Titlul lucrării

PROIECT DE DISERTAȚIE

Autor: ing. **Lacatus Robert-Andrei**

Conducător științific: **Titlu. ing. Damian-Mihai**

|  |  |
| --- | --- |
| DECAN  **Prof. dr. ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof. dr. ing. Honoriu VĂLEAN** |

Autor: **Robert Andrei Lacatus**

Titlul lucrării

1. **Enunțul temei:** *O scurtă descriere a temei proiectului de disertație*
2. **Conținutul proiectului:** *(enumerarea părților componente) Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie, Anexe.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, alte locuri dacă este cazul*
4. **Consultanți:** *ing. Prenume Nume (dacă este cazul)*
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declarație pe proprie răspundere privind**

**autenticitatea proiectului de disertație**

Subsemnatul(a) **Lacatus Robert-Andrei** , legitimat(ă) cu CI seria ZL nr. 328943 , CNP 1981201170028,

autorul lucrării: **Lacatus Robert-Andrei**

elaborată în vederea susținerii examenului de finalizare a studiilor de master la **Facultatea de Automatică și Calculatoare**, specializarea **Informatică aplicată,** din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Septembrie 2023 a anului universitar 2022-2023, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de disertație.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de disertație*.

Data Prenume NUME

Robert Andrei Lacatus

(semnătura)

**SINTEZA**

proiectului de disertație cu titlul:

Titlul lucrării

Autor: ing. **Lacatus Robert-Andrei**

Conducător științific: **Titlu. ing. Damian Mihai**

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 2](#_Toc477457095)

[1.1 Context general 2](#_Toc477457096)

[1.2 Obiective 2](#_Toc477457097)

[1.3 Specificații 2](#_Toc477457098)

[2 Studiu bibliografic 3](#_Toc477457099)

[3 Analiză, proiectare, implementare 4](#_Toc477457100)

[4 Concluzii 5](#_Toc477457101)

[4.1 Rezultate obținute 5](#_Toc477457102)

[4.2 Direcții de dezvoltare 5](#_Toc477457103)

[5 Reguli de formatare 6](#_Toc477457104)

[5.1 Formatarea paginii 6](#_Toc477457105)

[5.2 Titluri și stiluri 6](#_Toc477457106)

[5.3 Figuri, tabele și ecuații 7](#_Toc477457107)

[5.3.1 Figuri 7](#_Toc477457108)

[5.4 Tabele 7](#_Toc477457109)

[5.5 Ecuații 7](#_Toc477457110)

[5.6 Referințe bibliografice 8](#_Toc477457111)

[6 Bibliografie 9](#_Toc477457112)

# Introducere

## Context general

In cadrul lucrarii de dizertatie, am dorit sa aleg realizarea unei aplicatii de tip mobile, deoarece telefoanele mobile au reusit sa devina un obiect care au reusit sa ne usureze viata, dar si fata de care ne putem debarasa foarte greu.

Cand ma gandesc la termenul de smart phone sau telefon mobil, imi este foarte greu sa nu am in minte si partea de aplicatii mobile. Multe dintre acestea ne sunt cunoscute si le folosim zi de zi, precum Facebook, Instagram, WhatsApp sau Gmail. Chiar daca pentru multa lume, folosirea lor in exces ne poate afecta, eu cred ca folosindu-le asa cum ar trebui si intr-un scop cat mai informativ si mai educativ, acestea ne pot fi chiar de folos.

Pentru a intra direct in subiect, aplicatia ce am realizat-o este una de tip Airbnb, ce are in componenta sa o harta interactiva, ce va ajuta utilizatorii aplicatiei sa se gaseasca mult mai usor intre ei, sau sa ii redirectioneze mult mai rapid catre un obiectiv turistic ce se afla in apropierea hotelului sau apartamentului unde locuiesc. Ca sa se inteleaga mai bine ideea, voi prezenta doua exemplu. Primul, este cel legat de in care aplicatia este utilizata de un grup de prieteni. In cadrul aplicatiei, se poate creea un mic grup, in care pot participa mai multe persoane, prin intermediul unei invitatii, iar aplicatia, pe baza unui algoritm de calcul din spate, va prelua datele GPS a telefoanelor, iar acesta va arata un punct comun intre persoanele participante la grup, iar un traseu intre pozitia lor si pozitia comuna va fi afisata.

Un al doilea exemplu, este in cazul in care cel ce foloseste aplicatia, va gasi cu usurinta traseul catre obiectivele turistice din apropierea cazarii in care se afla.

De ce am dorit sa fac o astfel de aplicatie? Pentru ca, marea mea pasiune este calatoritul si fotografiatul si stiu cat de greu este sa tot cauti fiecare obiectiv turistic din orasul pe care il vizitezi sau cat de greu este sa cauti o cazare in apropriere de orice zona de interes dintr-un anumit oras. Aplicatiile pe care le cunoastem si sunt cele mai cunoscute, adica Airbnb si Booking, nu ofere informatii concrete despre zona in care te afli si ce obiective turistice se afla in apropierea locuintei tale. Toate informatiile sunt adaugate manual de catre cel ce detine proprietatea, iar de multe ori, nu sunt data exacte, ele fiind informatii false, doar pentru a atrage mai multi clienti.

Aplicatia poate fi folosita cu usurinta in domeniul turismului, fiind de ajutor atat celor ce detin proprietati si vor sa le ofere catre alte persoane, dar si celor ce doresc sa caute un loc de dormit pe parcursul sejurului in acel oras/locatie. Aplicatia nu este creata cu scopul de a avantaja doar una dintre parti, ci pe ambele.

Pe parcursul lucrarii, se vor observa etapele dezvoltarii codului dar si a structurii aplicatiei. Voi discuta despre partea de structura a aplicatiei, despre crearea bazei de date, design-ul facut(UI design) dar si despre codul ce va ajuta la rularea aplicatiei.

## Obiective

Obiectivele principale pentru aceasta lucrare de dizertatie ce are ca scop definirea si realizarea unei aplicatii mobile sunt urmatoarele:

* Realizarea unei data de basa ce va fi ca suport pentru pagina de Login si Register
* Conturile vor fi realizate pentru doua tipuri de utilizatori: Host si Guest
* Pagina de Host va oferi o fereastra unde fiecare utilizator de acest tip, isi poate introduce unitatea de cazare catre inchiriere
* Fiecare pagina cu o unitate de cazare, va contine informatii,poze, locatie si obiectivile turistice din aproprierea locatiei
* Pagina de Guest va oferi o fereastra de cautare a unitatilor de cazare, filtrand dupa locatia unde acesta doreste sa se deplaseze.
* Filtrarea unitatilor de cazare se face prin intermediul unui searchBar, unde utilizatorul de tip Guest, va introduce locatia dorita, iar aplicatia, pe baza codului si a bazei de date, va oferi lista cu unitatile de cazare din locatia respectiva.
* Utilizatorul de tip Guest, are o sectiune de tip Blog, unde acesta, impreuna cu alti utilizatori, pot oferi informatii legate de zonele unde au fost, obiectivele turistice din zona sau restaurantele din imprejurimi.
* Aplicatia va contine o optiune de „Contact Us” unde, in cazul unei probleme, poate contacta persoana support a aplicatiei.
* O ultima optiune a aplicatiei este harta interactiva unde utilizatorul poate creea un grup comun cu alti utilizatori( persoana ce creeaza grupul va putea invita alti utilizatori prin intermediul unui link)
* Odata ce grupul este creat, fiecare utilizator poate vedea pozitia persoanelor pe harta, iar prin cerere, pot afla un punct comun de intalnire, folosind aplicatia, care pe baza algoritmului, va calcula cel mai apropiat si bun punct de intalnire pentru toti si va oferi o imagine a traseului pe care ei trebuie sa il urmareasca.

Intrebarile care m-au ajutat cel mai mult in realizarea aplicatiei au fost urmatoarele:

* Cu ce este aplicatia mea diferita fata de altele ?
* De ce lumea ar prefera sa foloseasca aplicatia mea si nu pe cele existente deja pe piata?
* Cum as putea oferi posibilitatea de identificare a unei persoane prin coordonate GPS?
* Cum poate fi gestionat grupul ce va fi creat de utilizator?

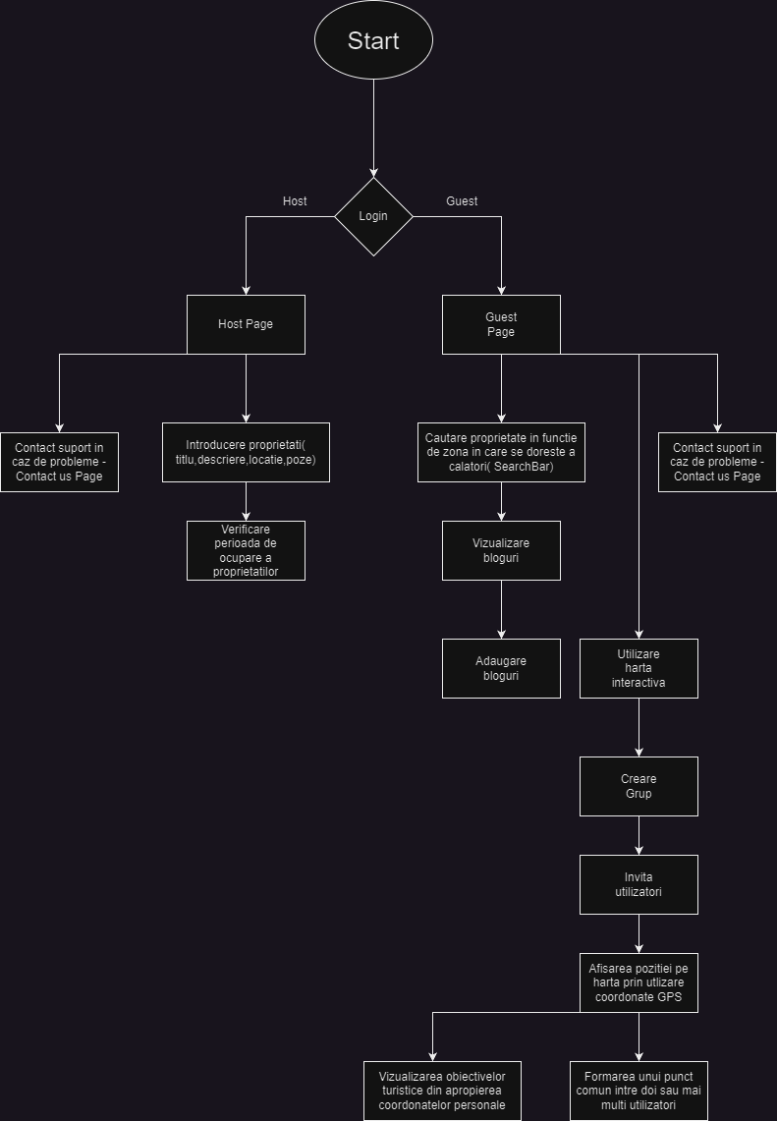
## Specificații

### Realizarea schemei de functionare a aplicatiei

Prin ceea ce am descris mai sus, in capitolul 1.2, tot ce am enumerat mai sus doresc sa obtin. Pentru a reusi sa obtin o astfel de aplicatie, mai intai trebuie plecata de la o structura foarte bine organizata.

Aceasta structura se refera la arhitectura aplicatiei( ex: cum interactioneaza bazele de date intre ele, cat de prietenoasa si usor de utilizat este interfata, care sunt limitarile, daca exista destule resurse de stocare a informatiei etc.). Prima data, am inceput cu ideea de a oferi aplicatiei un scop, acela de a fi o aplicatie folosita in scop turistic si care sa ajuta fiecare utilizator sa gaseasca mult mai usor, obiectivele de interes din apropierea unitatii de cazare sau din orasul pe care acesta il viziteaza.

Dupa ce scopul a fost definit, a trebuit sa ma gandesc la un mod cum va arata aplicatia, pe baza functiilor create.

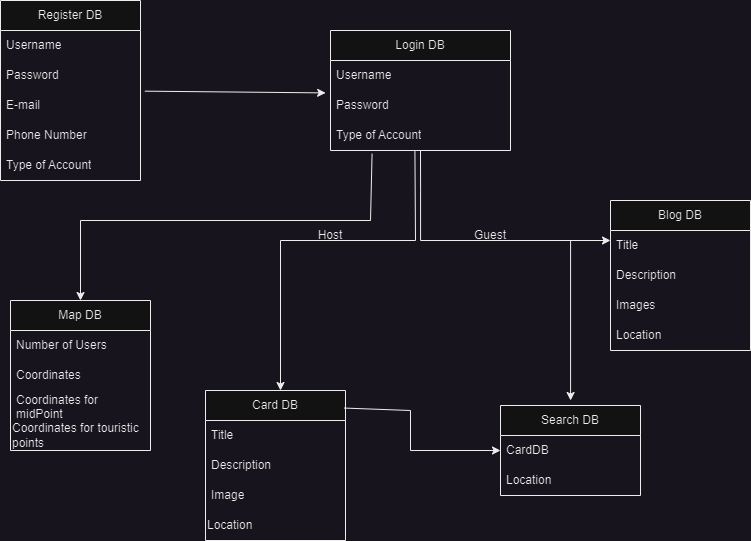


Figură 1 Schema functiilor

Dupa cum se poate observa in Figura 1 este reprezentata schema functiilor ce are ca scop evidentierea mai usoara a modului de functionare a aplicatiei. Intregul proces va pleca din partea de login, unde in functie de tipul contului pe care utilizatorul il are, acesta va avea functionalitati diferite, dar se vor impacta una pe cealalta. Dupa cum se poate observa, contul de tip Host, va putea face doar functionalitati precum „adaugare propriatate” si „revizuire perioada de ocupare a proprietatii”, pe cand cel de tip Guest, va reusi sa „ocupe o proprietate”,” sa scrie un blog”, „sa lase un review in cadrul proprietatii” si „sa creeze un grup pentru partea de harta”. Fiecare functionalitate va fi afisata si explicata pe parcursul lucrarii, impreuna cu secvente din aplicatie ce evidentiaza exact functionalitatea necesara.

### Realizarea schemei aferenta bazei de date

Cunoscand ce functii doresc sa folosesc, trebuie definita o baza de data si cum aceasta va interactiona cu restul aplicatiei.



Figură 2 Schema Database a aplicatiei

Urmarind schema din Figura 2, voi incepe explicatiile cu partea de **Register DB.** In componenta respectiva, utilizatorul isi va creea contul, care va fi ca si informatie de baza pentru restul componentelor. Dupa introducerea credentialelor in pagina de login, baza noastra de data va fi impartita in doua parti, cea pentru **Host** si cea pentru **Guest.**  Fiecare tip de utilizator va avea functionalitati diferite.

Dupa cum se poate observa, baza de date pentru utilizatorii de tip **Host** vor avea salvate urmatoarele informatii: Title, Description,Image si Location. Toate acestea descriu mai exact informatiile necesare pentru creearea unor carduri, carduri ce vor reprezenta modul in care utilizatorul de tip host va putea sa isi puna la dispozitie proprietatile ce doreste sa le ofere spre inchiriere. Fiecare card va avea access catre propria pagina, unde informatiile din baza de date, vor fi folosite pentru a putea contura pagina unde se afla si restul informatiilor.

Pentru utilizatorii de tip **Guest**, vom avea nevoie mai ales, de informatiile de tip locatie si card. De ce? Pentru ca atunci cand cineva doresti sa isi caute un apartament, o va face folosind locatia in care doreste sa ajunga. Dupa ce acesta a introdus localitatea in care doreste sa ajunga, lista cu toate cardurile din zona respectiva vor fi afisate. O alta baza de date pentru acest utilizator, este cea de blog, unde el sau ea vor putea creea propriile bloguri ce vor fi vizualizate si de alti useri, astfel impartasind povestile si experientele lor avute in locatiile respective.

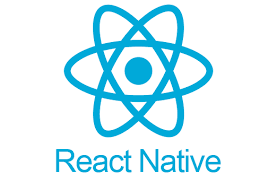
O ultima baza de date este cea folosita pentru partea de harta interactiva, unde utilizatorii pot folosi harta pentru a vedea obiectivele din apropierea locatiei lor, dar pentru a se intalni si cu alti utilizatori. De aceea vom avea nevoie de urmatoarele informatii: Coordonatele utilizatorilor, coordonatele pentru mid point si coordonatele obiectivelor din apropiere.

### Tool-urile folosite pentru realizarea aplicatiei

Pentru a putea duce un astfel de proiect pana la final, au fost nevoie de folosirea mai multor tool-uri, multe dintre ele fiind strange legate mai ales de partea de coding.

In primul rand, voi reexplica structura codului. Codul sursa a aplicatiei este impartit in 3 parti, **Front End**, **Back End** si **Data Base**.

Partea de **Front End** a fost realizata folosind React Native. Acest limbaj, are la baza biblioteci folosite in JavaScript si tehnologia folosita in React. Scopul acestuia este in principal de ajuta persoanele ce doresc sa se dezvolte pe partea de mobile development, deoarece scopul acestuia a fost si va ramane dezvoltarea aplicatiilor pentru iOS si Android. Am preferat sa folosesc React Native deoarece eram deja familiarizat cu React-ul, iar trecerea a fost una foarte simpla si deloc complicata. Daca in React, foloseam <div>,<img> sau <button>, in React Native, avem aceleasi componente, doar sub o alta denumire, precum <View>, <TouchableOpacity> sau <ImageBackground>.



Figură React Native logo

A doua parte, cea de **Back End**, a fost realizata folosind limbajul de programare Python. Multe lume ar fi folosit poate Node.js, dar in cazul meu, am preferat Python, deoarece este un limbaj cu care lucrez zilnic in cadrul companiei de care fac parte. Pentru mine, este un limbaj foarte maleabil si usor de utilizat, mai ales pe partea de Back End. In acelasi timp, am dorit sa il folosesc deoarece are inclus librarii din Flask si MongoDB.

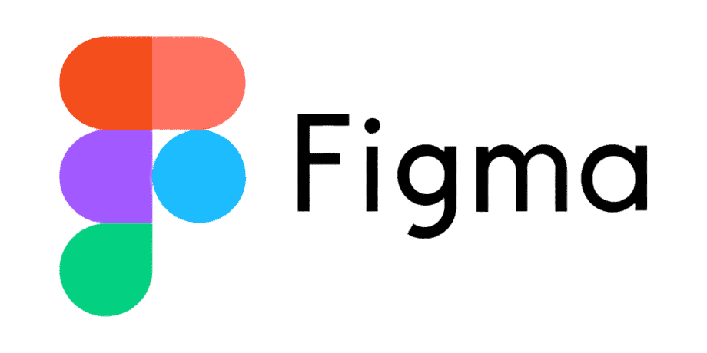
Flask-ul m-a ajutat foarte mult in configurarea si in structurarea codului de Back End si a bazei de date din MongoDB. In acelasi timp, am reusit , prin intermediul Flask-ului, sa conectez partea de FrontEnd cu cea de BackEnd.

Ultima partea a codului sursa este baza de date, pentru care am folosit MongoDB. MongoDB este o platforma dezvoltata in parteneriat cu cei ce au realizat Python si care reuseste sa ofere suport pentru realizarea bazelor de date, ce au ca si back End limbajul de programare Python. In cadrul backEnd-ului, printr-un simplu import a librariei „pymongo” si a scrierii catorva linii de cod, aplicatia creata s-a conectat fara probleme cu baza de date.



Figură 4 Python,MongoDb si Flask logo

Pentru realizarea design-ului am mers de folosirea programului Figma, un tool destul de folosit in domeniul UI/UX , mai ales si in special pentru dezvoltarea grafica a aplicatiilor mobile, atat pentru iOS cat si pentru Android. Aplicatia este una usor de folosit si nu necesita foarte multe cunostiinte pentru a-l folosi si este chiar recomandat folosirea aplicatiei Figma, mai ales pentru proiecte de inceput, fiind si open source si nu necesita platirea unui abonament lunar, precum alte aplicatii ( ex: Adobe Dreamweaver).



Figură Figma Logo

# Studiu bibliografic

Pentru inceput, voi discuta de cum a aparut aceasta idee de inchiriere a apartamentelor/camerelor de hotel folosind aplicatii mobile. Prima oara, ideea a apartinut celor de la Booking, ce au fost la baza o agentie de calatorii din Amsterdam si care au vazut oportunitatea de a creea si de a pune bazele unei aplicatii, ce poate fi usor accesibila pentru toata lumea si care pune la dispozitie oricarui utilizator, o lista cu toate hotelurilor/hostelurilor/motelurilor si resorturilor si carora le pot inchiria apartamentele/camere pentru o anumita perioada, folosind doar telefonul mobil. Aceasta idee a fost foarte revolutionara, ei fiind o perioada si lideri de piata.

Cu trecerea timpului, multi au incercat sa tina piept unui gigant precum cei de la Booking, dar nu au reusit, din cauza faptului ca au prins credibilitate iar lantulurile hoteliere au reusit sa aibe o relatie destul de puternica cu platforma deoarece le puneau la dispozitie clienti din intreaga lume. Acest lucru s-a schimbat, pana cand a aparut AirBnB, o aplicatie ce a venit cu aproximativ acelasi concept dar cateva modificari destul de benefice. Cei din cadrul echipei AirBnB s-au gandit la faptul ca sunt destul de multe persoane ce au mai mult de un apartament, domenii sau mai multe camere ce sunt nefolosite. Prin aceasta aplicatie, ei au oferit posibilitatea unor persoane simple, ce nu tin de un lant hotelier, sa isi puna apartamentele sau camere catre inchiriere, pe o perioada determinata. Lucru care s-a si intamplat. In momentul de fata sunt milioane de persoane ce folosesc aplicatia, atat ca si Host dar cat si ca Guest, iar conceptul a devenit unul de un succes enorm.

In momentul de fata, batalia pentru aceasta piata se tine intre AirBnB si Booking, doua platforme ce incearca sa isi atraga clientii cu cat mai multe optiuni si oferte, care si mai de care atractive.

Dupa ce am citit cateva articole si blog-uri, cel mai bun legat de acest topic, a fost cel publicat de cei de la The World Washere First. Ei au creat un articol intreg unde arata atat punctele slabe cat si cele forte ale ambelor aplicatii.

Pe baza a ceea ce am citit in cadrul articolului, ei au reusit sa ofere mai multe puncte de vedere, atat din punct de vedere a celui ce pune la dispozitie un apartament sau camera de hotel, dar si din punctul celui ce doreste sa inchirieze. Pe partea de inchiriat, AirBnB-ul este mult mai ieftin si mult mai aproape de multe ori de zonele de interes dintr-o anumita locatie. In acelasi timp, din cauza faptului ca ai o interactiune cu „Host-ul” apartamentului respectiv, acesta iti poate oferi mai multe informatii ce tin de imprejurimi precum : unde sa mananci, ce sa vizitezi, cum sa iti petreci viata de noapte si etc. Lucru ce nu se poate intampla la un hotel unde vei inchiria o camera de hotel pentru ca acolo ai o interactiune directa cu membrii hotelului, a carui job este strict de a te ajuta si de a acomoda cu noua camera ce ai inchiriat-o pentru o anumita perioada. Pe de alta parte, AirBnb-ul ofera si o mica nesiguranta pe partea de inchiere deoarece, vei locui in apartamentul unei persoane straine, nestiind nimic despre aceasta, iar un ultimii ani, au inceput sa apara si cateva cazuri stranii cu persoane ce au inchiriat apartamente prin AirBnB si au avut de suferit ( acesta este si un motiv pentru care AirBnB pune la dispozitie un serviciu ce te poate ajuta in astfel de probleme, serviciu ce este mult mai bun decat a celor din cadrul Booking). Booking-ul comparativ cu AirBnB este mult mai sigur, deoarece cel ce calatoreste, se va caza intr-un hotel, unde de multe ori si aproape de fiecare data, este mult mai sigur comparativ cu un apartament a unei persoane straine.( https://www.theworldwasherefirst.com/airbnb-vs-booking-com/)

Pe partea de Host, AirBnB-ul ofera taxe mai mici la fiecare inchiriere a apartamentului si pune la dispozitie o interfata mult mai placuta si usor de utilizat. Booking-ul, fiind putin mai complex si mai strict, vine in avantaj cu faptul ca piata este mult mai larga si iti ofera mai multi clienti comparativ cu AirBnB-ul.

Atat AirBnB-ul cat si Booking-ul au puncte forte cat si puncte slabe, iar asta depinde de tipul de persoana care esti atunci cand calatoresti, de preferintele pe care le ai dar si de bugetul de care dispui.



Figură 6 AirBnB si Booking Logo

Un lucru ce l-am observat in cazul ambelor aplicatii, nici un nu ofera informatii sau o harta, unde sa poti vizualiza mai usor, ce obiective turistice se afla in apropierea locului in care vei sta pentru perioada respectiva. Motivul principal a majoritatii persoanelor ce calatoresc este de a vizita, iar un astfel de „feature” este foarte folositor, mai ales pentru scapa vizitatorul de o munca suplimentara.

Ambele aplicatii ofera doar o harta, creata prin intermediul Google Maps, unde vei putea vizualiza strict unde se afla ceea ce ai inchiriat. Anumite informatii, cum ar fi obiectivele turistice din jur, apar doar in descrierea proprietatii. Un noua caracteristica ce ar putea fi de folosi, ar fi aceea ca, in cadrul hartai, sa apara distantele si traseele ce sunt intre proprietate si obiectivele din imprejurimi. O astfel de functionalitate ar ajuta enorm de mult, iar aplicatia ar putea fi folosita si in timpul calatoriei, nu doar atunci cand aceasta este folosita doar pentru inchirierea unui loc de dormit si atat.

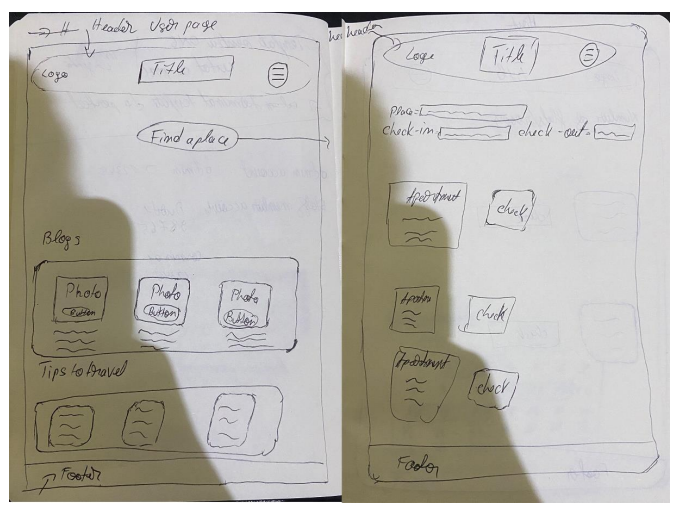
Prin aceasta lucrare de dizertatie, asta este ceea ce am dorit sa demonstrez, faptul ca se poate introduce o astfel de functionalitate, ce poate fi de folosit foarte multor persoane ce doresc sa calatoreasca si sa isi usureze multe pregatiri. Atunci cand cineva planuieste sa plece undeva, intr-o vacanta, pe langa cautarea locului perfect de a dormi, trebuia sa vada si daca au ceea ce vizita in apropierea locului unde se vor caza sau cat de departe este de obiectivele principale din locatia respectiva. Mai pe scurt, sa creeze un scurt intinerariu, care necesita timp si mai bine de o ora de research pentru persoanele respective.

# Analiză, proiectare, implementare

## Inceperea aplicatiei mobile – schitare, design si schema functionalitatii

Odata cu acest capitol, voi incepe sa discut si despre ce am reusit sa realizez pe parcursul dezvoltarii acestei lucrari de dizertatie.

Prima oara, am inceput cu design-ul ce a fost conceput in Figma, aplicatie software ce a fost explicata in capitolul precendent. La inceput, au fost foarte multe modele si foarte multe idei, multe dintre ele, chiar schitandu-le intr-un carnetel, pentru a avea o idee de inceput. In imaginile ce urmeaza, vor fi prezentate mai multe modele ce au ajutat ca si reper pentru dezvoltarea aplicatiei finale.

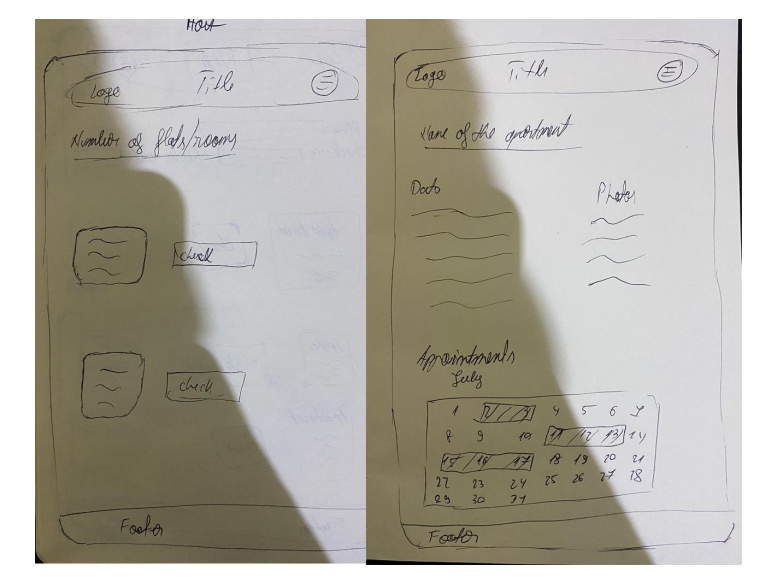


Figură 7 Schita pentru pagina de Guest

In Figura 7 se poate observa o prima schita pentru pagina de tip Guest, unde am dorit implementarea unui blog in pagina utilizatorului( a celui ce doreste sa inchirieze un loc pentru a dormit), blog ce avea diferite categorii. In acelasi timp, acesta ar fi continut si un buton de ‘Find a Place’ unde am fi fost redirectionati la o pagina de cautare(partea din dreapta a paginii) , pagina ce continea cateva dintre proprietatile ce ar fi fost promovate de catre utilizatorii de tip Host.

In Figura 8, este prezentata schita unei pagini de tip Host, unde utilizatorul avea o lista a propriilor domenii ce erau puse la dispozitie utilizatorilor de tip Guest. In partea din dreapta a paginii, se poate cum, odata ce butonul de check era apasat, acesta putea vedea toate datele legate de apartament( ceea ce a introdus el, preturi, poze si perioadele in care acesta era ocupat de cineva).

Chiar daca ideile mereu s-au schimbat pe parcurs, cu timpul, au aparaut si alte schite ce m-au ajutat la conceperea designului final.



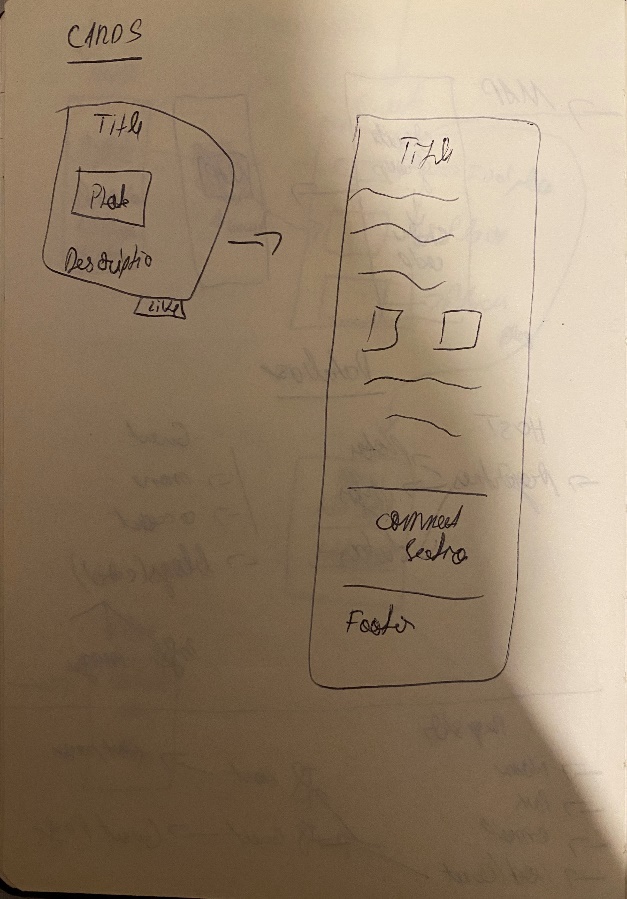
Figură 8 Schita paginii de tip Host

In Figura 9, asa cum este si descris, se poate observa cum a fost prima oara conceputa ideea de card, ce va fi folosit ca si “template” pentru persoanele ce doresc sa introduca proprietati. Prin intermediul acestor carduri, utilizatorii, atat de tip Host cat si cei de tip Guest, pot introduce informatii in cadrul platformei. Pentru a fi mai explicit, cardurile vor ajuta utilizatorii de tip Host sa introduca datele pentru proprietatile pe care acestia doresc sa le puna dispozitie pentru viitorii chiriasi. Pentru cei de tip Guest, cardurile vor ajuta ca un prim “template” de a creea blog-uri ce vor fi vizibile pentru toti utilizatorii platformei.

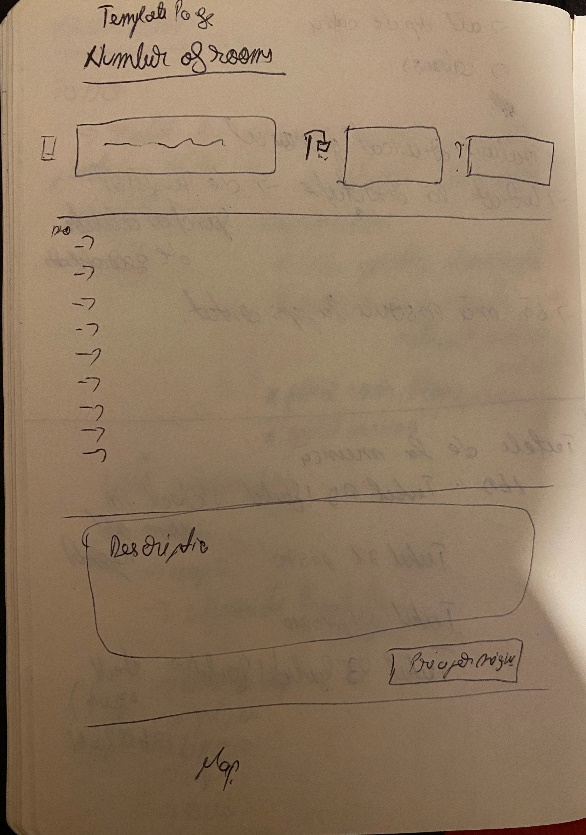
In Figura 10, este evidentiata schita pentru realizarea unei pagini, ce va ajuta utilizatorii de tip Host sa introduca datele legate de proprietatea ce o vor pune la dispozitie. Odata ce cardul creat precendent a fost finalizat, de indata ce acestia vor apasa pe card, o noua pagina va fi afisata. Pagina respectiva ii va ajuta pe viitorii utilizatori sa isi introduca toate datele aferente apartamentului sau unitatii de cazare.

Aceste informatii vor fi urmatoarele:

* Titlul
* Numar camere, bai, paturi
* Utilitati pentru fiecare camera din unitate
* O descriere mai concisa despre locul unde se afla mai exact apartamentului
* Ce obiective turistice are in apropriere de acesta
* Harta interactiva pe care acesta o poate seta pentru a fi mult mai usor de utilizat de catre viitorul chiriasi

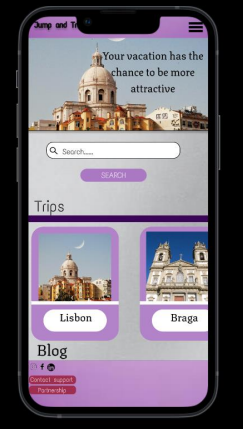


Figură 9 Schita pentru meniul de creare a cardurilor



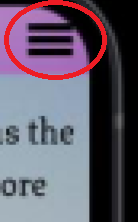
Figură 10 Schita pentru pagina de tip 'Template' a cardurilor

Dupa ce cateva idei au fost schitate cu creionul, am inceput sa incep partea de UI design folosind aplicatia Figma, pentru a reusi sa structurez mult mai usor partea de Front End, ce va intra in interactiune directa cu utilizatorul.



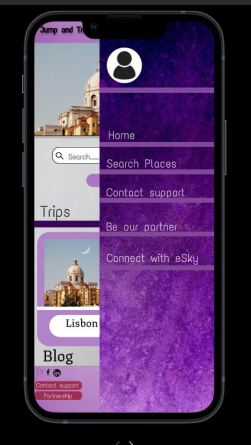
Figură 11 Design Figma- Prima pagina

In figura 11, se poate observa prima pagina realizata in Figma, unde am dorit sa merg pe acelasi principiul precum a celorlalte aplicatii( adica AirBnB si Booking), doar ca , in loc de a pune postari legate de anumite spatii de cazare, am preferat sa adaug anumite bloguri puse de catre fosti utilizatori a aplicatiei, care si-au povestit experientele in mai multe orase pe care le-au vizitat. Dupa cum se mai poate observa, este si un ‘Search’ bar, unde utilizatorul poate cauta locatia dorita.



Figură 12 Buton pentru a accesa meniul

In figura 12, este evidentiat, in chenarul rosu, trei linii negre, ce au ca si functionalitate, deschiderea meniului aplicatiei. Meniu ce contine urmatoarele functionalitati, ce sunt prezentate in figura 13.

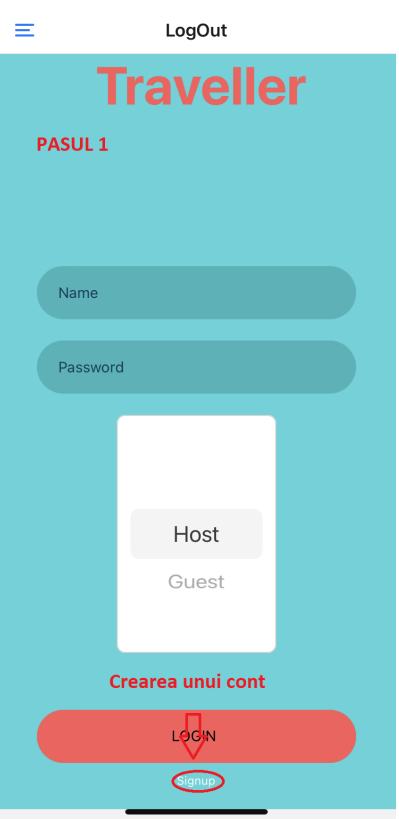


Figură Meniul Aplicatiei

In figura 13, figura contine urmatoarele functionalitati: Home, Search Places, Contact Support, Be our partner, Connect with eSky. Prin folosirea partii de Search Places, acesta ar redirectiona utilizatorul catre o pagina unde, se pot face cautari mai amanuntite despre locul in care doreste sa calatoreasca. Prin partea de contact support, clientul va reusi sa ia contact cu tot ce inseamna customer support, in cazul in care intampina mai multe probleme atunci cand foloseste aplicatia. Be our partner, se adreseaza clientilor de tip Host, ce doresc sa introduca in cadrul platformei diferite proprietati ce pot fi puse spre inchiriere. Ultima functionalitate, cea de Connect with eSky, reuseste sa faca o conectiune rapida intre aplicatie si platforma eSky, unde clientul va putea cumpara si bilete de avion, facand aplicatia mult mai usor de folosit si mult mai complexa, oferind posibilitatea de functionalitati multiple.

Acesta a fost doar un simplu model creat, model ce a ajutat doar pentru a tine cont de anumite functionalitati si cum vor fi aseazate si implementate acestea in cadrul proiectului.

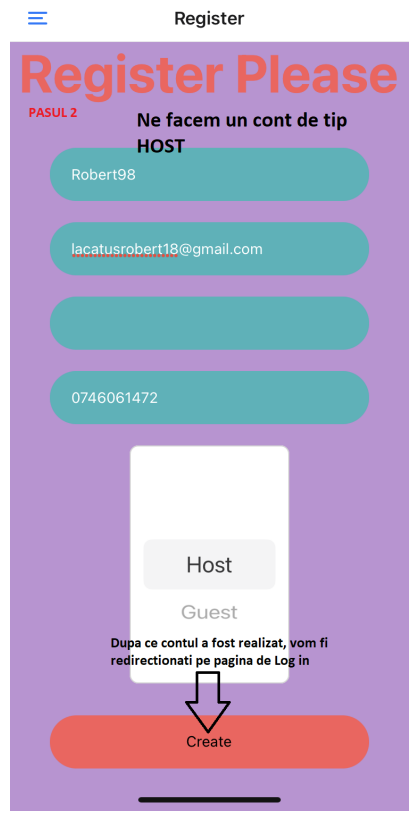
## Explicarea partii de FrontEnd a aplicatiei ( React Native)



Figură 14 Pagina de Login

Primul pas in cadrul aplicatiei, a fost crearea paginii de Login(Figura 14), unde utilizatorul va avea de introdus trei tipuri de date: nume, parola si tipul de utilizator (Host sau Guest). In functie de ce va alege acesta, Host sau Guest, va fi redirectionat pe pagini diferite. Dar mai intai de a se loga, el va trebui sa isi creeze un cont, folosind butonul de “Sign up”, unde va fi redirectionat spre pagina de inregistrare.

In pagina de register (Figura 15), vom observa cele patru campuri, plus un “check box” unde noul utilizator trebuie sa completeze toate campurile si sa aleaga tipul de cont dorit. Dupa ce toate campurile au fost completate, informatia va fi stocata in baza de date creata in mongoDB si dupa poate fi folosita pentru partea de LogIn.



Figură 15 Pagina de Register

Codul pentru partea de Register Page, a fost realizata in file-ul de tip Register.js, unde am creat mai multe functii ce ma vor ajuta in realizarea paginii. Functia ‘handleRegister’ are ca scop, salvearea datelor introduse in baza de date a aplicatiei si de verificare a tuturor ‘inputurilor’, daca acestea au fost sau nu completat. Interiorul functiei, se poate observa ca avem un termen numit ‘axios’. Axios, este o librarie creata special pentru a ne ajuta sa avem o conectiune intre un Front End si Back End, pe baza unui API. In cazul meu, am folosit axios pentru a ma conecta la pagina de Back End, numita ‘/register’. In aceasta pagina, toate datele (phone, username, password, e-mail si optiunea selectata) vor fi salvata in baza de date. Prin urmatorii pasi, adica ‘.then’ si ‘.catch’, functia va verifica daca toate datele au fost completate si daca ceea ce am introdus a fost salvat sau nu in baza de date.

Toate datele au fost salvate, urmeaza ca toate campurile sa fie ‘curatate’, iar informatia ce se afla in interiorul acestora sa fie stearsa pentru ca urmatorul utilizator sa poata sa isi introduca propriile date. Nu ne dorim ca datele recente sa fie vizibile de catre alti utilizatori. Tot acest procedeu a fost realizat prin folosirea functiei onRegister, unde vom seta ca toate campurile sa fie goale.



Figură Functii create in Register.js

In partea a doua a codului (zona de return), am inceput sa folosesc mai multe functii ce apartin librariilor din react, precum :<View>, <ScrollView>, <TextInput>, <Picker> si <TouchableOpacity>. Le voi explica treptat pe fiecare. <ScrollView> ajuta la navigarea printr-o pagina de mobile, scrolland pe telefon. Daca aceasta ar fi lipsit si as fi pus <View>, orice informatie ce se afla mai jos displayul telefonului,nu ar fi fost vizibila. <View> este folosit strict pentru afisa ceva si pentru a defini zonele de lucru din aplicatie. <TextInput> creeaza casute de tip input, unde utilizatorii isi vor putea introduce datele sau informatiile cerute. <Picke> este folosit pentru afisarea unei liste, din care utilizatorul prin intermediul interfetei, poate alege una, in functie de viitoarele actiuni. <TouchableOpacity> este folosit de foarte multe persoane ce lucreaza in acest domeniu pentru a defini un obiect ce poate fi convertit in buton. In loc de a folosi <Button>, in multe cazuri si este recomandat, se foloseste <TouchableOpacity> pentru a defini si alte obiecte din cadrul aplicatiei ca buton. Spre exemplu, daca dorim sa creem anumite carduri, in loc sa le facem fiecaruia butoane aferente, este mai recomandat sa le oferi acest atribut de buton, iar utilizatorul doar va apasa pe card, iar o noua fereastra va aparea.

O ultima parte a codului, este stilizarea acestuia ce sa realizat in constanta ‘styles’, Stilizarea este una obisnuita, ca si in CSS, fiind doar mici diferente de sintaxa.



Figură 17 Register.js - partea a doua a codului

Mergand mai departe in dezvoltarea codului, urmatoarea etapa ar fi logarea cu noul cont. Asa cum am mentionat, vor fi doua conturi separate, fiecare cu functionalitati diferite. Datorita acestui aspect, implementarea paginii de login este strans legata de cea de inregistrare. In figura 18, se poate observa pagina de Login creata, unde utilizatorii trebuie sa isi introduca numele si parola folosita in etapa de inregistrare, dar sa aleaga si tipul de cont pe care l-au creat(Host sau Guest). Alegerea tipului de cont este de tip ‘scroll’, pentru a putea fi mai usor de ales si interactionat cu interfata. Pentru ca aceasta interfata sa functioneze, ea trebuie sa fie conectat constat cu partea de Back End dar si cu baza de date, pentru a putea extrage datele si pentru a confirma crearea contului respectiv. Dupa fiecare logare, indiferent daca a fost una conforma sau nu, utilizatorul va primi un mesaj de tip ‘pop-up’ unde ii se va prezenta statusul logarii( daca aceasta a fost realizata cu succes sau nu).

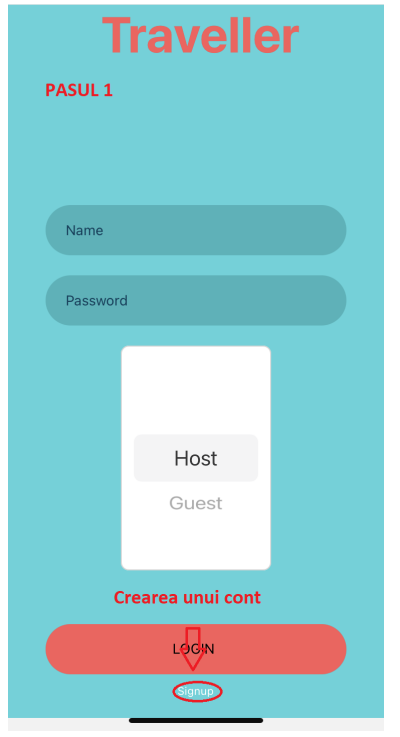


Figura 18 Pagina de Login

Acum, trebuie discutata si partea de cod. In figura 19, se poate observa prima parte a codului. Am preferat sa sar peste partea de explicare a librariilor importate pentru ca sunt folosite in aproape toate fisierele pe care le voi explica in aceasta lucrare. Dupa cum se poate observa, au fost create functii ce ne vora juta pe partea de navigare precum handleBlogs( ce o sa ne redirectioneze catre pagina de Bloguri) sau handleCreateAccount( ne va redirectiona catre partea de Register). Dupa folosindu-ne iar de functia de useState, am declarat mai multe variabile precum : selectedOption, username, password, displayMessage). Toate aceste variabile vor fi comparate cu ceea ce se afla deja in baza de date.

Dupa cum se observa in figura 20, prin intermediul functiei onPressLogin, ne vom conectata la partea de Back End folosind un API(API\_URL in cazul meu), a carei pagina este ‘/login’( fiecare functie din Back End are o ‘pagina’ proprie, astfel evitand greseli ce pot aparea in timpul creararii codului). Prin constanta ‘data’ ii spunem Back End-ului nostru mai exact ce dorim sa cautam in baza de date. Dupa ce aceste legaturi sunt realizate, se pot observa urmatoarele etapa in care verificam daca contul utilizat il regasim in baza noastra de date. In primul If verificam date precum nume si parola, in al doilea daca contul este de tip Guest( daca da, va fi redirectionat pe pagina lui proprie) si in al treaile, vom verifica daca utilizatorul este de tip Host( daca da, va fi redirectionat pe pagina lui proprie). In cazul in care, exista probleme sau neconcordante, in partea de Back End va aparea o eroare ce ne va indica faptul ca contul utilizat nu se regaseste in baza de date



Figura 19 Prima parte a codului pentru pagina de LogIn



Figura 20 Conexiunea dintre pagina de LogIn si BackEnd

In figura 21, mai este vizibila si o parte de catch(error). Aceasta secventa ne va anunta daca conexiunea a avut loc sau nu(daca comunicarea intrea Back End si Front End este functionala). Dupa cum se poate observa, va urma partea de return, unde ne structuram interfata. Diferete mari nu sunt fata de ceea ce am prezentat pentru pagina de register, doar o simpla functie unde este folosita doar pentru aceasta etapa a aplicatiei.



Figura 21 Partea a treia a codului pentru interfata de LogIn

In figura 22, avem functia Picker, ce este importata din libraria “@react-native-picker/picker”. Functia mi-a oferit un ajutor substantial deoarece nu a trebuit sa creez de la zero interfata de tip scroll. Folosind functia, trebuie sa adaug obtiune pe care doresc ca aceasta sa le afiseze, iar in partea de inceput a codului, unde am declarat variabilele necesare, a trebuit sa ofer o valoare de baza. In acest caz, am pus ca valoarea ‘Host’ sa fie valoarea de baza a acestei mici interfete de tip scroll. De fiecare data cand cineva se va loga, aceasta functie va avea valoare ‘Host’ ca si valoare ‘by default’.

Dupa cum se poate observa, apar mai multe functii de tip ‘TouchableOpacity’ pentru a creea butoanele de care vom avea nevoie. Fiecaruia buton i-am asignat prin intermediul onPress, o actiune ce va fi activata dupa apasarea butonului.

In finalul acestei parti de cod, se afla partea de design, pentru a putea structura, aseza si oferi un design anume interfetei.



Figura 22 Partea a 4-a a interfetei de tip LogIn

Presupunem ca ne-am creeat un cont de tip Host. Dupa ce ne-am logat folosind contul recent creeat, am fost redirectionat direct pe pagina principala a utilizatorului de tip Host (dupa cum poate fi observant in Figura 24). Dupa o scurta observatie, am introdus un calendar, unde utilizatorul poate vedea toate perioadele in care apartamentele sau unitatile de cazare sunt ocupate, un buton prin intermediul caruia, acesta poate introduce date in cardul pe care il va crea, card ce va fi folosit ca si intrare in pagina fiecarei unitati de cazare.



Figure 23 Introducere date card

In figura 23, se observa noua fereastra unde va fi redirectionat utilizatorul, dupa ce va apasa butonul “Add Card” din pagina precedenta(figura 24). In aceasta fereastra, utilizatorul va putea adauga numele cardului, o scruta descriere, o locatie(un oras) si o poza care va fi afisata ca imagine principala a cardului. Dupa ce cardul a fost creat, acesta va fi afisat mai jos, in lista utilizatorului. Fiecare utilizator are o lista proprie iar cardurile altor utilizatori de tip Host, nu vor fi vizibile. Acest lucru a fost posibil prin realizarea unei baze de date, un cardul este salvat impreuna cu numele utilizatorului, astfel, programul va sti sa filtreze informatia si va sti daca utilizatorul a creat sau nu carduri si ce carduri ii sunt aferente lui (in figura 24 se poate observa un exemplu de cum este afisat, dupa ce poza a fost introdusa).

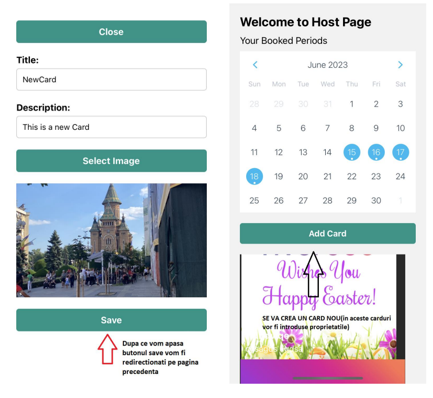


Figure 24 Pagina pentru utilizatorii de tip Host

Finalizand cu partea de Host, voi incepe sa povestesc despre ceea ce inseamna partea de Guest, in cadrul aplicatiei. Odata ce contul de Guest a fost creat in pagina de Register (Figura 25), iar dupa utilizatorul s-a logat cu acesta, va fi redirectionat catre o pagina, specifica utilizatorilor de tip Guest.

In cadrul paginii(figura 26). Utilizatorul poate cauta un loc unde doreste sa innopteze, folosind “Search Bar-ul”. Odata ce locatia a fost introdusa, in cadrul paginii vor fi afisate toate cardurile din locatia respectiva. Asa cum am explicat mai sus, cardurile sunt creeate de catre utilizatorii de tip Host, pentru a-si introduce unitatile de cazare pe care le pun la dispozitie.

Odata ce cardurile au fost afisate, utilizatorul poate apasa pe unitatea de cazare dorita si dupa ii va fi afisata o noua pagina, cu toate informatiile legata de unitatea de cazare respectiva.

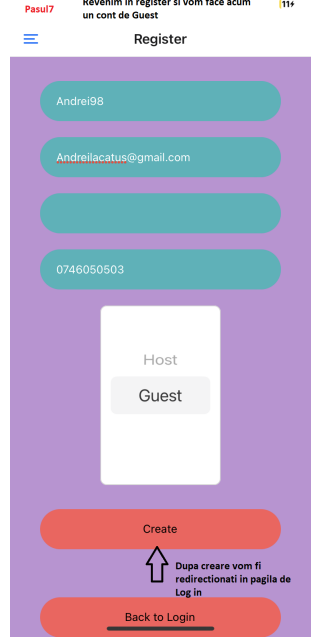


Figure 25 Inregistrarea unui cont de tip Guest

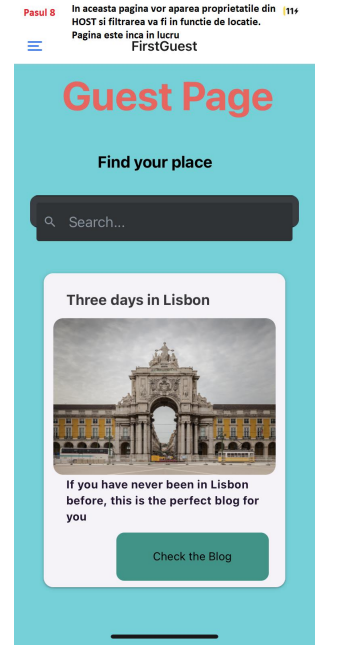


Figure 26 Guest page

Pentru a intelege mult mai bine cum a fost realizata aceasta pagina, voi incepe sa explic si codul implementat.



Figure 27 Prima partea a codului pentru pagina de Guest

Si in aceasta parte de explicare a codului, voi sari peste partea de import, deoarece librariile folosite, sunt la fel precum cele explicate mai sus. In aceasta parte a codului ( figura 27), am creeat functia “handleSearch” unde codul nostrum pentru FrontEnd este conectat la partea de BackEnd. API\_URL va fi acelasi, precum cel folosit mai sus, singurul lucru pe care l-am schimbat este tipul de informatie folosit si ‘pagina’ de BackEnd ce trebuie sa o acceseze( in acest caz, se numeste ‘/search’).



Figure 28 A doua parte a codului pentru pagina de Guest

In figura 28, avem partea vizuala a codului. Pe langa folosirea multor functii de baza, precum “<ScrollView>”,”<View>” si “<TouchableOpacity>”, a trebuit sa folosesc si functia “<SearchBar>”, functie de a fost importata din libraria “'@rneui/themed';”.

In interiorul functiei de “<SearchBar>”, a trebuit sa introduc un “placeholder”, o valoare( in cazul meu searchQuery, ce are valoare default), apoi o valoare de tip onChangeText( unde va fi valoare pe care o voi introduce eu, valoare ce va fi folosita de catre BackEnd sa caute informatiile necesare in baza de date), un containerStyle, un inputStyle si functia care se va apela dupa ce am finalizat de introdus valoarea dorita.

Dupa cum se poate observa in continuare in figura 28, am folosit si functia CardE, functie proprie create, ce ma va ajuta sa afisez cardurile ce doresc sa le caut. Pentru asta, m-am folosit de functia map, unde avem un “key” si un “index”, unde functia, de fiecare data va gasi cardurile pe care le doresc, le va afisa unul cate unul.



Figure 29 Functia Card

In figura 29, am atasat o imagine unde am creat functia Card, ce are la baza o librarie react, librarie numita “react-native-paper”. In acesta, trebuie sa definim cardul, trebuind sa ii creem un spatiu pentru titlu, unul pentru descriere, unul pentru imagine si o ultima componenta, cea de buton. In componenta de buton, trebuie sa oferim urmatoarea actiune, dupa ce butonul a fost apasat.In acelasi timp, a trebuit sa lucrez si la stilul acesteia, pentru a fi unul cat mai usor si placut pentru utilizator.



Figure 30 Parametrii folositi pentru functia Card

Pentru functia de CardE, ce este folosita in pagina de Guest, va fi nevoie anumiti parametri, parametri ce trebuie definiti si in cadrul functiei. Prin acesti parametri, vom putea creea carduri dinamice, unde informatia este introdusa de catre utilizator si extrasa din baza de date si nu este predefinita de catre un cel ce a creat codul.

In continuare, pentru a putea discuta de informatiile afisate in interiorul fiecarui card, va trebui sa revenim la pagina de Host unde, pentru fiecare card creat, utilizatorul trebuie sa introduca informatiile legate de spatiul de cazare pe care il pune la dispozitie. In acelasi timp, in aceasta pagina, el va reusi sa introduca datele locatiei ( coordonate), pentru a afisa pe harta mai bine, locul unde se afla spatiul de cazare. Pe langa aceasta mica harta, acesta va mai avea alte 3, unde va arata si va afisa distantele si traseele catre cele mai apropiate trei destinatii din jurul spatiului de care.

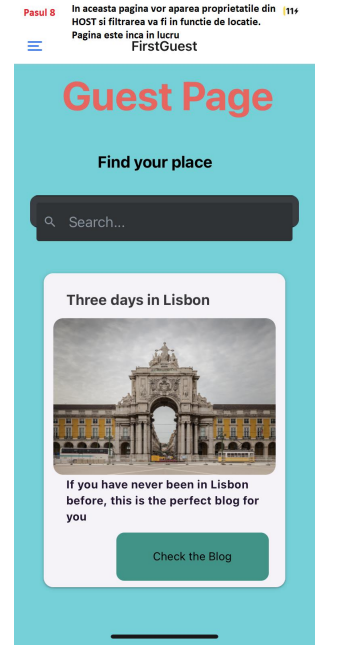


Figure 31 Pagina de adaugare informatii spatiu de cazare

Dupa cum se poate observa, in aceasta pagina, titlul, poza si descrierea scurta, sunt preluate din cardul pe care utilizatorul tocmai ce a apasat. In acest caz, mai intai va fi creat cardul, apoi pagina, pentru ca apoi, informatiile de pe card sa poata fi preluate in pagina unde se vor adauga si restul informatiilor. In continuarea paginii, utilizatorul poate adauga informatiile necesare in sectiunile de tip input ( detalii despre numarul de camere, bai, cate persoane pot locui in spatiul respectiv, dotarile camerelor si descriere generala despre zona in care se afla spatiul de cazare). Dupa ce aceste informatii au fost introduse, utilizatorul va trebui sa introduca coordonatele locului : altitudine si longitutidine, pentru a stabili exact locatia spatiului ce il va inchiria, iar dupa aceea, trebuie sa stabileasca cele trei obiective turistice din apropierea cazarii. Pentru o stabilire mai usoara, acesta va trebui sa mute al doilea pin de pe harta, liber, fara introducere de coordonate. Dupa ce toate informatiile au fost stabilite, ele vor fi salvate in backEnd si vor fi afisate utilizatorilor de tip Guest.

Acum voi trece la partea de cod, unde va fi afisata toata functionalitatea codului ce sta in spatele acestui “window”.



Figure 32 Setarea variabilelor

Pentru inceput, toate variabilele de tip “input”, vor fi setate la inceputul codului. Acestea vor fi setate folosind functia de useState, functie ce am folosit-o de foarte multe ori pe parcursul realizarii dizertatiei. Dupa ce au fost setate, a trebuit sa creez o variabila de tip data, unde am introdus toate variabilele, pentru a putea fi dupa cautate mai usor in baza de date.

Dupa cum se poate observa in figura 33, conectarea se face tot prin intermediul componentei axios, dorind sa cautam toate variabilele stocate in “data”. Ca si in cazurile precedente si in acest caz avem cele doua ramuri unde este verificat daca informatia a fost gasita sau nu in baza de date.



Figure 33 Conectarea cu partea de BackEnd

In figura 34, se poate observa cum am inceput partea de implementare a componentelor vizuale. Aceste componente sunt repititive, functiile folosite fiind unele ce au fost utilizate si in partile de cod precedent explicate. Nimic din ce este prezentat in figura 34 nu este nou, fiind doar cod reutilizat, dar sub o forma diferita si avand un “style componenet” diferit.



Figure 34 Implementarea componentelor vizuale

Dupa ce toate informatiile au fost salvate in baza de date, am dorit sa le si afisez in pagina de tip User, in pagina fiecarei componente de tip Card. Dupa ce utilizatorul a cautat o locatie dorita, ii vor fi afisate toate cardurile pentru locatia respectiva, iar dupa ce acesta va apasa pe unul dintre carduri, o pagina cu toate informatiile legate de acel loc vor fi afisate.

Pentru pagina respectiva, am creeat componenta CardPageG.js, unde toate informatiile despre locatia respectiva vor fi afisate. Dupa cum se poate observa in figura 35( **repune poza**), toate campurile ce au fost completate in “template-ul” utilizatorului de tip Host, au fost afisate aici, pentru a putea prezenta viitorului client, toate informatiile legate de viitoarea lui unitate de cazare.

A screenshot of a computer program

Description automatically generated

Figure 35 Pagina de informatii card – utilizator Guest

In cadrul codului, precum in fiecare alta componenta creata in cadrul proiectului de dizertatie, am dorit sa accesez baza de date, deoarece fara aceasta parte, nu am putea afisa informatiile necesare utilizatorilor.

Functia axios, ramane o componenta principala ce ne va ajuta sa gasim tot ce avem nevoie pentru a afisa in cadrul paginii de tip Guest. Folosind titlul cardului, vom face filtrarea in baza de date, in asa fel incat, sa putem gasi informatia corecta aferenta cardului pe care il accesam( Figura 36).



Figure 36 Cautarea informatiilor in baza de date

Dupa ce informatia a fost filtrata, vom dori sa preluam, tot ce este nevoie. Dupa cum se poate observa, dupa linia “let info = response=[‘data’][0]”, vom cere preluarea a tuturor informatiilor pe care le dorim din colectia “/searchInfo”.

Acum avem toate informatia necesara. Aceasta, dupa va fi afisata, asa cum este prezentat in urmatoarea secventa de cod (Figura 37).



Figure 37 Afisarea informatiilor

In figura 37, se observa cum dupa fiecare componenta **<Text>** introdusa in cod ( Exemplu : “Number of beds:”), avem direct o alta componenta <Text> unde informatia preluata din baza de date este chemata ( Exemplu: **<Text style={styles.describeB2}> {detailsText}</Text>** ). Prin aceasta metoda, toata informatia este chemata si va fi preluata direct din baza de date (informatia fiind si salvata acolo, putand fi chemata de fiecare data cand este nevoie).

Atat in partea de Guest cat si in cea de Host, am introdus componenta de Map, dar la care voi reveni putin mai tarziu, pentru a explica mai exact functionalitatea ei.

In acelasi timp, o alta functionalitate a paginii de tip Guest este introducerea de noi pagini de blog. Fiecare utilizator de acest tip are dreptul ce a introduce un nou blog, unde va introduce informatii legate de o anumita vacanta sau experienta avuta si pe care doreste sa o impartaseasca si cu restul persoanelor iubitoare de calatorii. Dupa cum se poate observa in Figura 39, avem un buton (“Go to Blog”) ce ne va trimite direct catre pagina de Blog.

A screenshot of a computer

Description automatically generated

Figure 39 Pagina Guest

A screenshot of a phone

Description automatically generated

Figure 40 Pagina de Blog

Odata ce am ajuns in pagina de Blog, avem prezentata o lista cu toate cardurile ce au fost create pana acum si care sunt salvate si ele in baza de date. Fiecare are o pagina a lui, unde sunt prezentate toate informatiile necesare despre blogul introdus. In acelasi timp, se poate observa cum in partea de sus a paginii este creat si un buton de “Create a Blog” ce are ca scop deschiderea unei noi ferestre, unde se poate creea cardul necesar pentru viitorul blog ce va introdus in lista.

De indata ce butonul a fost apasat, asa cum am mentionat, utilizatorul va fi redirectionat catre pagina unde poate introduce informatiile pentru noul card ( Figura 41). Informatiile ce pot fi introduse de catre utilizator sunt aceleasi ca si in cazurile precedenta, adica un titlu, o descriere dar si o poza.

Dupa ce aceste informatii au fost introduse si salvate, noul card va fi introdus la finalul listei, fiind ca si o continuare.

Acum, pentru a intelege mult mai bine, cum am realizat aceasta functionalitate, voi explica si codul ce a fost realizat.

A screenshot of a phone

Description automatically generated

Figure 41 Introducere informatii card Blog

Legat de pagina de Guest( partea de cod) am mai prezentat si in partea de inceput a dizertatiei, dar nu si despre aceasta functionalitate. Butonul “Go to Blog” din figura 39, are in spate codul ce este prezentat in figura 42.

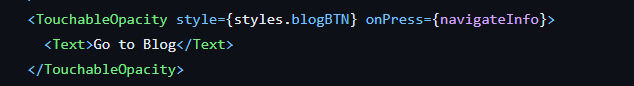


Figure 42 Cod Buton "Go to Blog"

Pentru acest buton, m-am folosit de functia <TouchableOpacity> ce face parte din libraria “react-native”, a carui text, a fost adaugat folosind <Text> . Functia din spate, “navigateInfo”, a fost introdusa prin atributul de “onPress”. In figura 43, este prezentata functia “navigateInfo” ce m-a ajutat sa fac trecerea din pagina de Guest, in pagina de Blog.

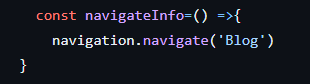


Figure 43 Cod "navigateInfo"

Ajungand in pagina de Blog, ne vom lovi de o functionalitate ce am folosit-o si in partea de introducere a proprietatilor de catre utilizatorii de tip Host. Butonul “Create a blog” are la baza codul ce se afla in Figura 43.

A computer screen with text

Description automatically generated

Figure 44 Functia butonului "Create a Blog"

Prin aceasta functie, utilizatorul va fi redirectionat catre pagina “CreateBlog” unde va urma aceeasi pasi precum utilizatorii de tip Host ce au dorit sa introduca o unitate de cazare in cadrul aplicatiei.

A computer screen shot of code

Description automatically generated

Figure 45 Creare Blog

Dupa cum se poate observa, functia are aceeasi structura ca si in cazul utilizatorilor de tip Host. Va trebui sa se introduca un titlu, o descriere si o poza. Dar pentru a putea realiza aceste etape, va trebui sa explic sa functiile folosite.

In figura 46, vom avea acelasi pas in care dorim sa ne conectam la partea de BackEnd, unde vom introduce titlul, descrierea si poza aleasa, iar functia **axios** ne va oferi ajutor pentru realizarea conexiunii cu partea de BackEnd.

A screen shot of a computer program

Description automatically generated

Figure 46 Conectarea la partea de BackEnd

In figura 47, dorim sa putem insera poze din telefon, ce vor fi afisate dupa in cardul viitorului Blog. Si in acest caz, am folosit functia “ImagePicker” din libraria “expo-image-picker” , functie ce ne va permite conexiune la telefonul utilizatorului si de extragere a pozei cerute de acesta.

A computer screen shot of a program

Description automatically generated

Figure 47 Functia handleSelectPhoto

Dupa ce toate informatiile au fost introduse, acestea vor fi salvate in baza de date, unde vor putea fi utilizate si introduse in partea de Blog. Daca ne vom uita iar in figura 40, vom observa ca avem un buton de refresh, care va introduce noile card-uri de fiecare data ce va fi actionat. In figura 48, functia refreshCard, este cea care ne ajuta cu aceasta functionalitate. Odata ce butonul a fost apasat, el va interactiona direct cu partea de BackEnd ce va extrage toate informatiile necesare din BackEnd. Dupa ce informatia a fost gasita, avem o conditie “**if (Array.isArray(response.data))”**, conditie ce va determina daca raspunsul venit din partea BackEnd-ului este de tip array. Daca raspunsul este unul pozitiv, atunci cardul va putea fi introdus.

A screen shot of a computer program

Description automatically generated

Figure 48 Functia refreshCard

Dupa ce informatia a fost filtrata si gasita, in Figura 49, cardurile vor fi introduse prin functia map, functie ce a mai fost folosita si explicata pe parcursul dezvoltarii lucrarii de dizertatie. Functia va prelua informatiile gasite in baza de date si le va introduce una cate una, la finalul liste.

A screen shot of a computer code

Description automatically generated

Figure 49 Functia map ce va introduce noile carduri

Continuand cu restul componentelor, una ce este folosita in fiecare aplicatie web sau mobile, este cea de “contact support”. In cazul aplicatiei mele ( figura 50), ea este denumita “Contact Us”. Scopul acesteia este de a intra in legatura directa cu personalul de tip “staff” a aplicatiei, ce va trebui sa ajute in rezolvarea problemelor intampinate de utilizatorii aplicatiei.

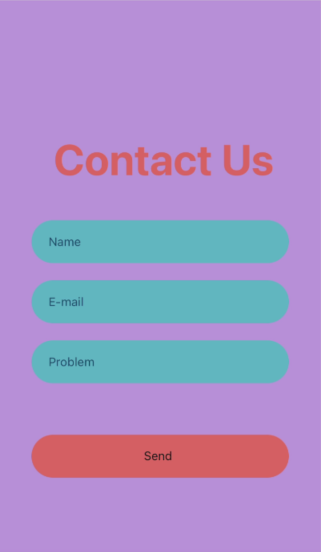


Figura 50 Pagina "Contact Us"

In cadrul acestei pagini, utilizatorul va trebui sa completeze toate cele trei campuri ( nume, e-mail si problema pe care a intampinat-o). Dup ace butonul “Send” a fost apasat, cei din cadrul “staff-ului” vor primi un e-mail cu problema pe care utilizatorul a intampinat-o.



Figura 51 Functiile create ce vor fi utilizate in pagina de Contact Us

Codul folosit pentru aceasta pagina este unul ce a mai fost refolosit si in alte sectiuni ale dizertatiei. Avem prima parte in care in functia “onPressContact” avem conexiunea cu partea de BackEnd si baza de date, deoarece dorim ca informatiile sa fie stocate in baza de date. Tot in cadrul “onPressContact” am dorit sa ma folosesc de functia “if” pentru a putea curata toate input-urile la fiecare apasare, dupa ce campurile respective au fost completate de catre utilizator( Figura 51).

In figura 52, se poate observa cum am creat “input-urile” in care informatiile vor fi introduce. Am folosit acelasi cod pentru introducerea datelor clientilor

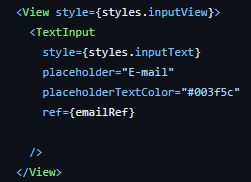


Figura 52 Folosirea functiei TextInput



Figura 53 Crearea partii vizuale a codului

## Explicarea partii de BackEnd a aplicatiei ( Python)

Dupa ce am explicat toata partea de FrontEnd a aplicatiei dezvoltate, partea care este vizibila si care interactioneaza cu utilizatorii, acum voi explica si partea de BackEnd, cea care ruleaza toate functionalitatile aplicatiei.

In partea de BackEnd, asa cum am explicat si in partea de inceput a lucrarii de dizertatiei, voi face toate functionalitatile aplicatiei, folosind limbajul de programare Python. Mai exact, tot ce presupune rularea de informatie, filtrare de utilizatori, salvarea de date, logarea si inregistrarea.

In cadrul proiectului, am folosit un singur fisier .py ce va fi rulat de fiecare data impreuna cu intreaga aplicatie. Fara rularea acestei aplicatii, utilizatorul nu va putea folosi aplicatia, deoarece din start, nu se va putea loga iar contul sau va aparea ca neexistent sau gresit. Pentru a se intelege mult mai bine, partea de BackEnd, salveaza fiecare informatie ce este introdusa in cadrul partii de frontend, stocand totul in baza de date si fiind folosita la fiecare comanda utilizata in partea de FrontEnd.

In urmatoarele randuri, voi prezenta cateva exemple de functionalitati, ce depind constant de partea de BackEnd. Un prim exemplu este partea de login si register. Fiecare cont ce este creat prin intermediul paginii de register, va fi stocat in baza de date. Folosind un API, BackEnd-ul va primi un semnal din partea FrontEnd-ului, prin intermediul caruia informatia ce va fi trimisa prin intermediul API-ului trebuie salvata in baza de date. Dupa ce contul a fost stocat in baza de date, va trebui ca utilizatorul sa se logheze cu contul recent creat. Atunci cand un utilizatorul doreste sa se logheze cu noul cont, acesta va introduce datele (nume, parola si tipul contului), FrontEnd-ul va trimite o cerere prin API catre BackEnd sa caute datele respective in baza de date, iar asa, contul este verificat si se va sti tipul de utilizator si ce informatii are acesta inregistrat in propriul cont.

Un alt exemplu de ce partea de BackEnd este una foarte important si una ce trebuie sa existe in fiecare aplicatie, este partea de cautarea unitatilor de cazare. Atunci cand un utilizator de tip Guest, doreste sa caute un spatiu de cazare, acesta va trebui sa faca o filtrare pe baza locatiei unde va dori sa se cazeze. Inca odata, partea de FrontEnd va trimite aceasta informatie ( adica numele locatiei) catre BackEnd, iar partea de BackEnd va trebui sa caute in baza de date, toate cazarile ce au ca si locatie acel oras ( spre exemplu Lisabona). Dupa ce informatiile au fost gasite, BackEnd-ul va trimite tot ce a gasit catre partea de FrontEnd, unde vor fi afisate pe rand sub forma unui card, ce poate fi dupa accesat.

In figura 3.3.2, se afla schema in care, este afisat cum interactioneaza fiecare componenta BackEnd cu componentele FrontEnd. Dupa cum se poate observa, pentru fiecare componenta complexa este legata de o “pagina” de BackEnd, ce o va folosi pentru a salva informatia in diferite colectii a bazei de date.

In baza de date Mongo, am creat mai multe colectii, pentru ca informatia trebuie sa fie sectionata pe capitole. Prin aceste capitol ma refer la faptul ca fiecare tip de informatie trebuie sa aibe propria ei colectie. Spre exemplu, datele pentru inregistrare, nu pot fi in aceeasi colectie cu informatiile legate de unitatea de cazare, deoarece partea de BackEnd va da eroare deoarece nu poate gasi informatiile pe care le dorim, pe baza filtrului pe care l-am ales.

Odata ce baza de date a gasit structurata, a trebuit sa fac si o conexiune intre FrontEnd si BackEnd, asa cum am spus pana acum. Aceasta legatura se va face prin intermediul unui API. In figura 3.3.1, am declarant API-ul pe care v-a rula BackEnd. Pentru a intelege mai bine cum functioneaza acest API, asa cum se observa si in figura, acesta este reprezentat de un IP ( in cazul meu, in zona portocalie este IP-ul folosit pentru localhost) si un Gateway ( in cazul meu, cel default, de 5000). Atat timp, cat file-ul de python ruleaza pe acelasi IP, FrontEnd-ul si BackEnd-ul pot comunica. In FrontEnd, acesta este introdus in partea de **axios** asa cum am prezentat si in alte parti de cod ( spre exemplu, figura 20).

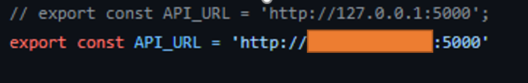


Figura .3.1 Declararea API-ului

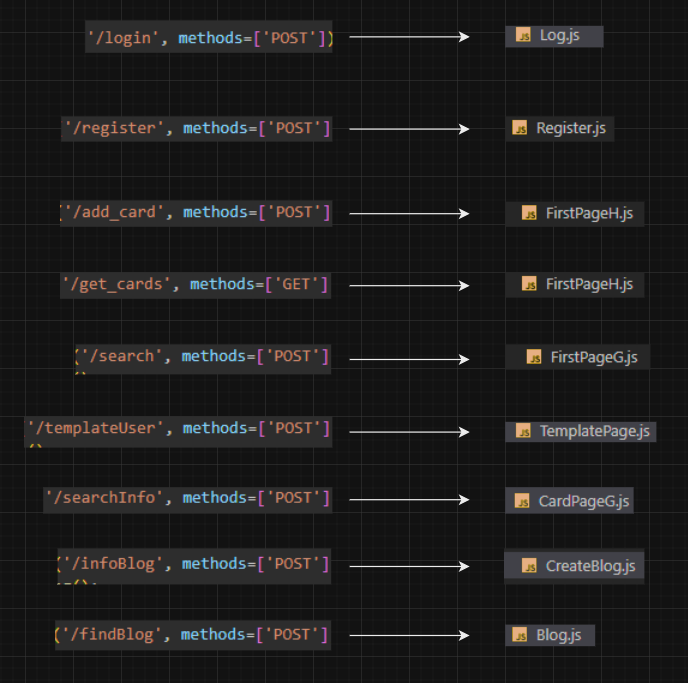


Figura .3.2 Schema conexiunilor intre BackEnd si file-urile de FrontEnd

# Concluzii

## Rezultate obținute

Evidențiați toate rezultatele pe care le-ați obținut și trageți concluzii din ele. Puteți prezenta o analiză critică a ceea ce ați realizat comparativ cu alte lucrări/studii anterioare.

Includeți o listă a contribuțiilor pe care le-ați avut în domeniul temei abordate.

## Direcții de dezvoltare

Descrieți direcțiile posibile de dezvoltare.

# Reguli de formatare

## Formatarea paginii

* + Dimensiunea paginii: A4
  + Margini: 2.5 cm (sus, jos, stânga, dreapta)
  + Antet și subsol: 1.27 cm de la marginea paginii
  + În antetul paginii (header): titlul capitolului, centrat, stil: Header\_style
  + În subsolul paginii: numărul paginii, centrat

## Titluri și stiluri

Titlurile capitolelor și subcapitolelor se marchează cu stilurile Heading 1 – 4, conform documentului model anexat în format Word. Descrierea stilurilor utilizate în document este prezentată în Tabelul 5.1.

Tabelul 5.1. Stiluri utilizate în acest document

| Nr. | Stil | Utilizat pentru | Format |
| --- | --- | --- | --- |
| 1 | Normal | Text normal | Font: (Default) Cambria, 12 pt, Justified, Line spacing: Multiple 1.1 li, Space After: 6 pt |
| 2 | Titlu | Titlul proiectului, prima pagină | Font: 24 pt, Small caps, Centered Line spacing: single, Space Before: 126pt, After: 0 pt, |
| 3 | Titlu2 | Titlul proiectului, pagina de prezentare | Font:14pt, Bold, Centered |
| 4 | Heading 1 | Titlurile capitolelor (nivel 1) | Font: 24 pt, Indent: Left: 0 cm Hanging: 0.76 cm, Space Before: 24pt, After: 12pt |
| 5 | Heading 2 | Titlurile subcapitolelor (nivel 2) | Font: 14 pt, Bold, Indent: Left: 0 cm  Hanging: 1.02 cm, Space Before: 18pt, After: 12pt |
| 6 | Heading 3 | Titlurile secțiunilor (nivel 3) | Font: Bold, Indent: Left: 0 cm Hanging: 1.27 cm, Space Before: 6 pt, After: 6pt |
| 7 | Heading 4 | Titlurile secțiunilor (nivel 4) | Font: Italic, Indent: Left: 0 cm Hanging: 1.52 cm, Space Before: 2 pt, After: 0 pt |
| 8 | Caption | Legenda figurilor și tabelelor | Font: Italic, Font color: Text 1, Line spacing: single, Space After: 10 pt, |
| 9 | Header\_style | Antetul paginii | Font: 10 pt, Italic, Centered, Border: Bottom: (Single solid line, Background 1, 0.5 pt Line width) |

## Figuri, tabele și ecuații

### Figuri

Figurile se inserează în text centrate, cu etichetă de numerotare și legendă (Caption) în partea de jos a figurii. Numărul figurii include și numărul capitolului, după exemplul prezentat în Figura 5.1.



Figura 5.1. Figură exemplu, stil: Caption

## Tabele

Tabelele se inserează în text centrate, cu etichetă și legendă (Caption) în partea de sus a tabelului, aliniată la stânga. Numărul tabelului include și numărul capitolului, după cum este prezentat, de exemplu, în Tabelul 5.1.

## Ecuații

Ecuațiile se inserează în text centrate, cu numerotare în partea dreaptă. Numărul ecuației include și numărul capitolului, conform exemplului din relația (5.1).

|  |  |
| --- | --- |
|  | (5.1) |

## Referințe bibliografice

Se recomandă ca citarea referințelor bibliografice să fie făcută în formatul IEEE.

În secțiunea Bibliografie sunt prezentate exemple pentru: o citare a unui capitol dintr-o carte [1], un articol publicat într-o revistă [2] și un articol publicat la o conferință [3].

Detalii cu privire la formatul citării diverselor tipuri de referințe pot fi găsite în [4] sau [5].

Referințele bibliografice se pot insera în text utilizând facilitățile Word de a adăuga surse și bibliografie unui document (References -> Citations & Bibliography). Dacă formatul IEEE pentru bibliografie nu este instalat implicit în Word, se poate descărca gratuit de la:

<https://bibword.codeplex.com/wikipage?title=Styles&referringTitle=Home>

Instrucțiunile de instalare pentru diferite versiuni de Word se pot obține de la aceeași adresă.

# Bibliografie

|  |  |
| --- | --- |
| [1] | P. Nume, "Titlul capitolului," în *Titlul cartii*, Oras, Editura, 2016, pp. 1-24. |
| [2] | P. Nume, "Titlul articolului," *Titlul revistei,* vol. 1, no. 2, pp. 22-30, 2016. |
| [3] | P. Nume, "Titlul articolului," în *Numele conferintei*, Oras, 2015. |
| [4] | "IEEE Citation Reference," 2009. [Online]. Available: https://www.ieee.org/documents/ieeecitationref.pdf. |
| [5] | "IEEE Editorial Style Manual," 2016. [Online]. Available: https://www.ieee.org/documents/style\_manual.pdf. |