Universitatea POLITEHNICA București
Facultatea Automatică şi Calculatoare
Departamentul Automatică şi Informatică Industrială

Platformă integrată de comunicare și planificare a deplasărilor cu scop turistic

Coordonator
SI. Dr. Ing. Adriana Olteanu

Absolvent
Diana-Cristina Cocea

2018

Cuprins

1. Introducere 3
1.1 Structura documentului 3
1.2 Obiectivele proiectului 4
2. Prezentarea aplicațiilor din același domeniu 5
3. Prin ce se diferențiază aplicația YourGuide 7
4. Tehnologii web 9
4.1 Instalare pachete 9
4.2 Interfața aplicației 10
4.3 Backend-ul aplicației 14
5. Implementarea și funcționalitatea aplicației YourGuide 25
5.1 Logare 25
5.2 Înregistrare 27
5.3 Recuperare parolă 29
5.4 Schimbare parolă 30
5.5 Alegerea destinației 31
5.6 Completarea informațiilor de către ghizi 32
5.7 Chatbot 36
5.8 Chat 41
5.9 Căutare utilizatori 43
5.10 Profil utilizator 43
6. Concluzii și dezvoltări ulterioare 47
6.1 Concluzii 47
6.2 Dezvoltări ulterioare 47
7. Bibliografie 49

1. Introducere

Se poate spune, pe drept cuvânt, că trăim într-o societate cu totul informatizată. Acest lucru se datorează faptului că ne dăm seama din ce în ce mai mult de faptul că sistemele de calcul ne ușureaza munca, viața prin apariția și dezvoltarea lor. În ultimii ani, aplicațiile desktop au început să fie tot mai puțin folosite, tehnologiile dezvoltându-se foarte mult în zona web. Cel mai evident avantaj al unei aplicații web este flexibilitatea acesteia. O aplicație web poate fi accesată de oriunde, folosind orice device conectat la internet. Acest lucru se datorează localizării aplicației pe un server, care poate fi accesat dacă există conexiune la internet.

1.1 Structura documentului

Primul capitol prezintă câteva noțiuni introductive referitoare la problema dată și obiectivele proiectului.

În capitolul 2 sunt prezentate alte aplicații cu aceeași temă, evidențiindu-se elementele noi pe care proiectul de față le aduce.

În capitolul 3 este prezentată ideea generală a soluției problemei pe care o propune proiectul. Se realizează un chestionar completat de persoane de vârste diferite, pe baza căruia se va implementa chatbot-ul.

În capitolul 4 sunt analizate tehnologiile folosite pentru implementarea interfeței aplicației și a logicii din spate. Vom vorbi despre alegerea tehnologiilor, framework-ul React pentru partea de front-end, dar și motivul pentru care se combină atât de bine cu Firebase, serviciul pentru centralizarea datelor.

În capitolul 5 se descrie funcționalitatea exactă a aplicației, fiecare pas pe care utilizatorul trebuie să îl facă pentru a putea folosi acest serviciu. Pe lângă descrieri, fiecare pas va avea asociate imagini din aplicație, pentru a se putea ilustra cât mai bine

funcționalitatea. Capitolul va descrie totodată procesul realizării aplicației. Fiecare etapă importantă din realizarea proiectului va avea exemple de cod, pentru o mai bună înțelegere. Vor fi, de asemenea, specificate motivele pentru care s-au ales soluțiile, logica, ordinea.

Capitolul 6 reprezintă partea concluziilor. Aici vom face un rezumat al funcționalităților, concluzionând cu ajutorul real pe care aplicația îl oferă utilizatorilor la problema care există în momentul actual. Vom discuta, de asemenea, și despre dezvoltările ulterioare pe care proiectul urmează să le aibă.

Capitolul 7 este și cel final, aici este prezentată bibliografia acestui proiect.

1.2 Obiectivele proiectului

Există o varietate foarte mare de aplicații, având diverse funcționalități care vin în ajutorul utilizatorilor. După o cercetare asupra aplicațiilor existente până în momentul de față, am descoperit faptul că oamenii încă au „probleme” la capitolul călătorie, indiferent de natura acesteia, spre exemplu, alegerea unui restaurant pentru servirea mesei, găsirea unui loc de relaxare, găsirea cazării potrivite. Toate aceste locuri nu sunt atât de ușor de găsit, deoarece oamenii au păreri și gusturi diferite, din acest motiv, părerile de pe un forum sau de pe diverse site-uri de promovare nu sunt întotdeauna relevante.

Aplicația „Ghidul tău” vine în ajutorul oamenilor prin găsirea soluțiilor prin cea mai simplă metodă – comunicarea dintre persoane. Lucrarea de față încurajează interacțiunea dintre oameni, deoarece ghidul din numele proiectului reprezintă, de fapt, chiar o persoană reală care vine în ajutorul utilizatorului. De asemenea, cel care folosește aplicația va putea să își definească itinerariul pe care l-a parcurs, adăugand poze, comentarii folositoare pentru alți utilizatori cu aceeași destinație.

Obiectivele aplicației:

- Găsirea ghidului ideal cu ajutorul chatbot-ului
- Comunicarea dintre utilizator si ghid
- Definire itinerariu

2. Prezentarea aplicațiilor din același domeniu

Există diferite aplicații în același centru de interes, aplicații care, însă, nu îmbină toate elementele necesare pentru ca utilizatorul să aibă parte de o experiență completă.

Aplicația “Flip the Trip” are puncte comune cu aplicația dezvoltată în această lucrare, obiectivul ei general fiind acela de a face legatura între persoane care au interese comune, doar din punct de vedere al călătoriilor. Persoanele pot comunica, îsi pot face propriul profil, pot adăuga poze si informații personale. Spre deosebire de aplicația de față, “Flip the Trip” nu are niciun mecanism prin care să se facă o legatură automată între utilizatori. Aplicația poate să se folosească, de asemenea, de datele utilizatorului, în cazul în care acesta are profil de Facebook și/sau Instagram.

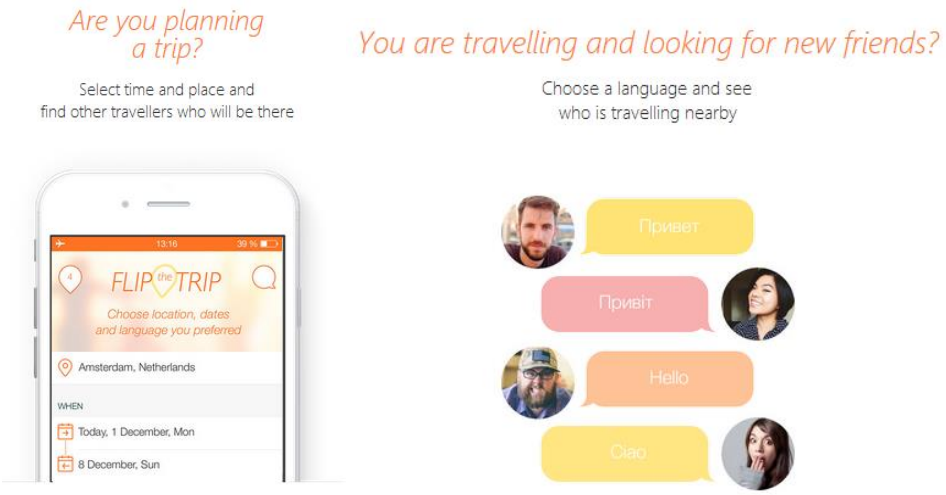


Fig 2.1. Pagina de prezentare a aplicației “Flip the Trip”

O altă idee de aplicație este “Ask a Stranger”, care, după cum îi sugerează și numele, încurajează turiștii să comunice cu persoanele necunoscute din aceeași zonă în care se află în momentul de față. Aplicația oferă puncte câstigătoare utilizatorilor care au ajutat, în acest mod, comunitatea poate crește și aplicația se poate dezvolta.

În concluzie, există în momentul actual o mulțime de aplicații în aria de interes a turismului. Acestea se diferențiază în funcție de scopul principal pe care doresc să îl realizeze. Majoritatea incurajează comunicarea între persoane, ceea ce confirmă faptul că experiența unei alte persoane este o foarte puternică sursă de informație.

“Your Guide” se diferențiază de restul aplicațiilor prin faptul că are implementat un chatbot care face legatura între persoane. Logica din spatele acestui chabot se bazează pe un chestionar adresat persoanelor ce fac parte din diverse categorii de vârstă. Pe baza răspunsurilor primite în urma chestionarului au rezultat anumite procentaje asociate întrebărilor chatbotului.

3. Prin ce se diferențiază aplicația YourGuide

Luând în considerare scopul aplicației și aplicațiile similare, funcționalitatea principală a proiectului de față este chatbot-ul.

Chatbot-ul face legatura dintre utilizatori, între cel care dorește să călătorească și ghid. Acestă legatura nu este făcută întâmplător, ci pe baza unei logici bine pusă la punct.

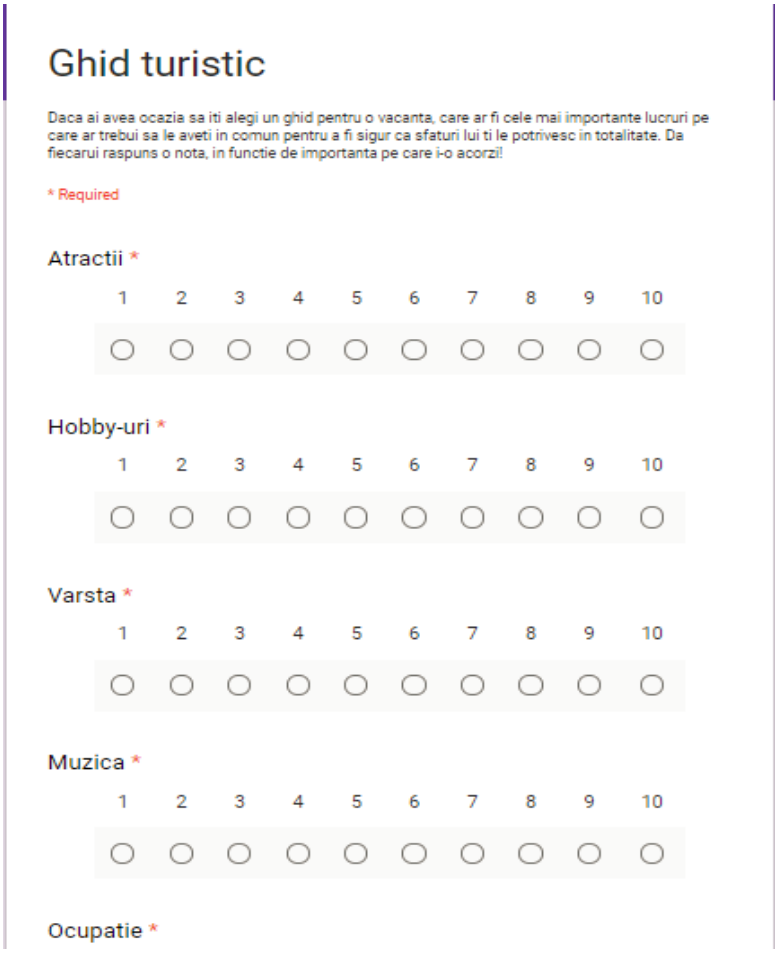
Într-o primă etapă, vizitatorul și ghidul vor trebui să aibă în comun lucruri esențiale:

- ✓ Destinația
- ✓ Limbile vorbite (cel puțin una să fie comună)
- ✓ Genul (în cazul în care călătorul specifică explicit faptul că dorește să aibă drept ghid o persoană de gen feminin/masculin)

Următoarele informații importante în alegerea ghidului potrivit sunt:

- o Atracțiile turistice
- o Hobby-urile
- o Vârsta
- o Genul muzical
- o Ocupația

Fiecare dintre informațiile de mai sus va avea anumite procentaje desemnate pe baza unui chestionar completat de 20 de persoane (12 persoane cu vârste cuprinse între 21-30 ani și 8 persoane cu vârste cuprinse între 41-50 ani).



Ghid turistic

Daca ai avea ocazia sa iti alegi un ghid pentru o vacanta, care ar fi cele mai importante lucruri pe care ar trebui sa le aveti in comun pentru a fi sigur ca sfaturi lui ti le potrivesc in totalitate. Da fiecarui raspuns o nota, in functie de importanta pe care i-o acorzi!

* Required

Atractii *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Hobby-uri *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Varsta *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Muzica *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Ocupatie *

Fig 3.1 Formular realizat pentru alocarea procentajelor

Întrebarea adresată în formular: “Dacă ai avea ocazia să îți alegi un ghid pentru o vacanță, care ar fi cele mai importante lucruri pe care ar trebui să le aveți în comun pentru a fi sigur că sfaturile lui ți se potrivesc în totalitate?”.

În urma notelor date fiecărui răspuns, au rezultat următoarele procentaje:

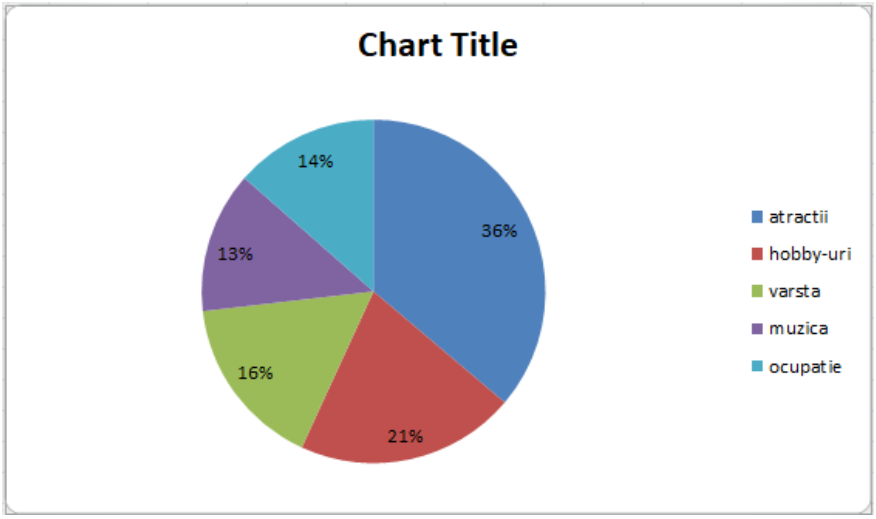


Fig 3.2 Procentaje rezultate în urma sondajului

4. Tehnologii web

Înainte de a alege tehnologiile web trebuie să ne punem întrebarea de ce fel de aplicație avem nevoie, Single Page Application (SPA) sau Multiple Page Application (MPA).

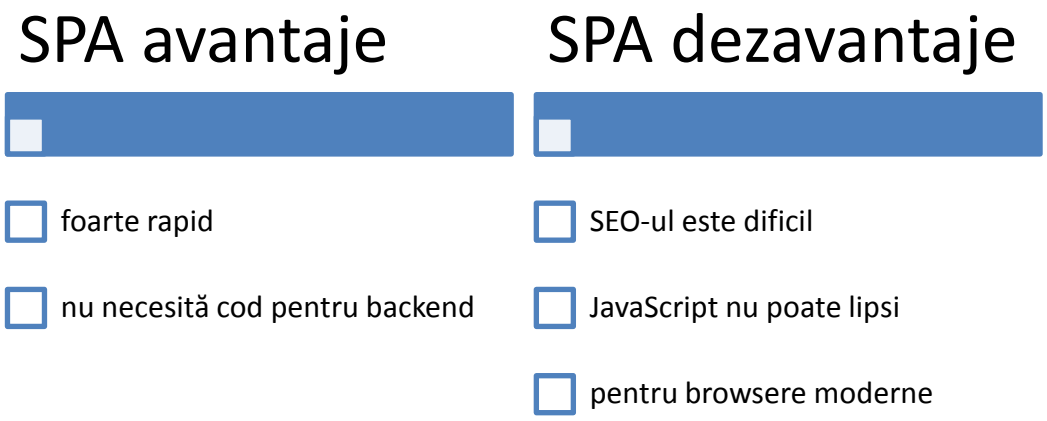


Fig 4.1 Avantaje și dezavantaje SPA

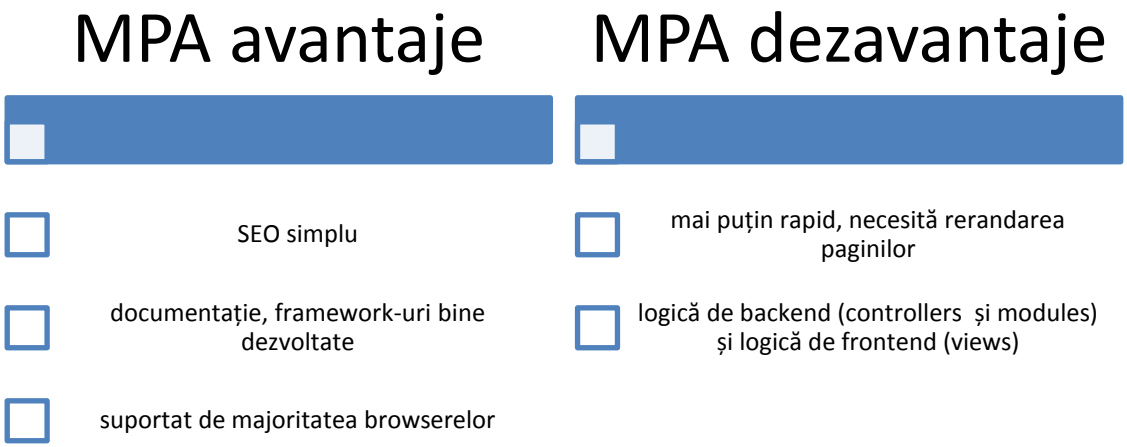


Fig 4.2 Avantaje și dezavantaje MPA

SEO - Server Engine Optimization

4.1 Instalare pachete

Pentru instalarea diferitelor pachete necesare proiectului am folosit NPM, un manager de pachete pentru modulele Node.js.

www.npmjs.com găzduiește mii de pachete gratuite de descărcat și de utilizat. Odată cu instalarea Node-ului, vom avea și managerul de pachete NPM instalat.
Un pachet din Node.js conține toate fișierele de care avem nevoie pentru un modul. Modulele sunt biblioteci JavaScript pe care le putem include în proiect.

Instalarea unui pachet se face foarte ușor, un exemplu este cel de mai jos:

Download "upper-case":

```
C:\Users\Your Name>npm install upper-case
```

Fig 4.3 Instalare pachet [13]

NPM creează un folder numit "node_modules", unde pachetul va fi plasat. Toate pachetele pe care le instalăm vor fi plasate în acest folder.

NPM ține evidența tuturor pachetelor și a versiunilor acestora și îi permite dezvoltatorului să actualizeze sau să elimine cu ușurință aceste dependențe. Toate aceste dependențe externe sunt stocate în interiorul unui fișier numit package.json.
Fișierul inițial poate fi creat cu ușurință utilizând CLI npm init (presupunând că NodeJS este instalat în sistem).

Pentru a folosi un pachet instalat trebuie să îl importăm în fișierul unde dorim să îl utilizăm.

```
import React from 'react';
import Select from 'react-select';
```

Fig 4.4 Importare pachete

Când suntem pregătiți să facem deploy pe producție, rularea npm run build va crea o variantă optimizată a aplicației în folderul build.

4.2 Interfața aplicației

Interfața este realizată doar pentru zona web, existând mai multe variante de framework-uri pentru implementare:

- React
- Angular
- Vue.JS

Dintre toate aceste tehnologii am ales să folosesc React, un framework pentru aplicațiile web, dezvoltat de inițiatorii Facebook. Acesta știe să randeze eficient doar componenta în care datele s-au schimbat, din acest motiv este foarte inteligent, spre deosebire de framework-ul Angular1, care rerandează întreaga componentă, în cazul în care ceva se modifică.

Abordarea React minimizează noile concepte care trebuie invățate, fiind mai apropiat de standardele JavaScript.

[1]: "Unul dintre lucrurile foarte bune la React este faptul că view templates (sabloanele de vederi) sunt create in JavaScript sau fișiere JSX. Avantajele rezultate:

- Nu trebuie să inventăm și să învățăm noi structuri de controale pentru a manipula HTML, de exemplu directive "for" din Angular. Putem folosi cod JavaScript pur, de exemplu "array.map" pentru a realiza iteratii intr-o colectie.
- Orice eroare de sintaxă va fi remarcată din timp la compilare

Managementul de stare este dificil și în cazul în care aplicația ajunge la o dimensiune mai mare, pot apărea uneori erori în aplicație. Pentru a ușura partea de management, se folosește foarte mult librăria Redux.

Redux se bazează pe trei principii:

- Întreaga stare a aplicației este stocată într-un obiect arbore, într-un singur loc de stocare;
- Singura modalitate prin care se poate schimba arborele de stare este emiterea unei acțiuni, un obiect ce descrie ceea ce s-a întâmplat;
- Pentru a specifica modul în care acțiunile transformă arborele de stare, se scriu reductori puri; "

Pentru aplicatia de față am ales să nu folosesc librăria Redux, deoarece dimensiunile aplicației nu necesitau acest lucru.

Angular reprezintă o alternativă pentru React. Este dezvoltat de Google și reprezintă, de asemenea, un framework foarte cunoscut și folosit.

Vue.JS reprezintă un framework de curând apărut. Este o combinație între React și Angular, prezentând avantajele amândurora. Drept rezultat, Vue.JS surprinde prin simplitatea pe care o oferă.

React folosește sintaxa JSX, în schimbul sintaxei de JavaScript obișnuit. Sintaxa JSX arată ca limbajul HTML.

```
import React from 'react';

class App extends React.Component {
   render() {
      return (
         <div>
            Hello World!!!
         </div>
      );
   }
}
export default App;
```

Fig 4.5 Exemplu sintaxă JSX

Chiar dacă este similar cu HTML, există câteva lucruri pe care trebuie să le avem în vedere atunci când lucrăm cu JSX:

- Dacă vrem să returnăm mai multe elemente, trebuie să le încapsulăm pe toate într-un element de container.

- Putem folosi propriile atribute personalizate pe lângă proprietățile și atributele obișnuite ale limbajului HTML. Când vrem să adăugăm un atribut personalizat, trebuie să folosim prefixul de date. Exemplu: data-myattribute = "somevalue".

- Expresiile JavaScript pot fi folosite în interiorul JSX. Trebuie doar să le folosim în interiorul acoladelor {}.

- Nu putem folosi declarații în interiorul JSX, dar putem folosi expresii condiționale.

- React recomandă utilizarea stilurilor inline. Când vrem să setăm stiluri inline, trebuie să folosim sintaxa camelCase.

```
import React from 'react';

class App extends React.Component {
   render() {
      var myStyle = {
         fontSize: 100,
         color: '#FF0000'
      }
      return (
         <div>
            <h1 style = {myStyle}>Header</h1>
         </div>
      );
   }
}
export default App;
```

Fig 4.6 Exemplu utilizare inline style

- Etichetele HTML utilizează întotdeauna nume de etichete cu litere mici, în timp ce componentele React încep cu literă mare.

Pentru crearea proiectului utilizând React am ales să folosesc pachetul create-react-app. Acesta configurează mediul de dezvoltare pentru a putea utiliza cele mai recente funcții JavaScript și optimizează aplicația pentru producție. Pentru a putea utiliza acest pachet va trebui sa avem instalată o versiune de Node.js nu mai veche de v6.
Create React App nu gestionează logica de backend sau baze de date, ci doar creează o legătură cu frontend-ul, astfel încât să îl putem folosi cu orice backend. Utilizează instrumente precum Babel și Webpack, dar funcționează fără vreo configurație.

Ca orice limbaj de programare, Javascript are, de asemenea, versiuni numite ECMAScript (scurt ES). În prezent, majoritatea browserelor suportă ES5.
Din păcate nu toate browserele suportă caracteristicile lui ES6 (specificațiile limbajului au fost finalizate în 2015). Babel este un transformator JS care convertește noul cod JS în cele vechi. Este un instrument foarte flexibil. Se pot adăuga cu ușurință presetări, cum ar fi es2015, es2016, es2017, astfel încât Babel le compilează la ES5.

Webpack este un instrument modular care are două seturi de funcționalități - Loaders și Plugins. Loaders transformă codul sursă al unui modul. De exemplu, sass-loader compilează fișiere SASS în CSS.

Pentru stilizarea aplicației am ales să folosesc clase din Bootstrap [9], Font Awesome [10] , Google Icons [11] și Flexbox [16]. Flexbox-ul ne oferă un control complet asupra alinierii, direcției, ordinii și mărimii elementelor noastre, și este foarte ușor de folosit.

4.3 Backend-ul aplicației

Pentru partea de back-end a aplicației am ales să folosesc o platformă care îmi asigură accesul din client la baza de date – Firebase, dezvoltat de Google.

Acest serviciu este usor de folosit și prezintă o mulțime de tool-uri foarte utile în dezvoltarea aplicațiilor, printre care și cele folosite în realizarea aplicației de față – serviciul de autentificare, de stocare și baza de date în timp real. Pentru toate aceste servicii ne sunt puse la dispoziție serverele Google, dezvoltatorii neavând nevoie sa își creeze un server propriu.

Baza de date este de tip NoSQL și are implementat protocolul de comunicație WebSockets.

WebSockets este principiul care stă la baza ideii de „bază de date în timp real”. Protocolul creează o sesiune activă atunci când se realizează o conexiune între client și server. Această conexiune este deschisă cât timp ambii (clientul si serverul) sunt activi. Protocolul HTTP nu realizează acest lucru, acesta nu memorează conexiunile, se bazează doar pe primirea unui request, căruia îi răspunde, după care conexiunea este închisă.

[3]: “Avantaje WebSockets:

- Protocol bidirecțional - fie clientul / server-ul poate trimite un mesaj celeilalte părți (în HTTP, cererea este inițiată întotdeauna de client și răspunsul este procesat de server - făcând HTTP un protocol unidirecțional)
- Comunicare full-duplex - clientul și serverul pot vorbi unii cu alții independent în același timp
- Conexiune unică TCP - După actualizarea conexiunii HTTP la început, clientul și serverul comunică prin aceeași conexiune TCP de-a lungul ciclului de viață al conexiunii Websocket

În concluzie, un canal bidirecțional, este mai atractiv pentru lucruri precum jocuri, aplicații de mesagerie, instrumente de colaborare, experiențe interactive și pentru cazuri în care avem nevoie de actualizări în timp real în ambele direcții.”

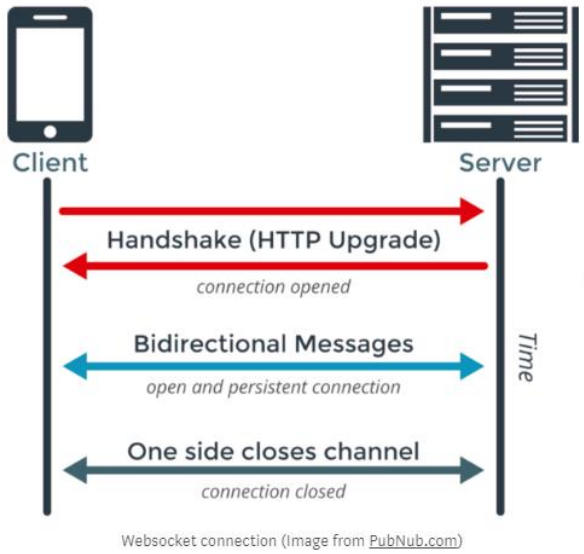


Fig 4.7 Principiu de funcționare WebSockets [3]

Limitări protocol HTTP:

- serverul nu poate deschide o conexiune cu clientul
- retrimite header-urile de fiecare dată - crește costul conexiunii
- protocolul TCP, care mai întâi face un handshake, apoi trimite datele

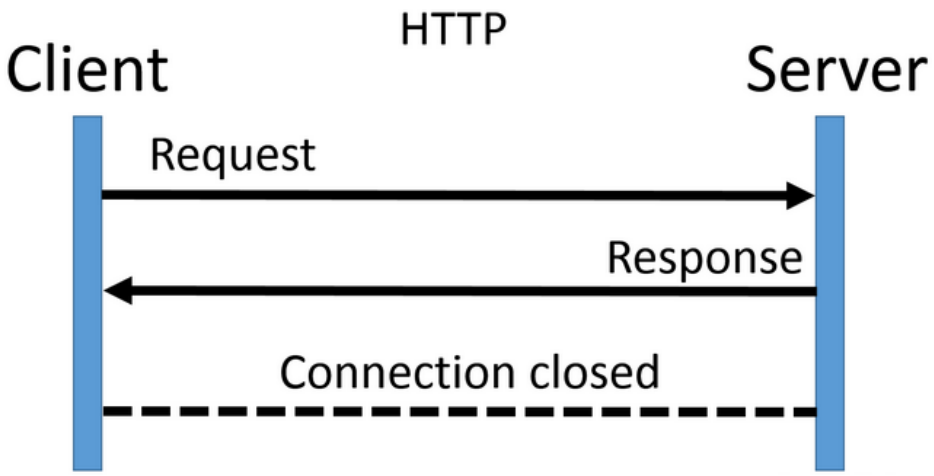


Fig 4.8 Principiu de funcționare HTTP [4]

WebSockets prezintă, de asemenea, ultimele doua limitări. Însă, atât header-urile costisitoare, cât și handshake-ul, nu mai sunt o limitare, ele fiind trimise / făcute o singură dată, la stabilirea conexiunii.

NoSQL, după cum îi spune și numele, în comparație cu bazele de date relaționale, nu se bazează pe conceptul de legături între tabele. O astfel de bază de date stochează datele sub forma unui arbore.

Datele pot fi accesate prin noduri, de exemplu, în cazul în care avem calea "/users/messages/text":

- "text" - nodul curent
- "messages" – nodul părinte
- "users" – nodul părinte al precedentului
- "/" – nodul rădăcină

Firebase oferă o interfață foarte intuitivă utilizatorilor:

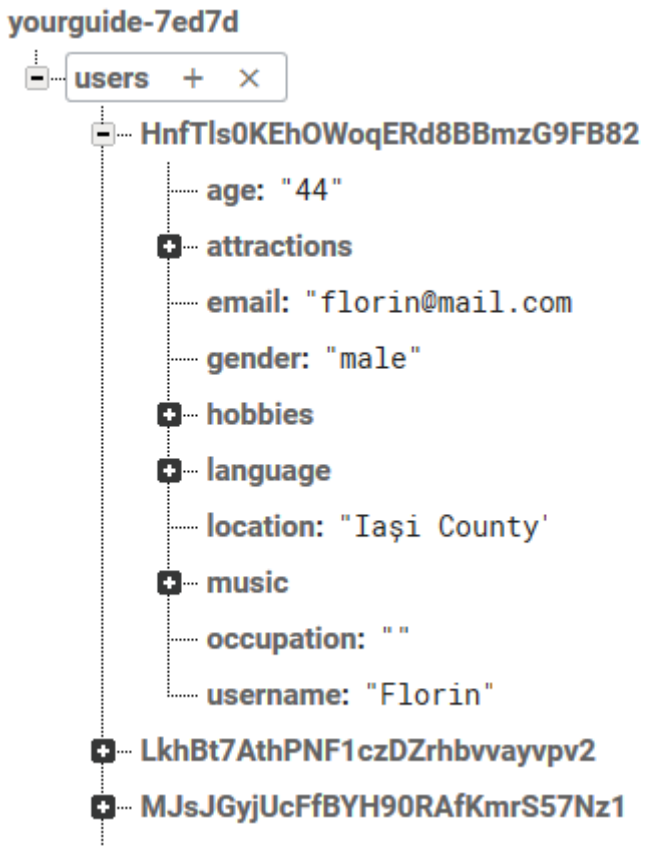


Fig 4.9 Structură bază de date în Firebase

Utilizare Firebase:

Inițializarea bazei de date se face adaugând informațiile de configurare generate de Firebase:

```
var config = {
    apiKey: "AIzaSyDku16OmHIX7yLJgrQB81XjjW-mgdN3Rac",
    authDomain: "yourguide-7ed7d.firebaseapp.com",
    databaseURL: "https://yourguide-7ed7d.firebaseio.com",
    projectId: "yourguide-7ed7d",
    storageBucket: "yourguide-7ed7d.appspot.com",
    messagingSenderId: "889317621423"
};
```

Fig 4.10 Informații configurare Firebase

Pentru a scrie sau a citi din baza de date avem nevoie de o referință către aceasta:

```
const db = firebase.database();
```

Fig 4.11 Referință către baza de date

Pentru operațiile de scriere în baza de date, putem utiliza metoda set() pentru a salva datele într-o referință specificată, înlocuind orice date existente pe acea cale.

De exemplu:

```
function writeUserData(userId, name, email, imageUrl) {
  firebase.database().ref('users/' + userId).set({
    username: name,
    email: email,
    profile_picture : imageUrl
  });
}
```

Fig 4.12 Exemplu Firebase folosire metodă set()

Pentru a citi datele și pentru a asculta modificările, utilizăm metodele on() sau once() din firebase.database.Reference.

De exemplu:

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});
```

Fig 4.13 Exemplu Firebase folosire metodă on()

Snapshot conține datele de la locația specificată din baza de date în momentul evenimentului. Datele din snapshot sunt preluate cu ajutorul metodei val().

În unele cazuri, este posibil să ne dorim un snapshot al datelor fără a asculta schimbări, cum ar fi atunci când se inițializează un element UI care nu se schimbă. Putem utiliza metoda once() pentru a simplifica acest scenariu: se declanșează o singură dată.

Acest lucru este util pentru datele care trebuie să fie încărcate o singură dată și nu se așteaptă să se schimbe frecvent sau să necesite ascultarea activă.

De exemplu:

```
var userId = firebase.auth().currentUser.uid;
return firebase.database().ref('/users/' + userId).once('value').then(function(snapshot) {
  var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
  // ...
});
```

Fig 4.14 Exemplu Firebase folosire metodă once()

Există, de asemenea, operația de update(). Aceasta ne permite să modificăm câmpurile deja existente. Tot cu această metodă putem sa creăm câmpuri care nu existau inițial.

Exemplu folosirea metodei update() pentru modificarea câmpurilor în aplicație:

```
let user = firebase.auth().currentUser
    if(user) {

        let rootRef = firebase.database().ref('users/' + user.uid);
        console.log(self);
        let newrootRef = rootRef.update({
            "age": self.state.age,
            "music": self.state.music,
            "language":self.state.language,
            "gender": self.state.selectedOption,
            "occupation": self.state.occupation,
            "hobbies": self.state.hobby,
            "attractions": self.state.attraction,
            "location": self.state.location
        });
    }
```

Fig 4.15 Utilizarea metodei update() în aplicație

Liste de date:

Utilizăm metoda push() pentru a adăuga date într-o listă în aplicații multiuser. Metoda push() generează o cheie unică de fiecare dată când un nou copil este adăugat la referința Firebase specificată. Prin utilizarea acestor chei generată automat pentru fiecare element nou din listă, mai mulți clienți pot adăuga copii în aceeași locație, în același timp, fără a crea conflicte de scriere. Cheia unică generată de push() se bazează pe o marcă de timp, astfel încât elementele din listă sunt ordonate automat în ordine cronologică.

Proprietatea .key a unei referințe push() conține cheia generată automat.

Exemplu utilizare:

```
// Create a new post reference with an auto-generated id
var newPostRef = postListRef.push();
newPostRef.set({
    // ...
});
```

Fig 4.16 Exemplu Firebase folosire metodă push()

În aplicație am utilizat metoda push() pentru crearea mesajelor la o anumită referință:

```
let guideRef = receiverRef.child(this.state.destination); //pe contul Elenei la receiver = Maria
guideRef.push({
  text: this.state.message,
  time: firebase.database.ServerValue.TIMESTAMP
})
```

Fig 4.17 Utilizarea metodei push() în aplicație

Fiecare mesaj din baza de date are o cheie unică si conține textul propriu-zis, sursa, destinația, dar si momentul când a fost creat:

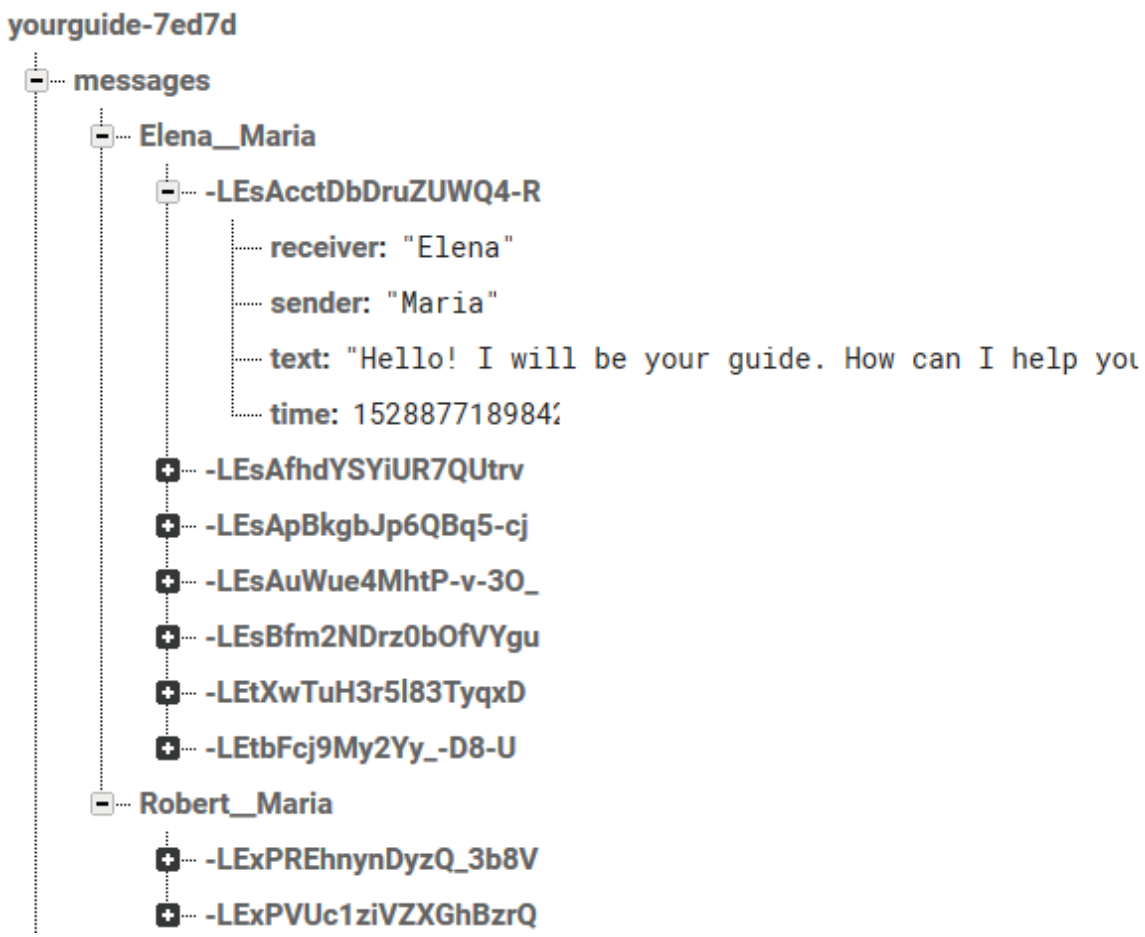


Fig 4.18 Firebase - Baza de date

Evenimente pentru copiii unui nod:

Evenimentele pentru copii sunt declanșate ca răspuns la operațiile specifice care se întâmplă copiilor unui nod dintr-o operație, de exemplu un nou copil adăugat prin metoda push() sau un copil care este actualizat prin metoda update().

- Child_added:

  Acest eveniment este declanșat o dată pentru fiecare copil existent și apoi din nou de fiecare dată când un copil nou este adăugat pe calea specificată. Ascultătorului i se transmite un snapshot care conține datele noului copil.

- Child_changed:

  Acest eveniment este declanșat de fiecare dată când un nod copil este modificat. Aceasta include orice modificare a descendenților nodului copil. Snapshot-ul transmis către ascultătorul evenimentului conține datele actualizate pentru copil.

- Child_removed:

  Acest eveniment este declanșat când un copil imediat este eliminat. Snapshot-ul trimis către blocul callback conține datele copilului eliminat.

- Child_moved:

  Evenimentele de tipul child_moved urmează întotdeauna evenimentul child_changed care a provocat modificarea ordinii elementului (pe baza metodei curente de ordonare).

Exemple de utilizare:

```
var commentsRef = firebase.database().ref('post-comments/' + postId);
commentsRef.on('child_added', function(data) {
  addCommentElement(postElement, data.key, data.val().text, data.val().author);
});

commentsRef.on('child_changed', function(data) {
  setCommentValues(postElement, data.key, data.val().text, data.val().author);
});

commentsRef.on('child_removed', function(data) {
  deleteComment(postElement, data.key);
});
```

Fig 4.19 Exemple Firebase utilizare evenimente child_added, child_changed și child_removed

Atașarea unui observator de valori la o listă de date va returna întreaga listă de date ca un singur snapshot pe care îl putem parcurge pentru a accesa copiii individual. Acest model poate fi util atunci când dorim să preluam toți copiii dintr-o listă într-o singură operațiune, în loc să ascultăm evenimente suplimentare adăugate de copil.

```
ref.once('value', function(snapshot) {
  snapshot.forEach(function(childSnapshot) {
    var childKey = childSnapshot.key;
    var childData = childSnapshot.val();
    // ...
  });
});
```

Fig 4.20 Exemplu Firebase utilizare motedă once() și parcurgere a valorilor retunate

Utilizare în cod pentru găsirea tuturor utilizatorilor și cheilor acestora unice:

```
let refDB = firebase.database().ref("users");
refDB.once("value").then(function(snapshot) {
  snapshot.forEach(function(childSnapshot) {
    let childKey = childSnapshot.key;
    let childData = childSnapshot.val().username; //Maria
```

Fig 4.21 Exemplu parcurgere valori în aplicație

Sortarea și filtrarea datelor în Firebase:

1. Metode utilizate pentru sortarea datelor:
   - OrderByChild:

     Ordonează rezultatele după valoarea unei chei copil specificate.
   - OrderByKey:

     Pune în ordine rezultatele folosind cheile copilului.
   - OrderByValue:

     Pune în ordine rezultatele folosind valorile copilului.

Exemplu de utilizare:

```
var myUserId = firebase.auth().currentUser.uid;
var topUserPostsRef = firebase.database().ref('user-posts/' + myUserId).orderByChild('starCount
```

Fig 4.22 Exemplu Firebase folosire metodă orderByChild()

```
var mostViewedPosts = firebase.database().ref('posts').orderByChild('metrics/views');
```

Fig 4.23 Exemplu Firebase folosire metodă orderByChild()

2. Metode utilizate pentru filtrarea datelor:
    - LimitToFirst():

      Setează numărul maxim de elemente pentru a fi returnate de la începutul listei de rezultate ordonate.
    - LimitToLast():

      Setează numărul maxim de elemente pentru a fi returnate de la sfârșitul listei de rezultate ordonate.
    - StartAt():

      Returnează elemente mai mari sau egale cu cheia sau valoarea specificată, în funcție de metoda de ordonare aleasă.
    - EndAt():

      Returnează elemente mai mici sau egale cu cheia sau valoarea specificată, în funcție de metoda de ordonare aleasă.
    - EqualTo():

      Returnează elemente egale cu cheia sau valoarea specificată, în funcție de metoda de ordonare aleasă.

Autentificare folosind Firebase:

Pentru a putea folosi serviciul de autentificare trebuie să creăm o referință către acesta:

```
const auth = firebase.auth();
```

Fig 4.24 Exemplu creare referință către serviciul de autentificare în aplicație

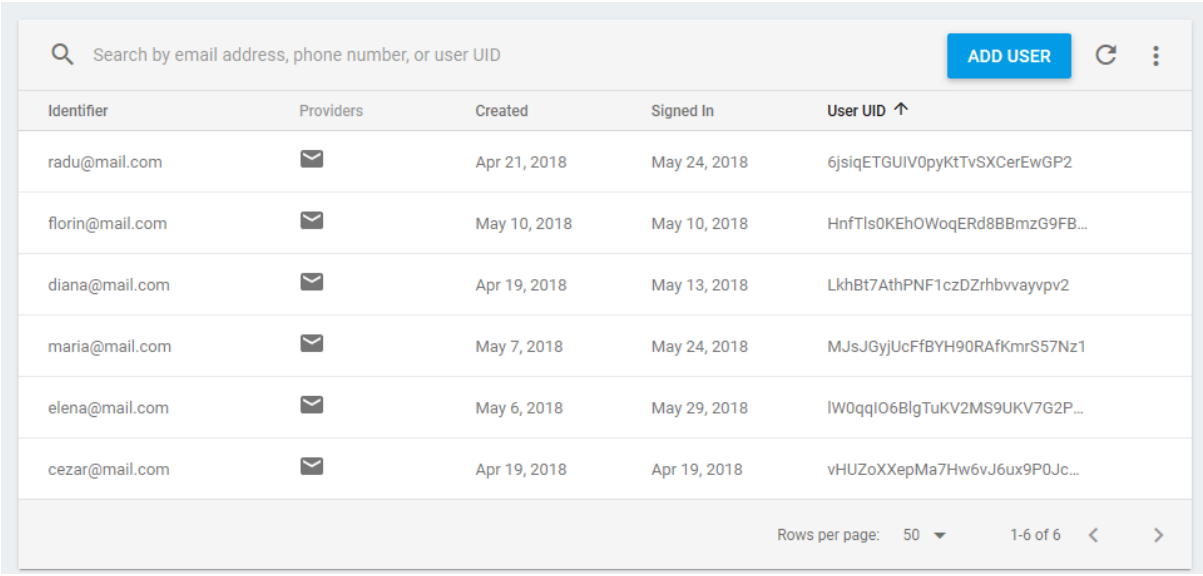


Fig 4.25 Interfață autentificare în Firebase

Firebase, pentru partea de autentificare, oferă utilizatorilor diverse posibilități:

- Autenficare normală (nume user + parola). În acest caz se poate opta pentru recuperarea parolei prin serviciul de mail
- Autentificare folosind contul de Facebook
- Autentificare folosind contul de Google
- Autentificare folosind contul de Twitter
- Autentificare folosind contul de GitHub
- Autentificare folosind numărul de telefon

Pentru acest proiect am ales să folosesc autentificarea obișnuită, utilizatorul se loghează cu adresa de mail și parola, iar în cazul în care acesta nu își mai amintește parola, și-o poate schimba folosind serviciul de mail.

5. Implementarea și funcționalitatea aplicației YourGuide

Vor fi prezentate următoarele ecrane:

- Sign Up
- Sign In
- Recuperare parolă
- Schimbare parolă
- Alegerea destinației de călătorie
- Completarea informațiilor de către utilizatorii care vor sa devină ghizi
- Chatbot
- Chat
- Căutare utilizatori
- Profil utilizator

5.1 Logare

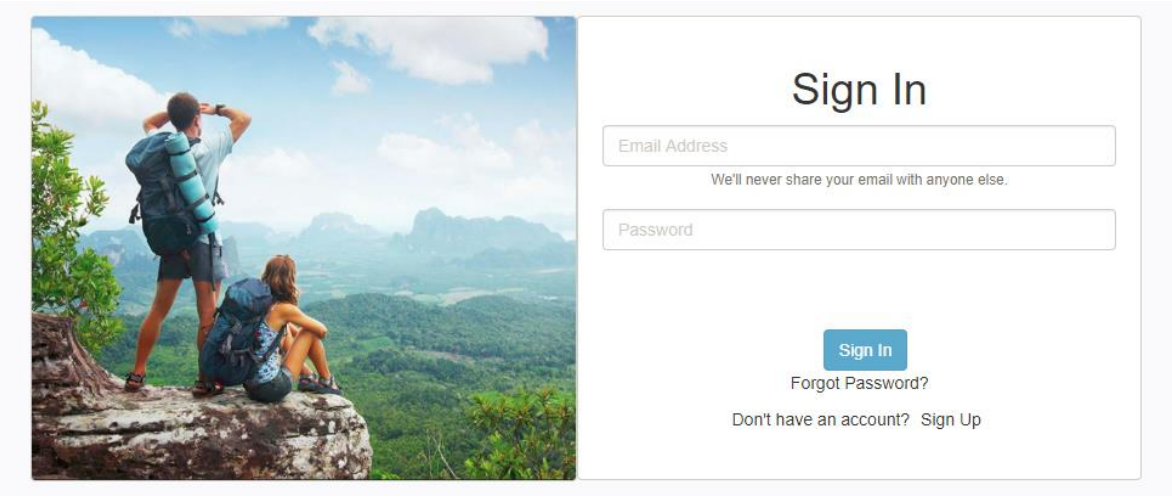


Fig 5.1 Ecran Logare

Ecranul de înregistrare conține două câmpuri:

- Email
- Password

Butonul “Sign In” rămâne inactiv până când utilizatorul completează toate câmpurile. Câmpul de email, pentru a fi valid, trebuie să aibă “@” în conținut. În momentul în care un utilizator acționează butonul de logare, acesta va fi redirecționat pe pagina de Home.

Pentru redirectări se folosește pachetul react-router-dom. Router-ul include mai multe componente, dar cele trei pe care le-am utilizat cel mai des sunt <BrowserRouter>, <Route> și <Link>. Primul, <BrowserRouter>, este de obicei un alias de "Router" și este componenta părinte care este utilizată pentru a stoca toate componentele <Route>. Componentele <Route> sunt cele care spun aplicației ce alte componente se afișează pe baza traseului. Componentele <Link> reprezintă modul în care creăm link-uri către aceste rute.

În exemplu de mai sus, avem rutele pentru recuperarea parolei și pentru înregistrare.

```
const PasswordForgetLink = () =>
  <p className="link">
    <Link to="/pw-forget">Forgot Password?</Link>
  </p>
```

Fig 5.2 Exemplu folosire Link

Pentru o mai bună gestiune am creat un fișier separat routes.js, unde definesc toate rutele utilizate în aplicație.

```
export const SIGN_UP = '/signup';
export const SIGN_IN = '/signin';
export const LANDING = '/';
export const HOME = '/home';
export const ACCOUNT = '/account';
export const PASSWORD_FORGET = '/pw-forget';
export const CHATBOT = '/chatbot';
export const GUIDE = '/guide';
export const CHAT = '/chat';
export const CHATONE = '/chat-chatbot';
```

Fig 5.3 Definire rute

Rutele și componentele pe care le accesează sunt definite in fișierul App.js.

```
class App extends React.Component {
  render() {
    return (
      <div className="App">
        <Router>
          <div>
            <Navigation/>

            <Route
              exact path={routes.LANDING}
              component={LandingPage}
            />
            <Route
              exact path={routes.GUIDE}
              component={GuidePage}
            />
            <Route
              exact path={routes.HOME}
              component={HomePage}
            />
            <Route
              exact path={routes.CHATBOT}
              component={ChatbotPage}
            />
```

Fig 5.4 Exemplu utilizare <Route>

5.2 Înregistrare

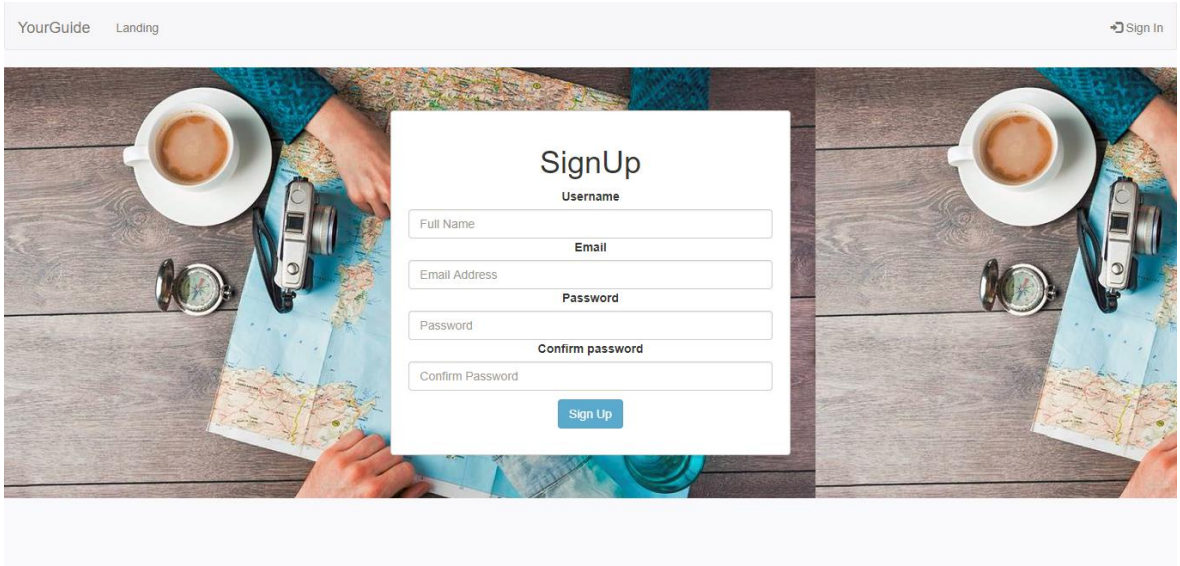


Fig 5.5 Ecran înregistrare

Ecranul de înregistrare conține patru câmpuri:

- Username
- Email

- Password
- Confirm Password

Butonul “Sign Up” rămâne inactiv până când utilizatorul completează toate câmpurile. Fiecare input are tipul specificat, de exemplu, nu se poate introduce un string pentru câmpul de email care să nu aibă “@” în conținut.

Înregistrarea în baza de date se face folosind serviciul de autentificare din Firebase. Metoda “doCreateUserWithEmailAndPassword” creează înregistrările în partea de autentificare.

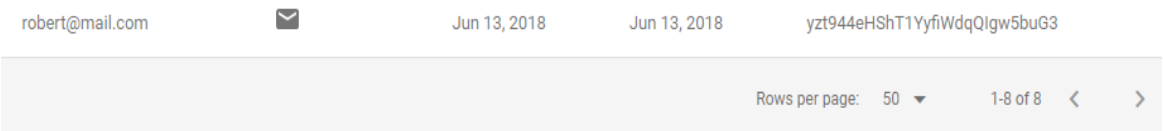


Fig 5.6 Firebase – autentificare

Metoda “doCreateUser” scrie înregistarea în baza de date creată de mine.

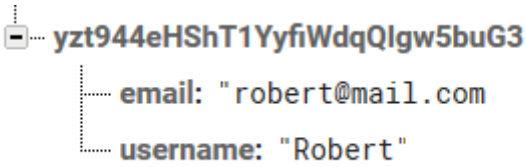


Fig 5.7 Firebase - Baza de date

```
auth.doCreateUserWithEmailAndPassword(email, passwordOne)
  .then(authUser => {

    // Create a user in your own accessible Firebase Database too
    db.doCreateUser(authUser.uid, username, email)
      .then(() => {
        this.setState(() => ({ ...INITIAL_STATE }));
        history.push(routes.HOME);
      })
      .catch(error => {
        this.setState(byPropKey('error', error));
      });
```

Fig 5.8 Utilizare metode “doCreateUserWithEmailAndPassword” și “doCreateUser”

În momentul în care un utilizator acționează butonul de înregistrare, acesta va fi redirecționat pe pagina de Home.

5.3 Recuperare parolă

În cazul în care introducem un utilizator care nu se află in baza de date, aplicația mă va atenționa.

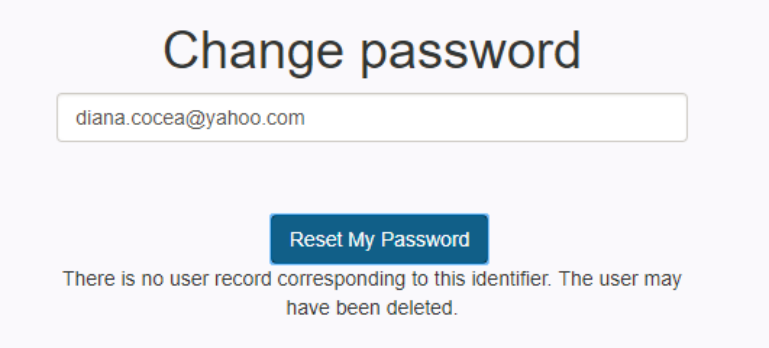


Fig 5.9 Ecran schimbare parolă

Recuperarea parolei se face tot cu ajutorul serviciului de autentificare, mai exact prin metoda “doPasswordReset”.

```
onSubmit = (event) => {
  const { email } = this.state;

  auth.doPasswordReset(email)
    .then(() => {
      this.setState(() => ({ ...INITIAL_STATE }));
    })
    .catch(error => {
      this.setState(byPropKey('error', error));
    });

  event.preventDefault();
}
```

Fig 5.10 Utilizare metodă “doPasswordReset”

Prin acest serviciu vom primi un mail pe adresa specificată. Email-ul respectiv conține un link care ne permite schimbarea parolei.

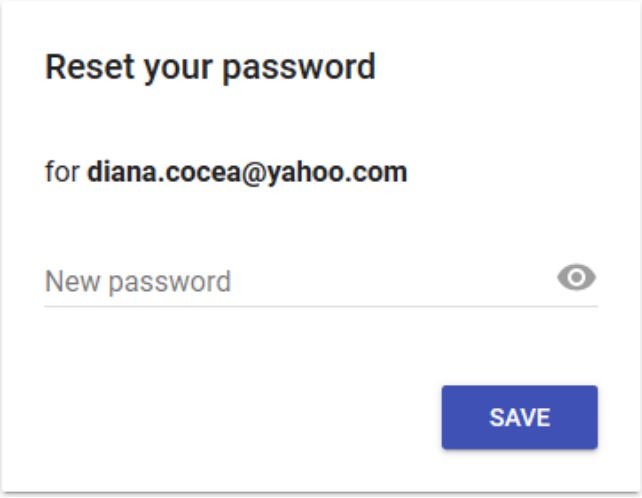


Fig 5.11 Ecran schimbare parolă prin email

5.4 Schimbare parolă

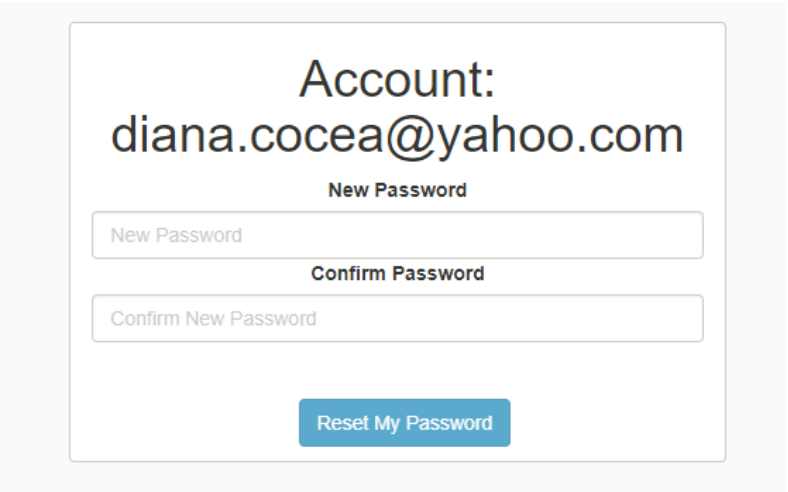


Fig 5.12 Ecran schimbare parolă

Recuperarea parolei se face tot cu ajutorul serviciului de autentificare, prin metoda "doPasswordUpdate".

```
onSubmit = (event) => {
  const { passwordOne } = this.state;

  auth.doPasswordUpdate(passwordOne)
    .then(() => {
      this.setState(() => ({ ...INITIAL_STATE }));
    })
    .catch(error => {
      this.setState(byPropKey('error', error));
    });
```

Fig 5.13 Utilizare metodă "doPasswordUpdate"

5.5 Alegerea destinației

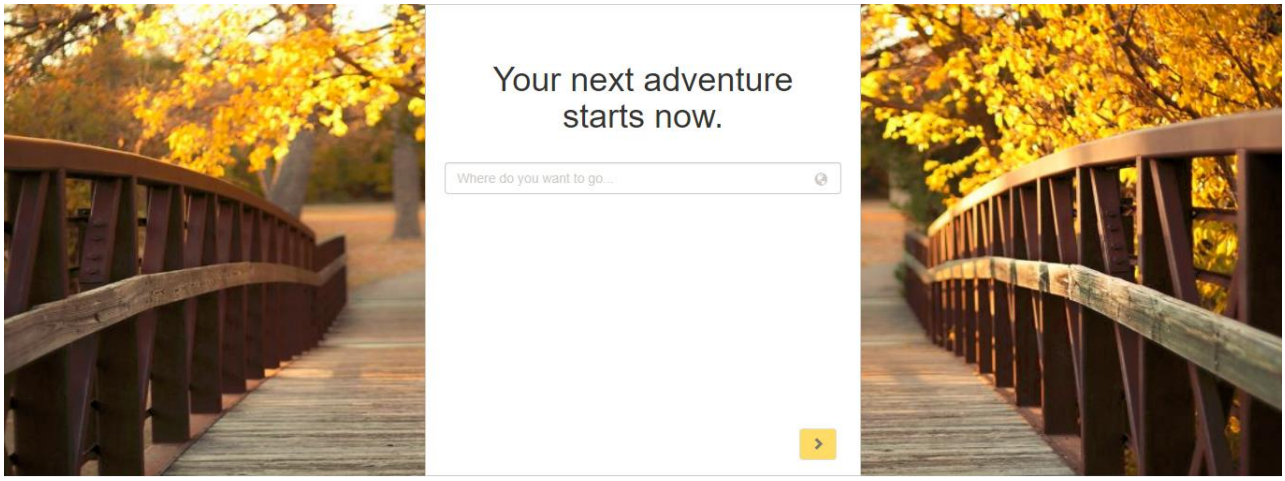


Fig 5.14 Ecran alegere destinație

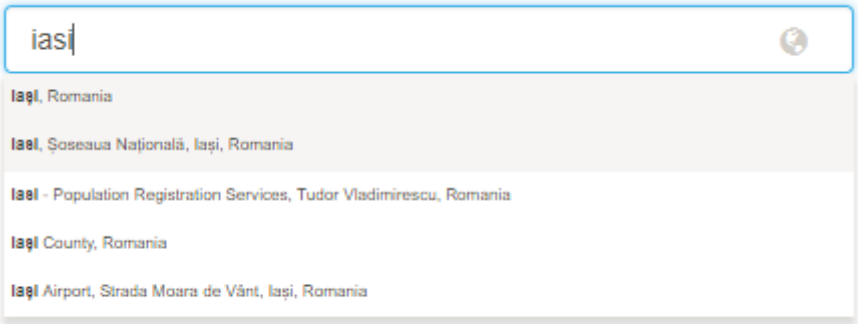


Fig 5.15 Sugestii locație

În fișierul index.html am inclus următoarea cheie:

```
<script src='http://maps.googleapis.com/maps/api/js?v=3&amp;libraries=places&key=AIzaSyB7Pdtjnuh18YQwwxg_f5o6_vc-yNWyNv4'></script>
```

Fig 5.16 Adăugare cheie API Google Maps

Pentru căutarea destinației am folosit pachetele react-google-maps-loader și react-google-places-suggest.

Rezultatele căutarii sunt filtrare, astfel încât ne sunt afișate subdiviziuni, de exemplu județe sau în Statele Unite, aceste niveluri administrative sunt state.

După ce se selectează destinația, butonul devine accesibil și ne trimite pe pagina Chatbot-ului. Pentru ca destinația aleasă să fie reținută, ne vom folosi de proprietățile din React.

Prin props putem trimite valori de la părinte la copil. În acest fel, valoarea destinației este trimisă către Chatbot.

```
const WrappedLink = (props) => {
  return (
    <Link to={{
      pathname: routes.CHATBOT,
      valueBack: props.valueBack
    }}>
      <button disabled={props.isValid} className="btn btn-default btn-color">
        <i className="glyphicon glyphicon-chevron-right"></i>
      </button>
    </Link>
  )
}
```

Fig 5.17 Valoare trimisă prin props - copil

```
<div className="btn-position">
  <WrappedLink valueBack={this.state.valueBack} isValid={isValid}/>
</div>
```

Fig 5.18 Valoare trimisă prin props – părinte

5.6 Completarea informațiilor de către ghizi

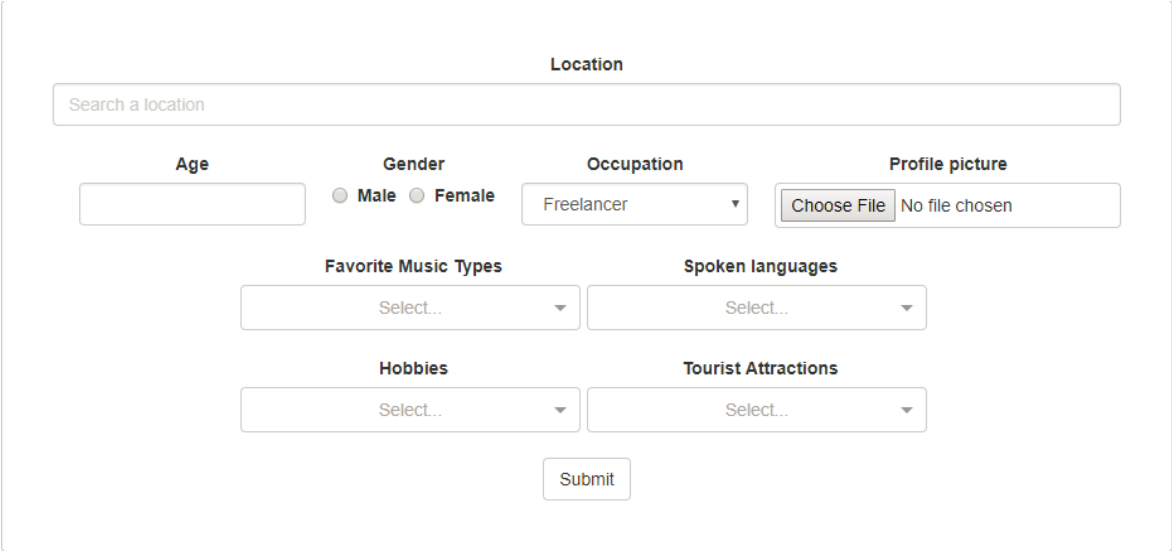


Fig 5.19 Completare informații ghid

Pentru ca un utilizator să devină ghid, acesta trebuie sa completeze informații personale esențiale.

Câmpurile de completat sunt:

- Locația (unde locuiește)
- Vârsta
- Genul
- Ocupația
- Imagine cu profilul real
- Genuri de muzică preferate
- Limbi vorbite
- Hobby-uri
- Atracții turistice de care este interesat în general

În momentul în care toate datele sunt completate și butonul submit este acționat, informațiile completate de fiecare ghid sunt înregistrare in baza de date.

Fig 5.20 Firebase – Baza de date

Pentru acest lucru, am folosit funcția update():

```
let rootRef = firebase.database().ref('users/' + user.uid);
console.log(self);
let newrootRef = rootRef.update({
    "age": self.state.age,
    "music": self.state.music,
    "language":self.state.language,
    "gender": self.state.selectedOption,
    "occupation": self.state.occupation,
    "hobbies": self.state.hobby,
    "attractions": self.state.attraction,
    "location": self.state.location,
 });
```

Fig 5.21 Utilizare funcție update() în aplicație

Aceste date pot fi în orice moment modificate de către ghid.

La accesarea paginii, informațiile completate vin din baza de date. Acest lucru este posibil prin utilizarea uneia dintre funcțiile specifice React-ului “componentWillMount()”. Fiecare componentă are mai multe "metode de ciclu de viață" pe care le putem suprascrie pentru a rula codul la anumite momente ale procesului. În cazul de față, funcția va fi apelată atunci când componenta este încărcată și datele sunt aduse din baza de date.

```
this.setState ({
    age: snapshot.val().age,
    music: snapshot.val().music.map(element => element.label),
    language: snapshot.val().language.map(element => element.label),
    gender: snapshot.val().gender,
    occupation: snapshot.val().occupation,
    attraction: snapshot.val().attractions.map(element => element.value),
    hobby: snapshot.val().hobbies.map(element => element.label),
    location: snapshot.val().location,
    loading: false
});
```

Fig 5.22 Aducerea datelor din baza de date

Dintre toate informațiile pe care ghid-ul trebuie să le completeze, încărcarea unei imagini de profil se face diferit.

Firebase Storage este o soluție autonomă pentru încărcarea conținutului generat de utilizatori, precum imaginile și videoclipurile de pe un dispozitiv iOS și Android, precum și pe Web. Firebase Storage este proiectat special pentru scalabilitate, securitate și reziliența rețelei.

- Scalabilitate: Fiecare fișier încărcat este susținut de Google Cloud Storage
- Securitate: Fișierele pot fi securizate pentru anumiți utilizatori sau seturi de utilizatori, utilizând regulile de securitate a stocării.
- Reziliența rețelei: Încărcările și descărcările sunt retrimise automat în cazul conexiunilor de rețea necorespunzătoare, astfel încât să nu trebuiască să le urmărim singuri.

Firebase Storage utilizează un sistem de fișiere pentru a structura datele. Fiecare fișier este accesat printr-o referință:

```
let storageRef = firebase.storage().ref('images/' + filename);
```

Fig 5.23 Referiță serviciul Storage – Firebase

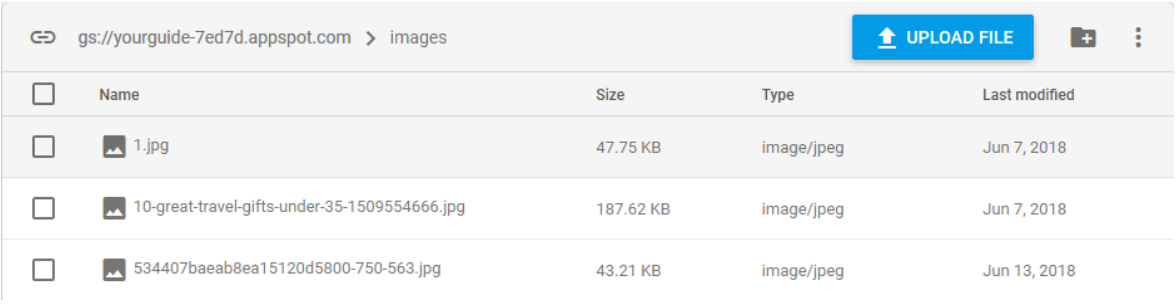


Fig 5.24 Firebase – Storage

Odată ce avem o referință, putem încărca un fișier cu o singură metodă:

```
let file = e.target.files[0];
```

```
let uploadTask = storageRef.put(file);
```

Fig 5.25 Încărcare fișier în Firebase - Storage

Observăm schimbările care se fac asupra state-ului cu ajutorul observatorului “state_changed” si metodei “on”:

```
uploadTask.on('state_changed', function(snapshot) {

}, function(error) {

}, function() {

    let downloadURL = uploadTask.snapshot.downloadURL;
    let user = firebase.auth().currentUser
    if(user) {

        let rootRef = firebase.database().ref('users/' + user.uid);

        let newrootRef = rootRef.update({
            "picture": downloadURL
        });
    }
});
```

Fig 5.26 Folosirea metodei on() și a observatorului state_changed

În cazul în care se observă o schimbare, se modifică și câmpul “picture” din baza noastră de date.

5.7 Chatbot

Chatbot-ul reprezintă următoarea etapă după alegerea destinației. Destinația pe care utilizatorul o alege reprezintă, de fapt, primul criteriu de selecție. Ghidul care urmează să fie ales pentru utilizator trebuie să provină din regiunea destinație.

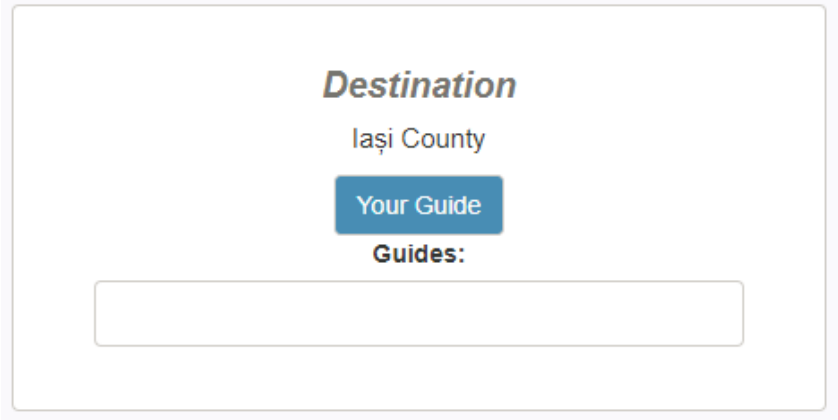


Fig 5.27 Ecran apariție ghid după parcugerea întrebărilor de către utilizator

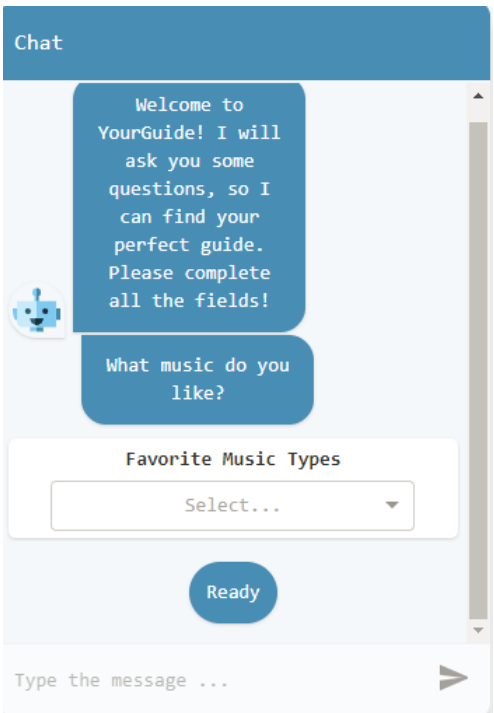


Fig 5.28 Ecran Chatbot – completare întrebări

Pentru realizarea chatbot-ului am folosit librăria din React – React Simple Chatbot [8]. Librăria m-a ajutat în construirea chatbot-ului prin logica trimiterii mesajelor, dar și vizual. Logica trimiterii mesajelor este următoarea:

```
<ChatBot className="center"
    steps={[
        {
            id: '1',
            message: 'Welcome to YourGuide! I will ask you some questions, so I can find your perfect guide.
            trigger: '3',
        },

        {
            id: '3',
            message: 'What music do you like?',
            trigger: 'music'
        },
        {
            id: 'music',
            component: (

                <SelectMusic onUpdate={this.onUpdateMusic}/>

            ),
            trigger: 'okMusic'
        },
```

Fig 5.29 Logica trimiterii mesajelor Chatbot-ului

Toate mesajele sunt puse într-un obiect steps. Trecerea de la un mesaj la altul se face printr-un trigger. Acest trigger trebuie să fie, de fapt, id-ul următorului mesaj.
Librăria are diverse funcționalități, cum ar fi:

- Previous Value (recunoaște mesajul transmis anterior)
- Speech Recognition
- Options(utilizatorul poate să aleagă opțiuni propuse de chatbot)
- Validator(pentru numere, de exemplu)
- Simple Form
- Custom Component

Dintre toate aceste funcționalități, în proiect am utilizat Custom Component. După cum îi spune și numele, acestă funcționalitate îi dă posibilitatea dezvoltatorului să își creeze propria componentă.
Pentru realizarea chatbot-ului am creat trei astfel de componente

- Alegere cu un singur răspuns
- Alegere multiplă
- Introducere câmp

Alegerea unui singur răspuns am utilizat-o pentru completarea câmpurilor Occupation (Student, Employee, Freelancer) și Gender (Female, Male, Both).

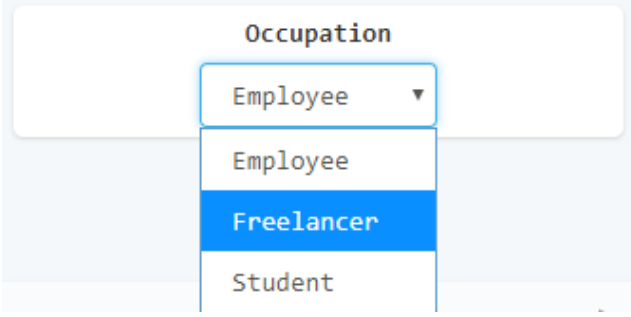


Fig 5.30 Alegere simplă

Alegerea multiplă este folosită pentru câmpurile: Music (17 variante), Languages (English, French, German), Hobbies (Music, Film, Sport, Art), Tourists Attractions (9 variante).Pentru realizarea acestei componente am folosit pachetul react-select, care imi permite selectarea mai multor variante de răspuns.

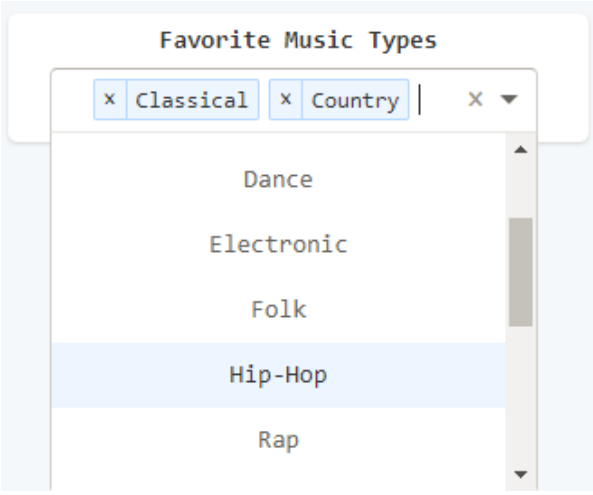


Fig 5.31 Alegere multiplă

Componenta introducere câmp este utilizată pentru Vârstă.

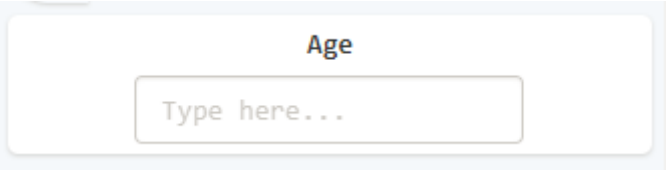


Fig 5.32 Introducere câmp

În momentul în care utilizatorul completează toate input-urile si acționează butonul Your Guide va apărea numele ghidului potrivit, în cazul în care există în baza de date o persoană care să respecte criteriile minime (locație, cel puțin o limbă comună și genul în cazul în care utilizatorul are o anumită preferință).

Restul răspunsurilor contează pentru a crea un procentaj asociat fiecărui potențial ghid. Acest procentaj este creat pe baza procentelor rezultate în urma chestionarului prezentat mai sus. Procentajul se calculează astfel:

```
procentage = 0.04 * nrAttractions + 0.052 * nrHobbies + 0.16 * nrAge + 0.007 * nrMusic + 0.14 * nrOccupation;
```

Fig 5.33 Calculare procentaj ghid

- nrAge este 1(diferența de vârstă între ghid și călător este mai mică sau egală cu 10 ani)
- nrAge este 0(diferența de vârstă între ghid și călător este mai mare de 10 ani)
- nrOccupation este 1 (ghidul și călătorul au aceeași ocupație)
- nrOccupation este 0 (ghidul și călătorul nu au aceeași ocupație)
- nrAttraction, nrHobbies, nrMusic reprezintă numărul de atracții, hobby-uri, respectiv tipuri de muzică pe care ghidul și călătorul le au în comun.

Ghidul potrivit al unui utilizator este cel care respectă cele trei cerințe minime și al cărui procentaj calculat mai sus este cel mai mare.
De exemplu:

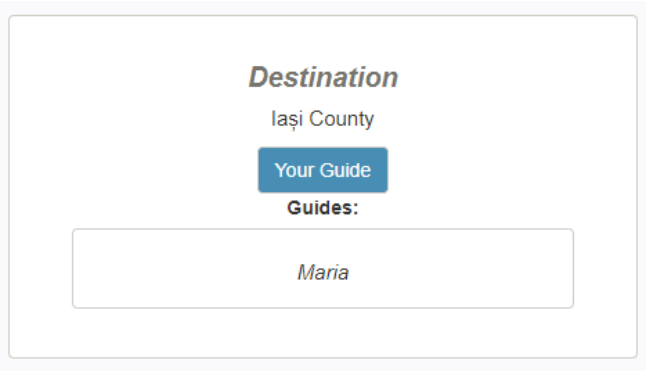


Fig 5.34 Ecran Chatbot – alegere ghid

5.8 Chat

După selectarea ghidului propus, se pornește o conversație cu acesta.Primul mesaj va fi default al ghidului.

Atunci când începe o conversație se creează automat un alt nod în baza de date – messages.

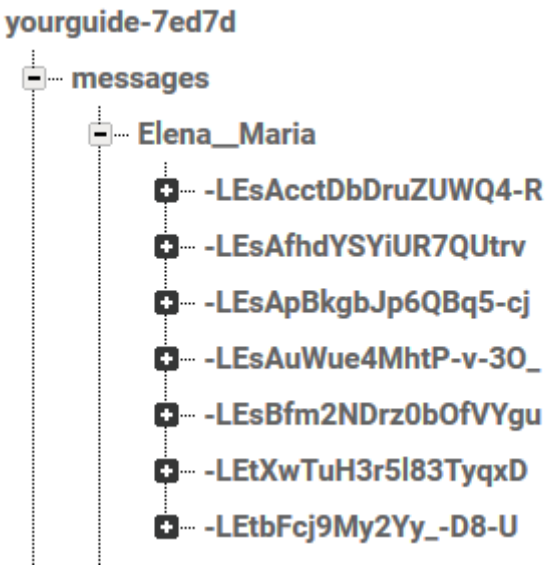


Fig 5.35 Firebase – baza de date

Numele copiilor nodului messages sunt formate din “nume călător__nume ghid”.

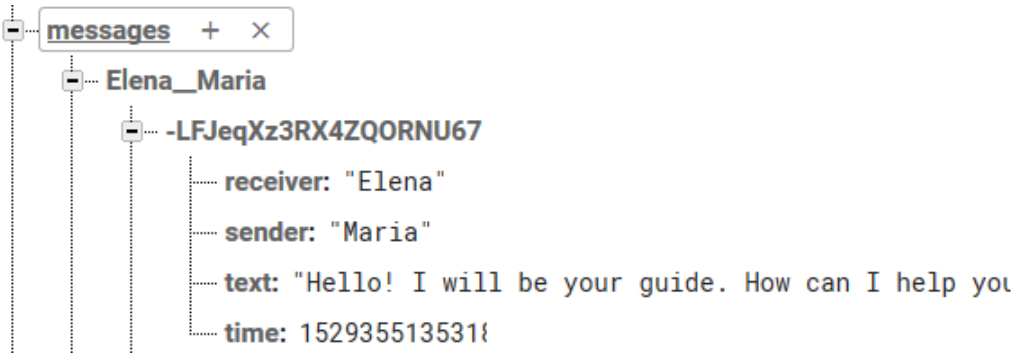


Fig 5.36 Firebase – Baza de date

Fiecare mesaj conține următoarele informații: cine este emițătorul, receptorul, textul propriu-zis si timpul la care a fost transmis. Timpul este determinat tot cu ajutorul serviciului Firebase: firebase.database.ServerValue.TIMESTAP.

În cazul în care utilizatorul are aceeași destinație, parcurge încă o dată etapa de completare a întrebărilor puse de Chatbot și pornește o altă conversație cu un alt ghid față de cel curent, atunci conversația cu ghidul curent se va șterge din lista de conversații a ambilor, inclusiv din baza de date.

Dacă din meniu se selectează opțiunea Chat, atunci utilizatorul vor fi direcționat pe pagina de conversații. Utilizatorii din această listă sunt cei cu care user-ul curent are deschise conversații, indiferent dacă acestea sunt deschise de el sau de alte persoane (utilizatorul curent este ghid).

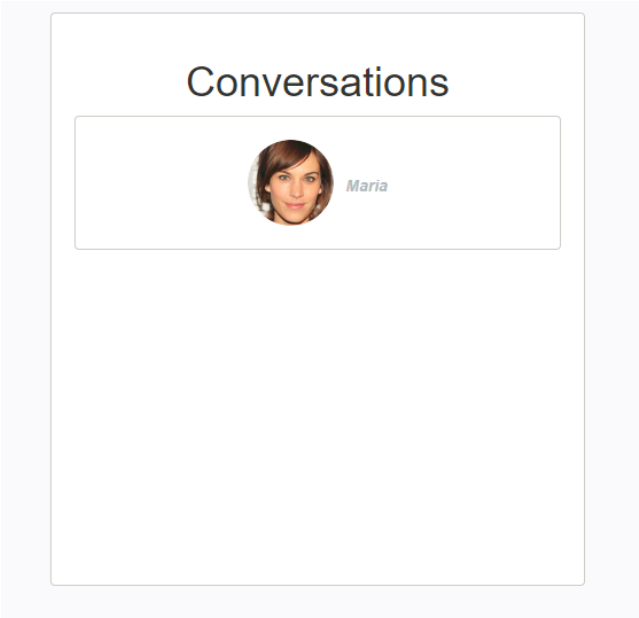


Fig 5.37 Ecran listă conversații

Ecranul prezintă numele utilizatorilor, dar si imaginea de profil.

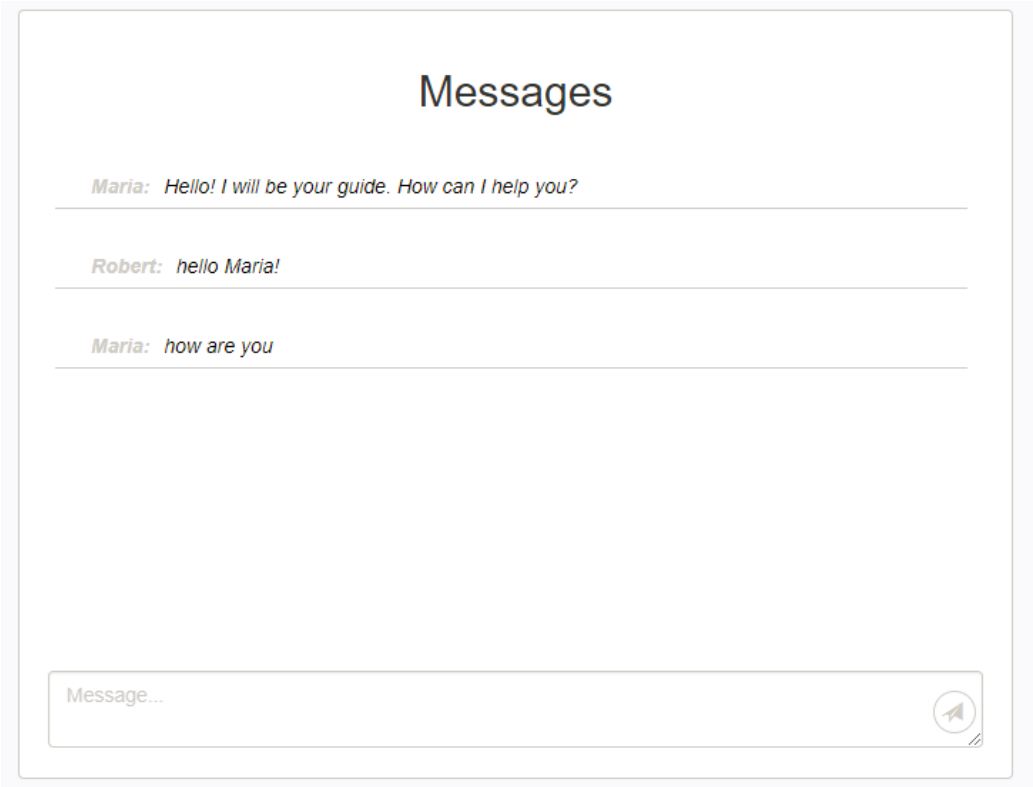


Fig 5.38 Ecran chat

Transmiterea mesajelor se face în timp real, fără să necesite timp de aşteptare. Acestea sunt într-o primă fază introduse în baza de date, apoi imediat retrimise către intefață.

5.9 Căutare utilizatori

Fig 5.39 Ecran căutare utilizatori

Căutarea utilizatorilor se realizează după username-ul din baza de date.

```
let filteredUsers = [];
for(let i=0; i<this.state.usernames.length; i++) {
    if(this.state.usernames[i].toLowerCase().indexOf(this.state.search.toLowerCase())!==-1) {
        filteredUsers[i] = this.state.usernames[i]
    }
}
```

Fig 5.40 Căutare în funcție de numele utilizatorilor

Se pot introduce în input atât litere mari, cât si litere mici, indiferent de poziția acestora în cuvânt.

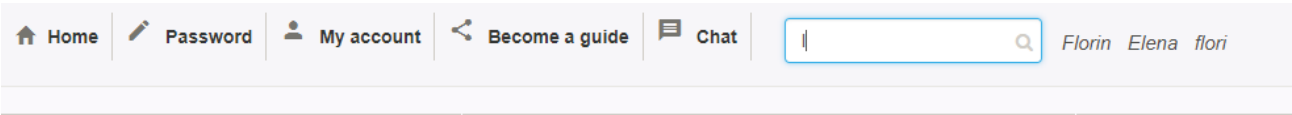


Fig 5.41 Căutare utilizator în funcție de litera tastată

5.10 Profil utilizator

Accesarea numelui căutat mă va duce pe o rută ce are ca și parametru numele utilizatorului.

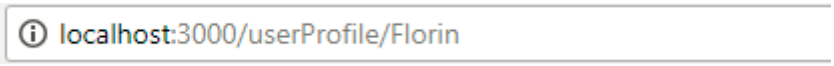


Fig 5.42 Rută - parametru

Am realizat acest lucru în codul de mai jos, utilizând instrumentele de mai sus: Link – react-router-dom și props – React.

```
const WrappedLink = (props) => {
    return (
        <Link className="no-decoration"
            to={{
                pathname: `/userProfile/${props.username}`,
                username: props.username
            }}
        >
        <p className="margin-top-minus">{props.username}</p>
        </Link>
    )
}
```

Fig 5.43 Link - parametru

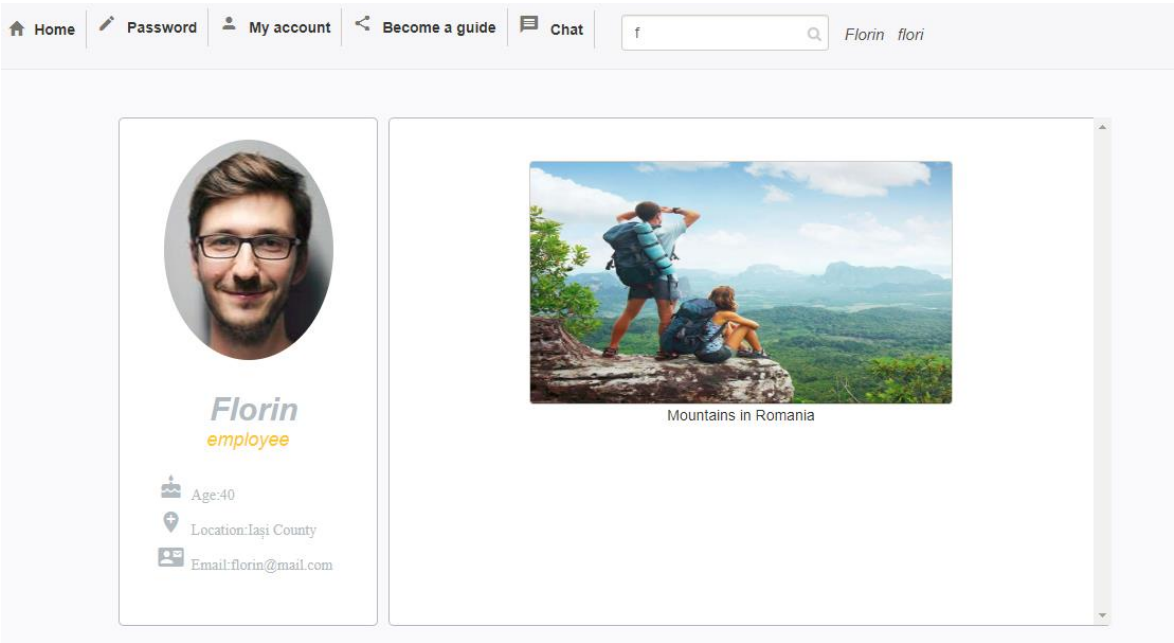


Fig 5.44 Rezultatul căutării

Acest ecran prezintă câteva informații despre utilizator, dar și pozele cu descrierile asociate pe care acesta le-a adăugat.

Informațiile despre utilizator sunt:

- Numele
- Ocupația
- Vârsta
- Locația
- Email-ul

Pentru a afișa pozele cu descrierile am construit un array, fiecare iterație a acestui array având două proprietăți: image și description.

```
arrayImages.push({"image": childSnapshot.val().image, "description": childSnapshot.val().description});
```

Fig 5.45 Creare array

Pentru a itera prin acest array creat am folosit funcția map():

```
generateImages() {
    return(
        this.state.images.map((image, key) => {
          return(
            <div>
                <div>
                    <img className="picture" src={image.image}/>
                </div>
                <div>{image.description}</div>
            </div>
          );
        })
    )
}
```

Fig 5.46 Utilizare funcție map()

Spre deosebire de ecranul anterior, profilul utilizatorului logat conține un buton de uploadare poză, adăugare descriere și stergere imagine alături de descriere.

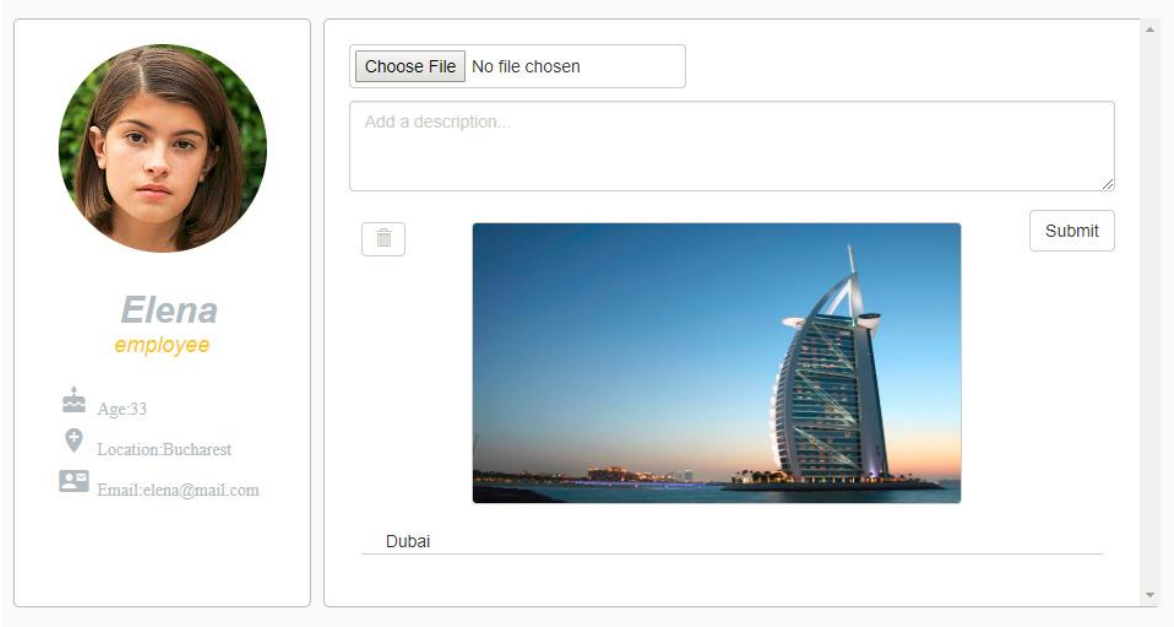


Fig 5.47 Ecran profil utilizator logat

Imaginile se adaugă în câmpul images, care este un nod copil al user-ului.

Fig 5.48 Firebase - Baza de date

6. Concluzii și dezvoltări ulterioare

6.1 Concluzii

Aplicația YourGuide este un instrument care vine în ajutorul călătorilor, ce le oferă un suport prin alegerea unei persoane compatibile. În modul acesta, călătorii primesc sfaturi, opinii, ce îi pot ajuta să își organizeze mai bine vacanța și să profite de orice informație în plus oferită, evitând astfel neplăcerile care pot apărea în orice călătorie.

Ca orice instrument nou apărut, trebuie încercat pentru a ne da seama dacă are succes sau nu. Aplicația, în stadiul de față poate fi folosită, datorită acestui lucru este și urcată pe server, iar lumea poate avea acces la ea. Există însă multe îmbunătățiri și adăugiri pe care le voi trata în subcapitolul următor.

Pentru realizarea acestei aplicații am folosit mai diferite tehnologii web, majoritatea fiind în totalitate noi pentru mine, atât Firebase, cât și React. Am ales să folosesc serviciul Firebase datorită ușurinței cu care se face configurarea și legatura cu clientul.

6.2 Dezvoltări ulterioare

Aplicația necesită multe îmbunatăți viitoare:

- Serviciul de notificări pentru chat, în momentul în care utilizatorul nu se află pe browser sau pe pagina aplicației, acesta să poată fi informat dacă primește noi mesaje. Firebase ne ajută inclusiv aici, prin serciviul Cloud Messaging.

- Chat-ul să aibă inclusă opțiunea de a transmite imagini, utilizatorul să poată atașa documente sau să poată da share locației.

- Utilizatorul să își poată modifica comentariile.

- Utilizatorul să poată grupa pozele, comentariile în funcție de excursie.

- În momentul în care numărul utilizatorilor va crește, căutarea locației să fie una mai restrânsă, adică să nu rămână la nivel de județ, ci să se extindă la nivel de oraș / localitate.

- Utilizatorul, la finalul experienței cu ghidul ales, să poată da recenzie pentru serviciile oferite, astfel încât conținutul profilului ghidului să conțină o medie a recenziilor primite. În acest fel, se încurajează implicarea din partea ghizilor, dar și o informație relevantă pentru viitori utilizatori cu care ghidul va intra în contact.

- Trecerea la Progressive Web Apps, cu ajutorul acestei tehnologii aplicația va fi disponibilă pe orice dispozitiv. Deși aplicația este una responsive,

PWA aduce în plus și pictograma pe ecranul dispozitivului. Accesarea acesteia ne duce pe browser.

- Includerea unor avantaje pe care ghizii le pot avea în cazul în care, după un anumit număr de clienți, ghidul obține un scor mare.

7. Bibliografie

[1] Typescript, React și Redux. Să alegem cele mai bune tool-uri pentru aplicaţiile web – Iunie 2018

https://www.todaysoftmag.ro/article/2117/typescript-react-si-redux-sa-alegem-cele-mai-bune-tooluri-pentru-aplicatiile-web

[2] Firebase – Aprilie 2018
https://firebase.google.com/docs

[3] Înțelegerea capabilităților tehnologiilor de comunicare web de astăzi - Iunie 2018
https://medium.com/platform-engineer/web-api-design-35df8167460

[4] Real Time Streaming with WebSocket – Iunie 2018
https://happycodingbox.blogspot.com/2017/01/real-time-streaming-with-websocket.html

[5] Basic intro to React Router v4 – Iunie 2018
https://medium.com/@thejasonfile/basic-intro-to-react-router-v4-a08ae1ba5c42

[6] Google Maps Platform – Aprilie 2018
https://developers.google.com/maps/documentation/javascript/get-api-key

[7] 5 tips for Firebase Storage – Iunie 2018
https://firebase.googleblog.com/2016/07/5-tips-for-firebase-storage.html

[8] React Simple Chatbot – Mai 2018
https://lucasbassetti.com.br/react-simple-chatbot/

[9] Bootstrap – Aprilie 2018
https://getbootstrap.com/docs/4.0/components/dropdowns/

[10] Font Awesome – Mai 2018
https://www.w3schools.com/icons/fontawesome_icons_intro.asp

[11] Google Icons – Mai 2018
https://www.w3schools.com/icons/google_icons_intro.asp

[12] Stilizare react-select – Aprilie 2018
https://npmcdn.com/react-select@1.0.0-beta13/dist/react-select.css

[13] Node.js NPM – Aprilie 2018
https://www.w3schools.com/nodejs/nodejs_npm.asp

[14] Create React App – Aprilie 2018
https://reactjs.org/docs/add-react-to-a-new-app.html

[15] What are NPM, Yarn, Babel, and Webpack; and how to properly use them? – Iunie 2018
https://medium.com/front-end-hacking/what-are-npm-yarn-babel-and-webpack-and-how-to-properly-use-them-d835a758f987

[16] Flexbox – Mai 2018
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

[17] ReactJS – JSX – Aprilie 2018
https://www.tutorialspoint.com/reactjs/reactjs_jsx.htm

[18] All you need is React and Firebase – Aprilie 2018
https://www.codementor.io/yurio/all-you-need-is-react-firebase-4v7g9p4kf

[19] Intro into React and Firebase – Aprilie 2018
https://css-tricks.com/intro-firebase-react/

[20] User Authentication – Aprilie 2018
https://css-tricks.com/firebase-react-part-2-user-authentication/

[21] A complete Firebase in React Authentication Tutorial – Aprilie 2018
https://www.robinwieruch.de/complete-firebase-authentication-react-tutorial/