument de mapare relaționară între obiecte. Instrument care transformă clasele Python în tabele de baze date relaționare și transformă automat apelurile de funcții în instrucțiuni SQL.

SQLAlchemy oferă o interfață standard care permite dezvoltatorilor să scrie cod pentru a comunica cu o mare varietate de motoare de baze de date.

# Funcționalitățile aplicației

Aplicația presupune crearea unei platforme care oferă utilizatorilor fără un cont posibilitatea de a căuta un teren de sport. Utilizatorii care își vor crea un cont sunt acei utilizatori care vor putea să-și adauge o bază sportivă, un teren sportiv sau o bază de sport.

Utilizatorii fără un cont în aplicație vor putea căuta un teren de sport după anumite filtre. Totodată vor putea vizualiza informații despre orice teren de sport care poate fi accesat din pagina de *Home* sau din urma unei căutări. Pe pagina terenului vor putea vedea informații precum: calendarul, datele de contact ale proprietarului, prețul pentru o oră, descrierez, adresa terenului, facilitățile și programul de funcționare al acestuia.

## Înregistrare

Proprietarii terenurilor de sport au posibilitate de a-și creea un cont cu ajutorul căruia pot adăuga un teren. Pentru crearea unui cont, un utilizator trebuie să introducă următoarele date:

* **username**: este unic la nivel de utilizator, necesar pentru autentificarea în aplicație.
* **password**: parola necesară pentru autentificarea în aplicație.
* **email**: adresa de *email* a utilizatorului. Aceasta va fi folosită atunci când un utilizator va adăuga un teren nou. La adăugarea unui teren de sport adresa de email va fi una dintre datele de contact.
* **firstName**: numele utilizatorului.
* **lastName**: prenumele utilizatorului.
* **phoneNumber**: numărul de telefon al utilizatorului. La fel ca și în cazul adresei de email, numărul de telefon va fi adăugat automat unui teren creat de către utilizator în secțiunea *Contact Data*.

Pentru secțiune de înregistrare au fost introduse validări pentru pentru toate câmpurile, niciun câmp nu poate fi lăsat necompletat. În cazul în care utilizatorul lasă un câmp necompletat, acesta nu va putea să își creeze contul, butonul de *Register* fiind dezactivat. Validările pentru fiecare câmp în parte sunt după cum urmează:

* **username**: pe lângă faptul ca nu poate fi lăsat necompletat, acesta trebuie să conțină cel puțin 5 caractere.
* **password**: parola trebuie să conțină cel puțin 6 caractere. Atunci când introduce parola, utilizatorul are posibilitatea să o vadă sau aceasta să rămână ascunsă.
* **email**: adresa de *email* a fost validată folosind o expresie regulată în conformitate cu RFC 5322.
* **firstName**: numele utilizatorului trebuie să conțină cel puțin 3 caractere.
* **lastName**: prenumele utilizatorului trebuie să conțină cel puțin 3 caractere.
* **phoneNumber**: pentru validarea numărului de telefon am folosit biblioteca *validator[[3]](#footnote-3).*

După completarea toate câmpurilor, dacă toate date care au fost introduse sunt corecte din punct de vedere al validărilor butonul de *Register* se activează. După ce este apăsat butonul de *Register* se face un apel către server pentru înregistrare. În urma apelului trei cazuri sunt posibile:

1. *username-ul* este folosit de către un alt utilizator. Utilizatorul rămâne pe pagina de *Register*, iar un mesaj care îl informează să își aleagă un alt *username* este afișat.
2. adresa de *email* este folosită de către un alt utilizator. Utilizatorul rămâne pe pagina de *Register*, iar un mesaj care îl informează să își aleagă altă adresă de *email* este afișat.
3. *username* și adresa de *email* nu sunt folosite de către niciun alt utilizator. Înregistrarea se face cu succes, iar utilizatorul este redirecționat către pagina *Home*, iar de acolo are posibilitate să intre din bara de navigare pe pagina de *Login*. Caz în care utilizatorul se poate autentifica cu succes.

## Autentificare

Se face pe baza de *username* și de parola. Butonul de *Login* este dezactivat în cazul în care cele două câmpuri nu sunt completate. După completarea acestora utilizatorul va putea apăsa pe butonul de *Login*, moment în care se va face un apel către server și se vor verifica datele de autentificare. În cazul în care credențialele sunt incorecte, un mesaj care să sugereze acest lucru este afișat. În cazul în care credențialele sunt corecte, însă utilizatorul nu corespunde de o conexiune la rețeaua de internet, o fereastră cu un mesaj de eroare corespunzător este afișată.

Dacă autentificarea se face cu succes, utilizatorul va primi un *token* care va fi inclus în ulterioare apeluri către server.

## Adăugarea unui teren de sport

Aceasta se poate face doar după ce utilizatorul și-a creat un cont în aplicație si este autentificat. Un utilizator poate să își adauge toate terenurile pe carea acesta le deține, nefiind o limită asupra numărului maxim de terenuri pe care le poate adăuga un utilizator. După ce au fost create, terenurile sunt disponibile în pagina *My Playing Field.* Pentru a adăuga un teren nou de sport, un utilizator trebuie să intre pe pagina *Add Field,* după care să completeze următoarele câmpuri:

* **Title**: titlu terenului de sport.
* **Type**: tipul de sport care se practică pe acel teren.
* **Number Of Players**: numărul maxim de jucători pe care îl suportă acel teren.
* **Price/hour**: prețul pe oră pentru închirierea acelui teren împreună cu moneda în care se plătește acesta.
* **Street**: strada pe care se află terenul.
* **Street Number**: numărul adresei la care se află terenul.
* **Country**: țara în care se află terenul.
* **City**: orașul în care se află terenul, utilizatorul putând alege orașul dintr-o listă cu toate orașele din lume.
* **Region**: regiunea în care se află terenul.
* **Postal Code**: codul poștal aferent adresei.
* **Description**: descrierea terenul. În acest câmp utilizatorul poate adăuga ce detalii dorește, precum ar fi o descriere a sportului care se practică în acea locație, când a fost înființat terenul, ce fel de clienți își dorește, etc. .
* **Image**: utilizatorii au posibilitate ca la crearea unui teren nou să adauge o imagine care va fi folosită la afișarea acestuia .

După ce toate câmpurile au fost completate cu succes utilizatorul poate apăsa pe butonul de *Add Playing Field*, după apăsarea butonului toate datele care au fost introduse vor fi validate, în cazul în care un câmp nu a fost completat, s-au a fost completat incorect utilizatorul va fi informat în privința acestui lucru. Informarea se va face prin înroșirea câmpului completat incorect și apariția unui mesaj ajutător sub acesta. Pe lângă faptul că niciun câmp nu poate fi lăsat necompletat, validările care au fost adăugate sunt următoarele:

* **Title**: acesta trebuie să conțină cel puțin 5 caractere, însă este de preferat ca titlu să fie destul de sugestiv și să conțină mai mult decât 5 caractere.
* **Type**: acesta trebuie să conțină cel puțin 3 caractere.
* **Number of players**: acest câmp trebuie completat cu cifre.
* **Street**: trebuie să conțină cel puțin 5 caractere.
* **City**: acesta trebuie să conțină cel puțin 3 caractere.
* **Region**: trebuie să conțină cel puțin 4 caractere.
* **Postal Code**: validarea codului poștal se face în funcție de țara pe care a ales-o utilizatorul.
* **Description**: acest câmp trebuie să conțină cel puțin 10 caractere.

După ce a fost adăugat un nou teren, acesta este disponibil în pagina *My Playing Fields*, pe pagina *Home*, în rezultatul căutărilor, dar și în secțiunea din partea dreapta a unor pagini, secțiune în care sunt afișate ultimile 5 terenuri adăugate.

Un teren adăugat va conține descrierea, adresa și datele de contact, însă adăugarea facilităților cât și a programului de de deschidere trebuie făcută ulterior de pe pagina terenului.

Din pagina *My Playing Fields*, utilizatorul are acces la terenurile adăugate de către acesta. Pe pagină vor fi afișate toate terenurile proprietarului, fiecare teren va fi un card.

|  |  |
| --- | --- |
| Fig. 3 Teren De Sport din pagina My Playing Fields | După cum se poate vedea în figura alăturată, fiecare card conține un titlu, tipul de sport, prețul pentru o oră, adresa cât și două butoane. Butonul de *Delete* îi permite utilizatorului să își steargă terenul. Odată cu ștergerea cardului din pagina *My Playing Fields* se șterg și toate datele din baza de date despre acesta, date precum ar fi: descrierea, facilitățile acestuia, programul de deschidere, dar și toate programările făcute acestuia. Butonul de *Update* |

permite utilizatorului să actualizeze detalii despre teren.

## Căutare

Un utilizator are posibilitate de a căuta un teren de sport fără a avea un cont pe site. Căutare se face în funcție următorii parametri:

* **Type**: tipul de sport al terenului pe care îl cauți.
* **City**: orașul în care dorești să se afle terenul.
* **Number of Players**: numărul de jucători pe care îl dorești ca terenul să-l aibă.

Căutarea se poate face în funcție de unul dintre acești parametri, sau în funcție de toți. Pentru a face o căutare, utilizatorul trebuie să intre pe pagina *Search*, să-și aleagă parametrii doriți și să apese pe butonul de *Search, a se vedea* Fig. 4.

|  |  |
| --- | --- |
| Fig. 4 Secțiunea de căutare | Fig. 5 Selecție tip de sport |

După cum se poate vede în Fig. 4 fiecare câmp este un *drop-down*[[4]](#footnote-4)*.* Fiecare element conține datele care se află în baza de date la momentul actual, a se vedea Fig. 5. Spre exemplu dacă pentru moment în baza de date există doar tipurile de sport: Basket, Fotbal, Hndbal, Ping Pong, Rugby si Volleyball, atunci când se selecteaza secțiunea *Type* va arăta ca în Fig. 5.

## Adăugare de detalii și actualizare unui teren

După ce un utilizator își adaugă un teren, acesta are următoarele două posibilități prin care poate să actualizeze informațiile despre el:

1. De pe pagina *My Playing Fields*, utilizatorul poate să șteargă un teren sau să actualizeze informațiile acestuia, informațiile pe care le-a introdus la crearea terenului.
2. Din pagina terenului respectiv, utilizatorul are acces la următoarele funcționalități: ștergerea terenului, actualizarea informațiilor acestuia, adăugarea de facilități, adăugarea unui program de deschidere cât și acces la calendarul terenului. Utilizatorul poate adăuga un evenimet nou în calendarul terenului sau poate șterge un eveniment.

## Resetare parolă

Un utilizator care și-a uitat parola are posibilitatea să o reseteze apăsând pe butonul *Forgot Password* de pe pagina de *Login*. După apăsarea butonului, se va deschide o fereastră în care utilizatorul trebuie să introducă adresa de *email* cu care s-a înregistrat și să apese pe butonul de *Reset Password*. În urma acestei operațiuni, utilizatorul va primi un email cu un link care conține un token cu ajutorul căruia se va reseta parola.

Accesând link-ul pe care l-a primit în email, utilizatorul va intra pe o pagină în care va trebui să își introducă noua parola.

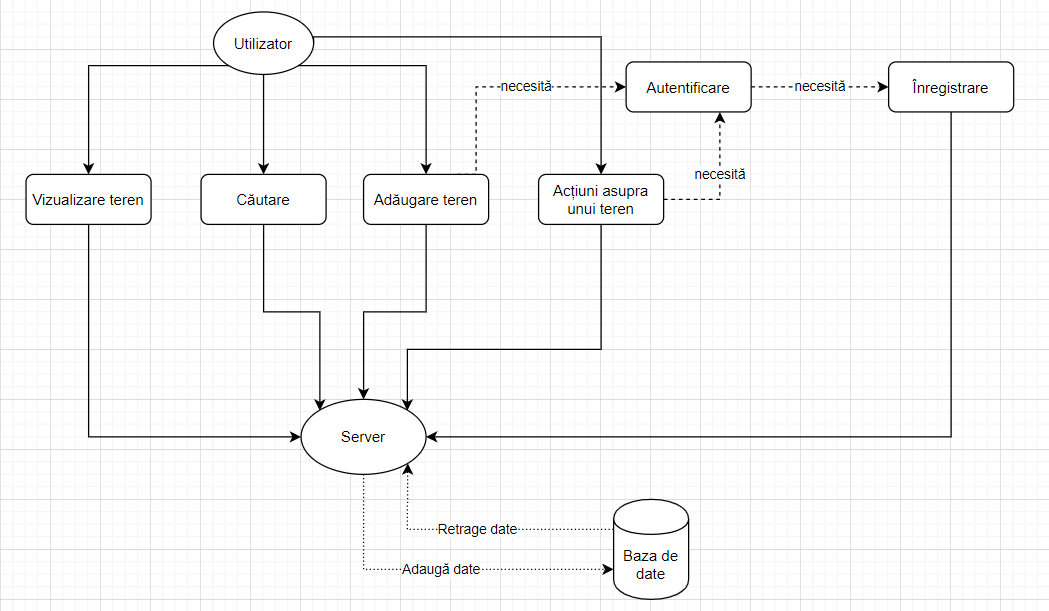


Fig. 6 Structura funcționalităților

# Structura aplicației

Aplicația constă într-un server și un client, serverul fiind creat în Flask, iar clientul în React, structura proiectului pentru partea de *back-end* este după cum se poate vedea în Fig. 8, iar pentru partea de *front-end* se poate vedea în Fig. 7.

|  |  |
| --- | --- |
| Fig. 7 Structură proiect front-end | Fig. 8 Structură proiect back-end |

## Front-end

Partea de client a fost scrisă cu ajutorul IDE[[5]](#footnote-5)-ului WebStorm. După cum se poate vedea și în Fig. 7 partea de front-end este formată din mai multe pachete:

* Pachetul *components* conține toate componentele care au fost utilizate în mai multe locuri.
* Pachetul *containers* conține în principal paginile care sunt încarcate atunci când utilizatorul dorește să ajungă la o anumită cale.
* Pachetul *css* conține fișierele css oferite de către Boots[[6]](#footnote-6)trap.
* În pachetul *main-page* se află pagina principală a aplicației. Pe acestă pagină se află bara de navigare, iar totodată peste această pagină se construiesc celelalte pagini.

Pentru stilizare s-au folosit fișiere css, bootstrap cât și componente oferite de către Material UI[[7]](#footnote-7).

* Pachetul *icons* conține pictogramele care au fost folosite în cadrul apicației precum cea de la *hide/show password*.

Apelurile către server au fost făcute folosind biblioteca *Axios*. Aceasta este o bibliotecă care ne ajută să facem cereri http către surse externe, în cazul meu către serverul *Flask*.

În aplicațiile scrise în React se obișnuiește ca datele să fie preluate de API[[8]](#footnote-8)-uri externe pentru a putea fi afișate în paginile noastre web. O modalitate prin care se poate construi această funcționalitate este folosint API-ul *Fetch* oferit de JavaScript. *Fetch* este capabil de a prelua datele externe, însă are anumite limitări.

O metodă mai populară pentru a face această operațiune este folosind librăria *Axios*. Această librărie este proiectată pentru a gestiona cererile și răspunsurile http.

Este folosită mai des decât *Fetch* deoarece oferă mai multe funcționalități și funcționează și pe browsere mai vechi. Axios se ocupă de răspunsuri folosind *Promises*, astfel încât este simplificat și ușor de utilizat în codul nostru. Axios folosește metode precum *get( )* și *post( )* care efectuează cereri http GET și POST pentru regăsirea sau crearea de resurse. După spusele celor de la Mozilla[[9]](#footnote-9), un obiect de tipul *Promise* este folosit pentru manipularea calculelor asincrone care au câteva garanții importante care sunt dificil de gestionat cu metoda de *callback[[10]](#footnote-10)*.

Pentru crearea paginilor s-au folosit componente. O componentă este o bucată de cod independentă și reutilizabilă. Rolul unei componente este același ca al unei funcții JavaScript, dar funcționează în izolare și returnează o un cod HTML prntr-o funcție de randare. Componentele sunt de două tipuri: componente clasă și componente funcții. În realizarea lucrării s-au folosit ambele tipuri de componente.

## Back-end

Partea de server a fost scrisă cu ajutorul IDE-ului PyCharm. După cum se poate vedea în Fig. 8 partea de back-end conține mai multe fișiere python:

* Fișierul *app.py* conține partea principală a serverului, acolo se află rutele care sunt apelate de către client.
* Fișierele care se termină cu *Model* conțin declarerea tabelelor din baza de date și funcțiile care au fost folosite pentru: crearea, ștergerea, actualizarea sau extregerea unor intrări.

Atunci când un utilizator se autentifică în aplicație se va face un apel din client la ruta */login* cu metoda *POST*, apelul va conține username-ul utilizatorului cât și parola acestuia. Pe partea de server se va verifica dacă username-ul există în baza de date, iar dacă acesta există se va verifica dacă parola pe care a introdus-o este cea din baza de date. Dacă totul este în regulă, username-ul și parola sunt corecte, serverul va trimite un răspuns către client care conține un *token*. Aceste token este folosit pentru a verifica dacă utilizatorul este autentificat sau nu atunci când va apela anumite rute de pe server.

De exemplu pentru a adăuga un teren nou în baza de date, utilizatorul va face un apel la ruta */playingField* cu metoda POST. Apelul va conține toate datele necesare creării terenului, date precum: titlu, tipul sportului, descriere, adresă, etc.. Pe lângă toate aceste date, apelul trebuie să conțină și token-ul obținut în urma autentificării.

Token-ul creat în urma autentificării este un *JSON web token*. Acesta este un standard conform RFC 7519. Crearea lui a fost realizată cu ajutorul bibliotecii *jwt.* Pentu a verifica dacă un apel către server conține token-ul respectiv am creat decoratorul care se poate vedea în Fig. 9. Cu ajutorul acestui decorator se verifică dacă printre parametrii cu care se face o cerere la o anumită rută se află token-ul, în cazul în care tokenul este prezent se încearcă decodarea acestuia cu funcția *jwt.decode*. Dacă token-ul este valid acesta va putea fi decodat, iar cererea către server își va continua execuția, dacă token-ul nu este valid se va arunca o eroare, iar cererea din partea clientului va primi ca și răspuns o eroare.



Fig. 9 Decoratorul cu ajutorul căruia este verificat token-ul

Un token este necesar pentru acele rute de pe server care pot fi apelate doar de către un utilizator autentificat. De exemplu, un utilizator care nu are un cont nu poate șterge un teren din aplicație, un teren putând fi șters doar de către utilizatorul care l-a adăugat.

# Bibliography

Bayer., M. (2006). *The Python SQL Toolkit and Object Relational Mapper*. Preluat de pe https://www.sqlalchemy.org

Bootstrap, T. (fără an). *Bootstrap - The most popular HTML, CSS, and JS library in the world.* Preluat de pe https://getbootstrap.com

Facebook Inc. (2020). *React*. Preluat de pe React - A JavaScript library for building user interfaces: https://reactjs.org

JetBrains. (2000-2020). *PyCharm The Python IDE for Professional Developers*. Preluat de pe https://www.jetbrains.com/pycharm/

JetBrains. (2000-2020). *Webstorm The smartest JavaScript IDE*. Preluat de pe https://www.jetbrains.com/webstorm/

Jones, M. (2015). *RFC 7519 - JSON Web Token*. Preluat de pe https://tools.ietf.org/html/rfc7519

MIT. (2016). *React Validator*. Preluat de pe https://www.npmjs.com/package/react-phone-number-input

MIT. (2020). *MATERIAL-UI: A popular React framework*. Preluat de pe https://material-ui.com

P. Resnick, E. (2008, October ). *RFC 5322 - Internet Message Format.* Preluat de pe https://tools.ietf.org/html/rfc5322

Ronacher, A. (2015). *Flask | The Pallets Projects*. Preluat de pe https://palletsprojects.com/p/flask/

SportBooking.ro. (2015). *SportBooking*. Preluat de pe SportBooking: http://sportbooking.ro

SportMadeSimple. (2020). *Playfinder*. Preluat de pe https://www.playfinder.com/luton

[Fig. 1 Aplicația SpoortBooking 9](#_Toc31468707)

[Fig. 2 Aplicația Playfinder 10](#_Toc31468708)

[Fig. 3 Teren De Sport din pagina My Playing Fields 16](#_Toc31468709)

[Fig. 4 Secțiunea de căutare 17](#_Toc31468710)

[Fig. 5 Selecție tip de sport 17](#_Toc31468711)

[Fig. 6 Structura funcționalităților 18](#_Toc31468712)

[Fig. 7 Structură proiect front-end 19](#_Toc31468713)

[Fig. 8 Structură proiect back-end 19](#_Toc31468714)

1. Stripe este o companie Americană cu sediul în San Francisco, California. Programele dezvoltate de către ei permit persoanelor fizice și juridice să efectueze și să primească plăți pe internet. [↑](#footnote-ref-1)
2. HOC (Higher-Order Component) - o componentă de clasă superioară este o tehnică avansată în React pentru reutilizarea logicii componentelor. Ele nu fac parte din API-ul Reac în sine. Acestea sunt un model care apare din natura compozițională a React-ului. [↑](#footnote-ref-2)
3. validator - este o bibliotecă open-source licențiată de către MIT cu ajutorul căreia se pot valida diverse câmpuri precum: numărul de telefon, adresa de email, etc. [↑](#footnote-ref-3)
4. *drop-down*: un element de control grafic care permite utilizatorilor să aleagă o valoare dintr-o listă. [↑](#footnote-ref-4)
5. IDE - mediu de dezvoltate care pune la îndemana unui dezvoltator un set de programe cu ajutorul căruia codul este scris mai ușor. [↑](#footnote-ref-5)
6. Bootstrap - este un framework css open-source, destinat dezvoltatorilor web. Acesta conține șabloane de design bazate pe JavaScript și css, formulare, butoane, etc.. [↑](#footnote-ref-6)
7. Material UI - framework pentru React care oferă componente cât și teme, framework-ul oferă lucruri gratuit, însă există și lucruri premium pe care le oferă acesta. [↑](#footnote-ref-7)
8. API - un set de definiții de sub-programe, protocoale și unelte pentru programarea de aplicații și software. [↑](#footnote-ref-8)
9. Mozzila - este o comunitate formată în anul 1998 de către membrii Netscape. Cei de la comunitatea Mozzila își propun să păstreze internetul deschis și accesibil pentru toți. [↑](#footnote-ref-9)
10. callback - o metodă de modă veche de a manipula funcții asincrone [↑](#footnote-ref-10)