

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**LET'S TRAVEL - Aplicație web de planificare a
călătoriilor**

propusă de
Gabriela Ivașcu

Sesiunea: *Iulie, 2019*

Coordonator științific
Prof. Colab. Florin Olariu

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

LET'S TRAVEL - Aplicație web de planificare a călătoriilor

Gabriela Ivașcu

Sesiunea: *Iulie, 2019*

Coordonator științific
Prof. Colab. Florin Olariu

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Let's Travel – aplicație web de planificare a călătoriilor”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

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

Iași, data

Absolvent Prenume Nume

(semnătura în original)

CUPRINS

Introducere.....	6
1.1 Motivație.....	6
1.2 Introducere în temă.....	6
1.3 Structura lucrării.....	7
Contribuții.....	8
Capitolul 1 - Descrierea problemei.....	9
Capitolul 2 - Abordări anterioare.....	10
Capitolul 3 - Descrierea soluției.....	13
3.1 Descrierea proiectului.....	13
3.2 Arhitectura proiectului.....	14
3.3 Implementarea proiectului.....	17
3.3.1 Înregistrare și autentificare.....	17
3.3.2 Stocarea datelor cu Firebase Realtime DB.....	18
3.3.3 Pagina principală(dashboard).....	20
3.3.4 Pagina de planificare a călătoriei.....	23
3.3.5 Timeline cu planul final.....	29
3.3.6 Interfața aplicației.....	30
Concluziile lucrării.....	32
1. Concluzii.....	32
2. Direcții de viitor.....	32
Bibliografie.....	33
Anexe.....	34

INTRODUCERE

1. Motivație

Ideea aplicației a provenit din dorința de a avea o metoda mai eficienta de organizare a unei excursii.

Călătoritul este o activitate foarte îndrăgită de fiecare dintre noi dar nu și partea în care trebuie să o organizăm. De asemenea, având activitățile deja planificate, plecăm în călătorii fără stres și fără a pierde timpul pe mijloace de căutare pentru idei de activități și recomandări de localuri, restaurante, obiective turistice.

O altă problemă este faptul că există o mulțime de aplicații ce oferă servicii de călătorii dar puține aplicații ce oferă un mediu de planificare împreună cu recomandări și idei de activități.

2. Introducerea în temă



Figura 1: Logo-ul aplicației

Lucrarea realizată constă într-o aplicație web ce ajută utilizatorul la organizarea propriei vacanțe. Aceasta se adresează călătorilor care nu vor să piardă timpul prețios din vacanță și să aibă activitățile planificate din timp. Rolul aplicației este să ofere idei de activități și în același timp un mediu în care să le poți organiza și vizualiza într-un mod cât mai simplu pentru utilizator. În plus, oferă posibilitatea de a păstra într-un singur loc toate planurile pentru călătoriile următoare și cele din trecut.

3. Structura lucrării

Lucrarea conține trei capitole generale în care sunt detaliate mai multe informații despre lucrare.

În **primul capitol** este descrisă problema întâmpinată în procesul de planificare a unei excursii.

În **al doilea capitol** sunt prezentate câteva aplicații deja existente ce au abordat problema descrisă în Capitolul 1 și cu ce rezolvări au venit.

În **al treilea capitol** este descrisă soluția oferită problemei, cu arhitectura proiectului alături de funcționalitățile principale și implementarea acestora.

CONTRIBUȚII

În momentul de față este plin de aplicații web ce oferă servicii de călătorit, iar pentru ca aplicația sa fie una din cele de top, trebuie sa fie eficientă, să ofere mai multe funcționalități și cu o interfață cât mai ușor de folosit pentru utilizator.

În plus, pentru realizarea acestui proiect am vrut sa folosesc tehnologii moderne care sa fie cât mai eficiente în realizarea proiectului.

A fost nevoie de o perioadă de căutare a unor API-uri potrivite, care sa ofere cât mai multe detalii despre locații din locul în care utilizatorul dorește să călătorească pentru a face recomandări cât mai precise. După multe căutări de API-uri care pot fi folosite în scop necomercial, am găsit ca soluție cel de la Foursquare Places, descris în Anexa 6 cum funcționează. Ca alternative fiind TripAdvisor dar ce poate fi folosit doar în scopul comercial sau Triposo.

În restul aplicației m-am gândit sa folosesc tehnologiile oferite de Google, cum ar fi Firebase pentru partea de server din Anexa 2, Angular pentru partea de client din Anexa 1, Material pentru interfață și API-urile din cadrul serviciului de Google Maps descrise în Anexa 7, astfel ele având același dezvoltator se îmbină perfect.

CAPITOLUL 1 - DESCRIEREA PROBLEMEI

Aplicațiile de călătorii sunt foarte multe și diverse, dar în momentul de față nu există o aplicație ce să cuprindă toate aceste servicii la un loc.

Călătoritul este o activitate foarte îndrăgită de populație dar nu și partea în care trebuie să o organizăm, să pierdem ore în șir pentru a găsi idei de activități, recomandări pentru cel mai bun loc în care să luăm cina sau cel mai bun traseu de parcurs. În plus, nimeni nu vrea să piardă timpul prețios din vacanțe cautând locații și ar prefera să aibă toate aceste lucruri deja organizate înainte de plecare.

Problema principală a aplicațiilor existente în momentul de față este că puține oferă un mediu în care poți planifica și primi recomandări de locații, în plus să aibă și traseele exacte pe hartă în funcție de modul de deplasare.

CAPITOLUL 2 - ABORDĂRI ANTERIOARE

Google Trips

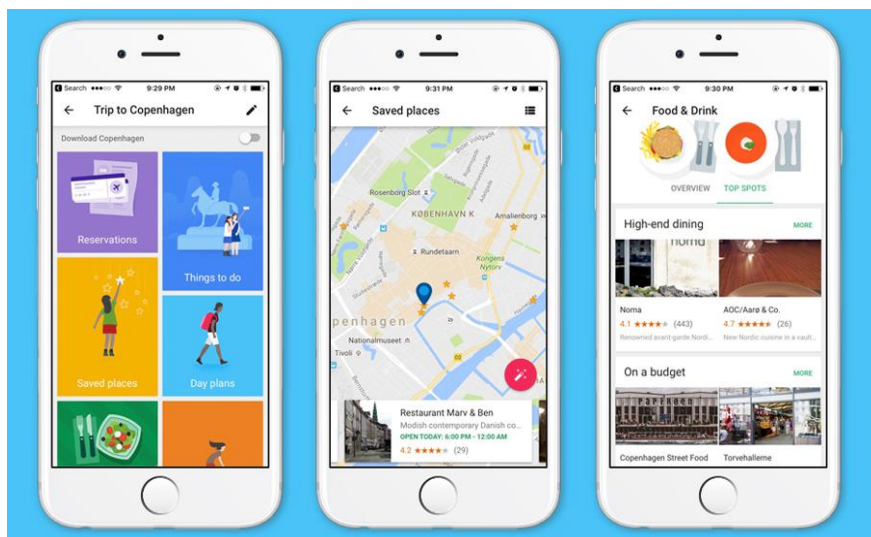


Figura 2: Aplicația actuală Google Trips [1]

Această aplicație este cea mai completă din punctul meu de vedere pe tema aceasta. Practic, ai accesul la toate informațiile pe care Google le are la dispoziție în acest domeniu. Primești recomandări de locații, excursii, restaurante și transport. Până în momentul actual, nu există aplicație care să ofere gratuit toate aceste beneficii. Aplicația existentă este pentru Android/iOS.

Începând cu data de 5 august 2019, google nu mai oferă suport pentru aceasta deoarece au lansat un nou serviciu de călătorii, disponibil pe <https://www.google.com/travel>.

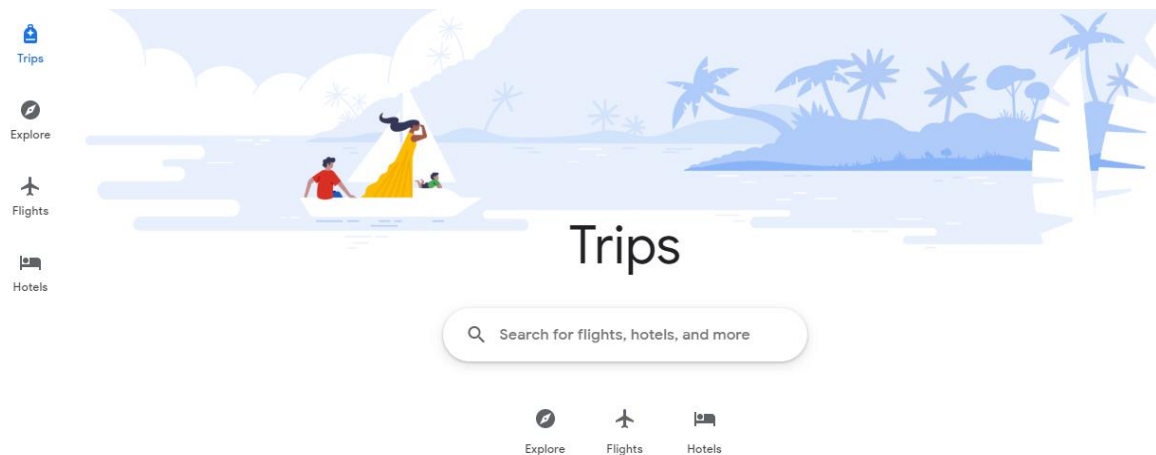


Figura 3: Noul serviciu oferit de Google pentru călătorii [2]

Triplt

Oferă posibilitatea de a planifica o excursie pe zile și ai o varietate de posibilități pentru transport și activități. Nu oferă recomandări de localuri, iar accentul este pus pe transport, traseu, bilete și rezervări.

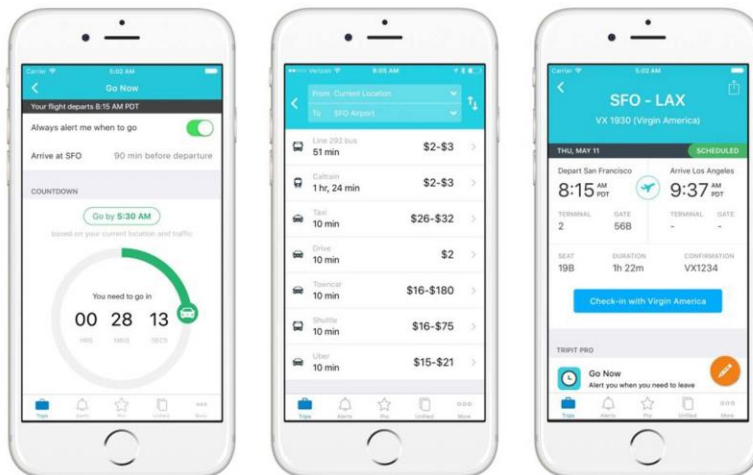


Figura 4: Aplicația Triplt pe mobil [3]

Inspirock

Oferă posibilitatea de a planifica o vacanță pe zile, de asemenea îți generează tururi pentru fiecare zi, la care tu poți adăuga și șterge atracții. Vine cu recomandări de activități și obiective

turistice. De asemenea oferă informații utile cum ar fi prețurile, orele de deschidere, trasee. Dezavantajul este că nu oferă recomandări pentru restaurante/localuri.

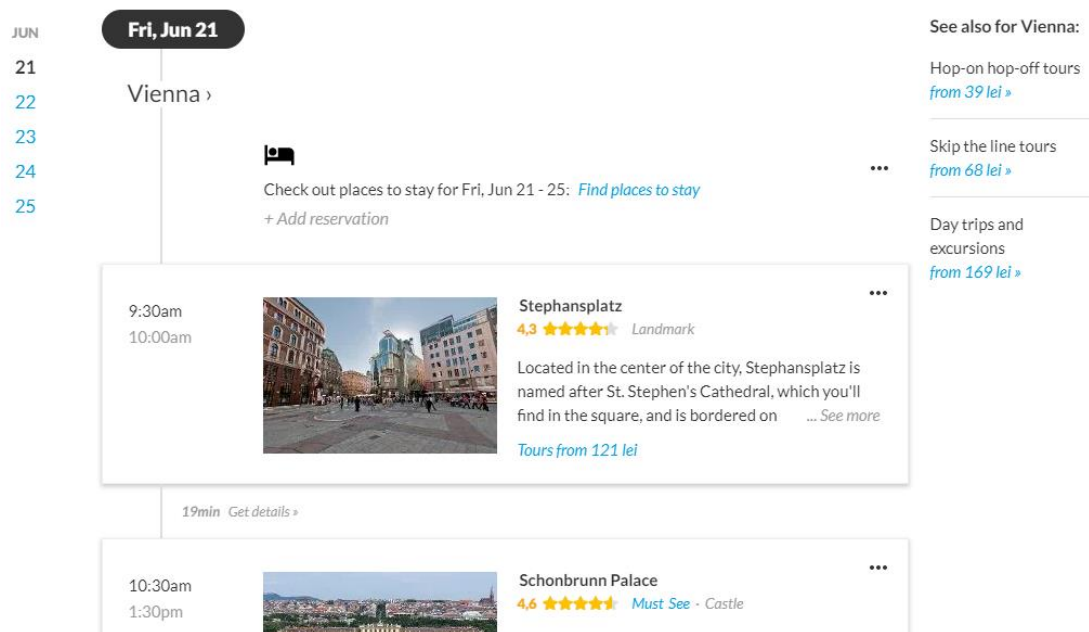


Figura 5: Timeline-ul oferit de Inspirock [4]

CAPITOLUL 3 - DESCRIEREA SOLUȚIEI

3.1 Descriere proiect

Proiectul realizat este o aplicație web în care un oarecare utilizator cu acces la internet își poate planifica un program pentru excursia dorită.

Cum este exemplificat și în Figura 6 de mai jos, modul de funcționare a aplicației este:

Un utilizator oarecare intra pe website-ul proiectului și se autentifică. După logare, utilizatorul este direcționat pe dashboard-ul personal și are posibilitatea de a își crea planul pentru următoarea excursie. După completarea destinației și datele corespunzătoare, acesta își poate crea un program pentru fiecare zi. Aplicația oferă o gamă largă de recomandări precise din locul în care călătorește. Recomandările sunt din zona de restaurante, obiective turistice și magazine. Pe lângă aceste recomandări, utilizatorul poate să adauge zborul sau hotelul la care are rezervare dar și activități personalizate care nu se află în cele prezente în aplicație. În plus, are posibilitatea de a își căuta un loc anume.

La final utilizatorul poate vizualiza locațiile împreună cu traseele alternative pe harta oferită de Google Maps în funcție de modul de deplasare, cum ar fi mersul pe jos, cu mașina sau bicicleta. De asemenea, activitățile se regăsesc și într-o listă în care poate adăuga/șterge activități și a schimba ordinea lor. După organizarea fiecărei zile, va avea o sumarizare într-o cronologie pentru tot parcursul vacanței și detalii cu adresa obiectivelor, distanța dintre ele și timpul estimativ de parcurs. Utilizatorul își poate vizualiza de pe pagina de dashboard toate planurile pentru vacanțele viitoare dar și cele ce au trecut.

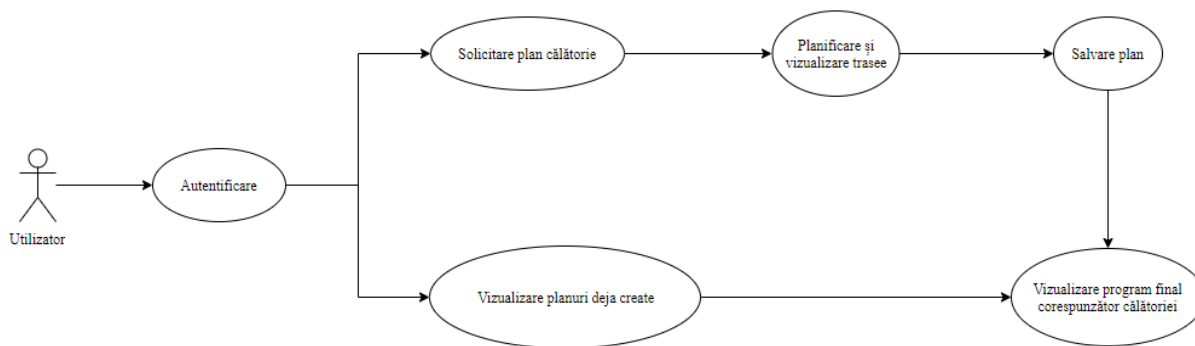


Figura 6: Diagramă UseCase a proiectului

Recomandările de localuri se realizează cu ajutorul API-ului oferit de Foursquare Places detaliat în Anexa 6. Acesta conține date cu recomandări de localuri dar și detalii despre acestea, cum ar fi numele, adresa, coordonatele, categoria etc. Ca și alternativa mai bună pentru acest API, ar fi fost cel de la TripAdvisor dar acesta poate fi folosit doar în scop comercial.

3.2 Arhitectura proiectului

În Figura 7 de mai jos este prezentat într-o diagrama contextul în care se afla aplicația. Elementele principale prezente fiind:

1. **Utilizatorul** ce poate fi oricine cu acces la internet
2. **Aplicația web** ce oferă un mediu de planificare atractiv și îmbină serviciile oferite de API-urile de mai jos
3. **Foursquare API** ce ajută ca recomandările din cadrul aplicației să fie cât mai precise
4. **Google API** ajută la integrarea hărții oferite de Google în aplicație
5. **Google Firebase** oferă serviciul de autentificarea și în plus mediul pentru stocare a datelor despre călătorii

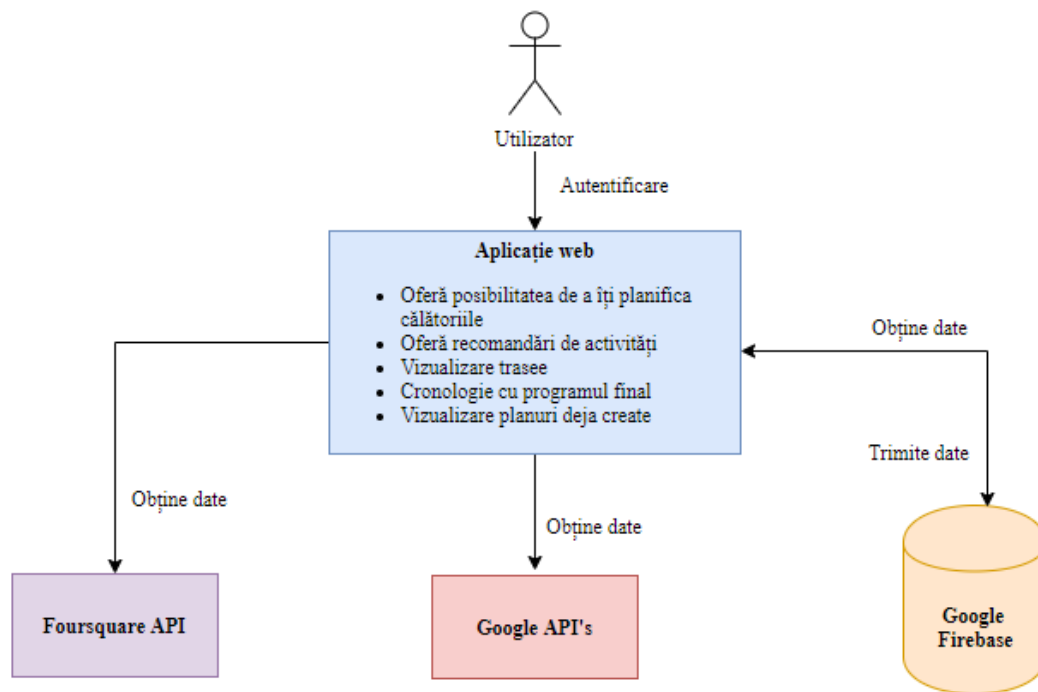


Figura 7: Diagramă contextuală

Arhitectura aplicației este prezentată mai jos în Figura 8 într-o diagrama ce cuprinde elementele din capitolul de mai sus alături de componentele principale ale aplicației.

Componentele principale ale aplicației fiind:

- Componenta de login ce se ocupă de autentificarea în aplicație
- Dashboard-ul în care utilizatorul are posibilitatea de a solicita organizarea unei excursii dar își poate și vizualiza toate planurile pe care le-a salvat
- Componenta de planificare(planner) în care are loc planificarea propriu zisă și are acces la recomandările de activități
- Componenta de timeline în care are o privire de ansamblu la toată călătoria

Iar sursele externe sunt:

- Firebase authentication
- Firebase Realtime Database
- Foursquare Places API
- Google Maps API
- Google Directions API
- Google Places API

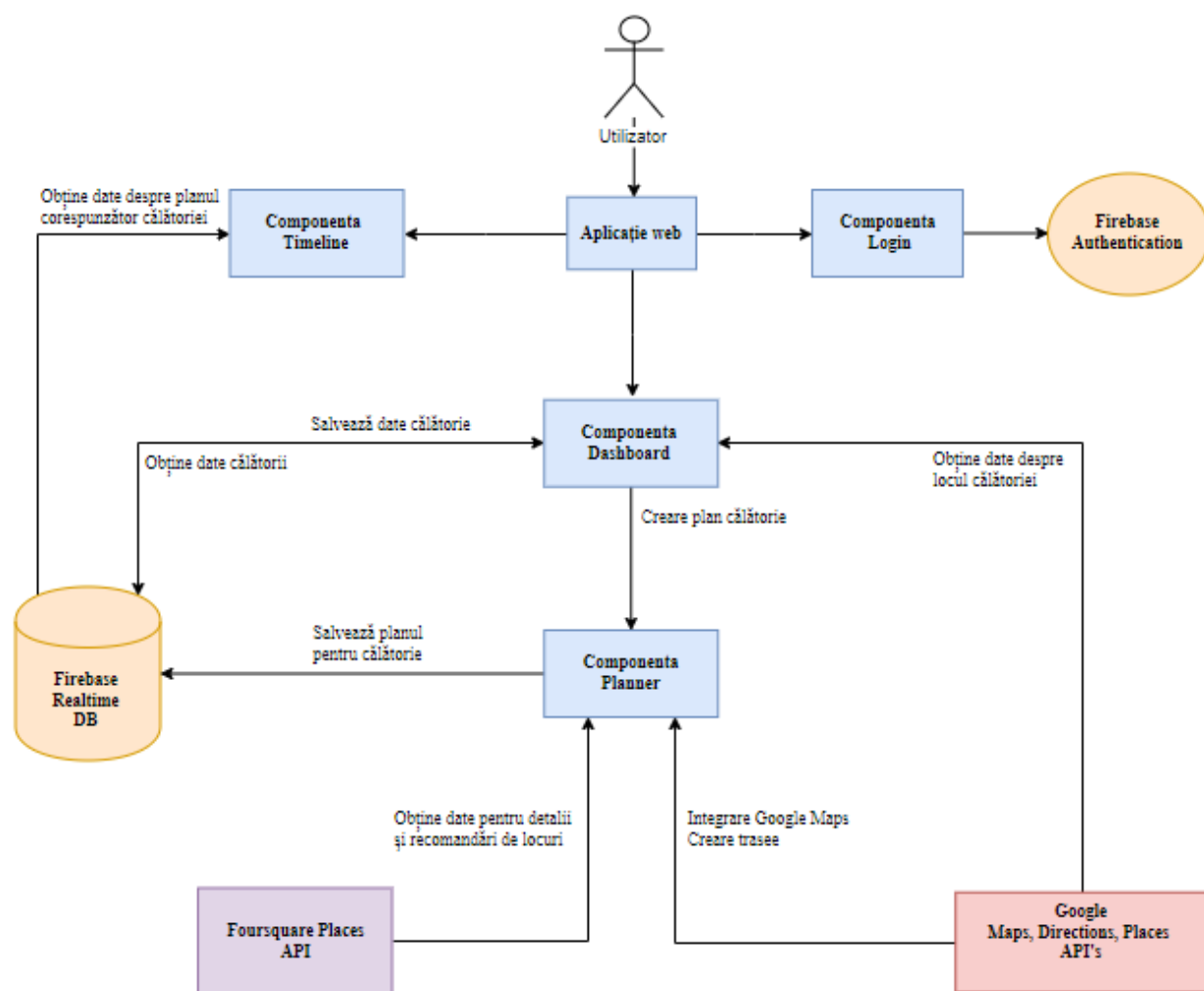


Figura 8: Arhitectura generala a aplicației

3.3. Implementarea proiectului

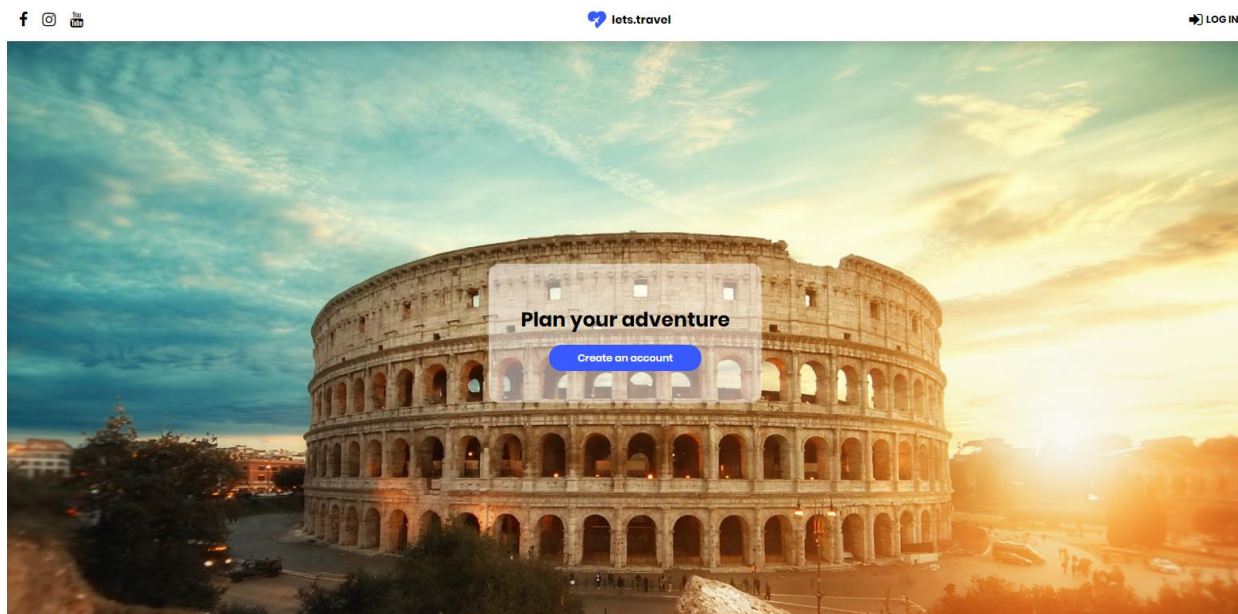


Figura 9: Pagina de pornire a aplicației

3.3.1 Înregistrare și autentificare

După cum se poate vedea în Figura 9, primul pas al utilizatorului este de a se autentifica. Dacă utilizatorul nu are un cont creat, acesta se poate înregistra completând un formular cu email și parolă sau se poate loga direct cu contul de Google.

Autentificarea se realizează utilizând serviciul de autentificare de la Firebase ce este descris și în Anexa 4. Apelarea serviciului de autentificare se poate vedea în Figura 10 de mai jos.


```

41
42   doRegister(value){
43     return new Promise<any>((resolve, reject) => {
44       firebase.auth()
45         .createUserWithEmailAndPassword(value.email, value.password)
46         .then(res => {
47           resolve(res);
48         }, err => reject(err))
49     })
50   }
51
52   doLogin(value){
53     return new Promise<any>((resolve, reject) => {
54       firebase.auth().signInWithEmailAndPassword(value.email, value.password)
55         .then(res => {
56           resolve(res);
57         }, err => reject(err))
58     })
59   }
60

```

Figura 10: Înregistrare și logare

Cum se poate vedea în Figura 11, direct din consola de la Firebase, se poate vizualiza tabela cu utilizatorii înregistrați în aplicație.




Identifier	Providers	Created	Signed In	User UID ↑
gabriela.ivascu95@gmail.com		Apr 30, 2019	Jun 23, 2019	ReG4K2KycmQZ
test_gabi@gmail.com		May 25, 2019	May 25, 2019	hKVply09CJUcl9
gabriela12@gmail.com		Jun 3, 2019	Jun 3, 2019	w3QMvPlssUUV:

Figura 11: Tabela din consolă de la Firebase cu utilizatorii înregistrați

3.3.2 Stocarea datelor cu Firebase Realtime DB

Firebase Realtime Database este o baza de date NoSQL găzduită pe cloud, care permite stocarea și sincronizarea datelor în timp real.

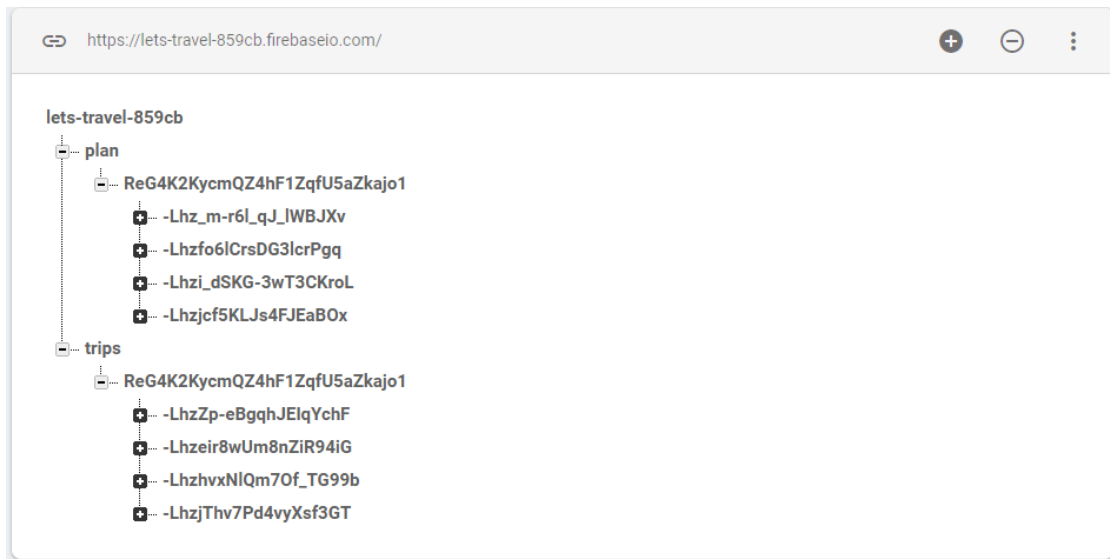


Figura 12: Baza de date de la Firebase

După cum se poate vedea în Figura 12 de mai sus, în baza de date din Firebase am stocat călătoriile și planurile pentru ele, fiecare având asociat id-ul user-ului autentificat.

```

.....endDate: "Thu Jun 06 2019 00:00:00 GMT+0300 (Eastern Eurc
.....location: "Vienna"
.....placeld: "ChIJn8o2UZ4HbUcRRluiUYrlw
.....startDate: "Wed Jun 05 2019 00:00:00 GMT+0300 (Eastern Eurc
.....tripId: "-LhzZp-eBgqhJEIqYch
  
```

Figura 13: Structura pentru călătorie din baza de date

În Figura 13 este exemplificat care sunt datele stocate pentru o călătorie, iar în Figura 14 de mai jos este reprezentată stocarea pentru planul unei călătorii.

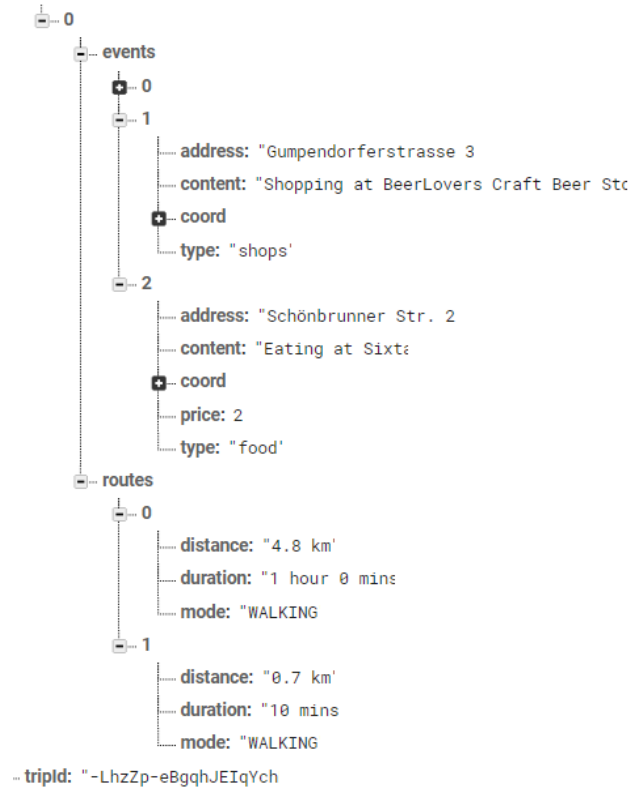


Figura 14: Structura pentru un plan de călătorie din baza de date

3.3.3 Pagina principală (dashboard)

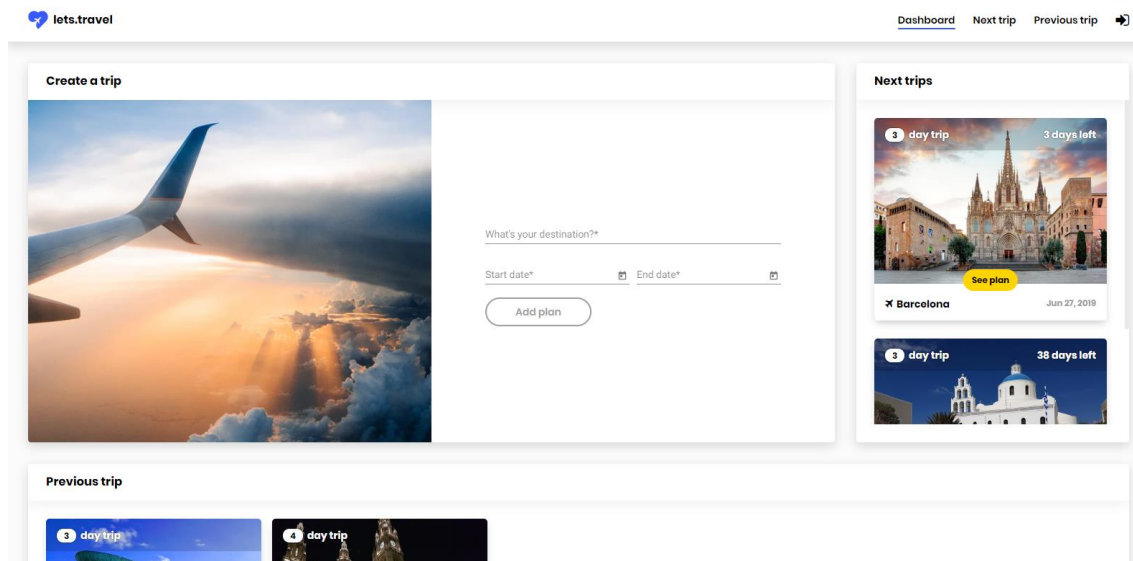


Figura 15: Dashboard

Pe pagina principală, vizibilă în Figura 15, ai posibilitatea sa creezi un plan nou prin completarea formularului. Autocomplete-ul pentru destinație vizibil în Figura 16, am folosit directiva *matGoogleMapsAutocomplete* pe componenta de input pentru a activa funcția de autocomplete oferită de API-ul Google Maps Autocomplete ce este amintit și în Anexa 9.

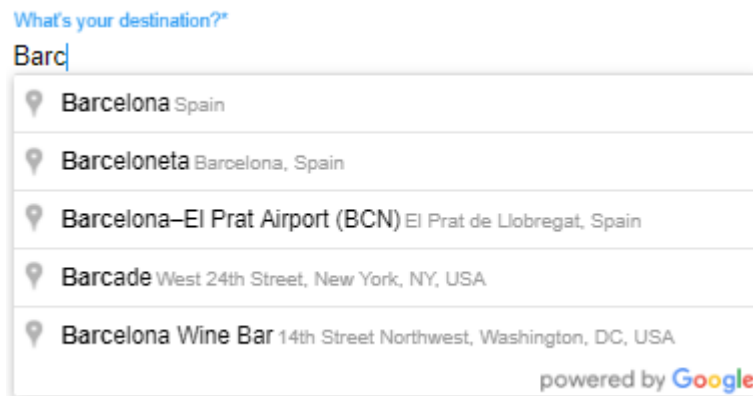


Figura 16: Autocomplete-ul oferit de Google Places API

După selectarea unui element din listă, prin evenimentele *onLocationSelected* și *onAutocompleteSelected* avem accesul la detaliile oferite de acest API pentru locații. Din aceste detalii am folosit coordonatele, numele locației și id-ul locului, după cum se poate observa și în Figura 17 de mai jos.

```
71 onLocationSelected(location: Location) {  
72   | this.formGroupTrip.patchValue({ coord: location });  
73 }  
74  
75 onAutocompleteSelected(result: PlaceResult) {  
76   | this.formGroupTrip.patchValue({ location: result.name });  
77   | this.formGroupTrip.patchValue({ placeId: result.place_id });  
78 }  
79
```

Figura 17: Evenimentele emise la autocomplete

De asemenea, pe pagina de dashboard sunt afișate și planurile pentru călătoriile următoare sau cele din trecut. Pentru afișarea pozelor pentru fiecare călătorie am folosit metoda *getDetails()* din cadrul serviciului oferit de Google Places API, cum se poate observa în Figura 18. Această metodă are ca parametru id-ul locului pentru care facem request, iar când primim răspunsul,

primim în *placeResult.photos* un vector cu 10 url-uri de poze, dar în proiect este nevoie doar de una.

```
90     let service = new google.maps.places.PlacesService(map);
91     service.getDetails({
92       placeId: placeId
93     }, (placeResult, status) => {
94       if (status == google.maps.places.PlacesServiceStatus.OK) {
95         this.photosPlace.set(placeId, placeResult.photos[3].getUrl({
96           maxWidth: 500,
97           maxHeight: undefined
98         }));
99       }
100     });
```

Figura 18: Apelare metoda `getDetails()` pentru a obține url-ul pentru pozele corespunzătoare locației

La apăsarea butonului *Add plan*, se face trimitere formularul, se creează o călătorie și utilizatorul este redirecționat la componenta de planificare, după cum se poate observa și în Figura 19.

```
116   onSubmit() {
117     let tripKey = this.firebaseService.createTrip(this.formGroupTrip.value);
118     this.startPlanService.setLocation({ value: this.formGroupTrip.value, key: tripKey });
119     this.router.navigateByUrl('/add-plan');
120   }
```

Figura 19: Evenimente declanșate la submit

Adăugarea unei călătorii în baza de date se realizează conform Figura 20, în care obținem id-ul utilizatorului curent și realizăm o nouă intrare în tabela *trips* cu datele obținute în pasul precedent.

```

32 createTrip(data) {
33   var newPostKey = firebase.database().ref().child('trips').push().key;
34   let userId = firebase.auth().currentUser.uid;
35
36   var updates = {};
37   updates['/trips/' + userId + '/' + newPostKey] = {
38     tripId: newPostKey,
39     location: data.location,
40     startDate: String(data.startDate),
41     endDate: String(data.endDate),
42     placeId: data.placeId
43   };
44   firebase.database().ref().update(updates);
45   return newPostKey;
46 }

```

Figura 20: Funcția de adăugare a unei noi călătorii

3.3.4 Pagina de planificare a călătoriei

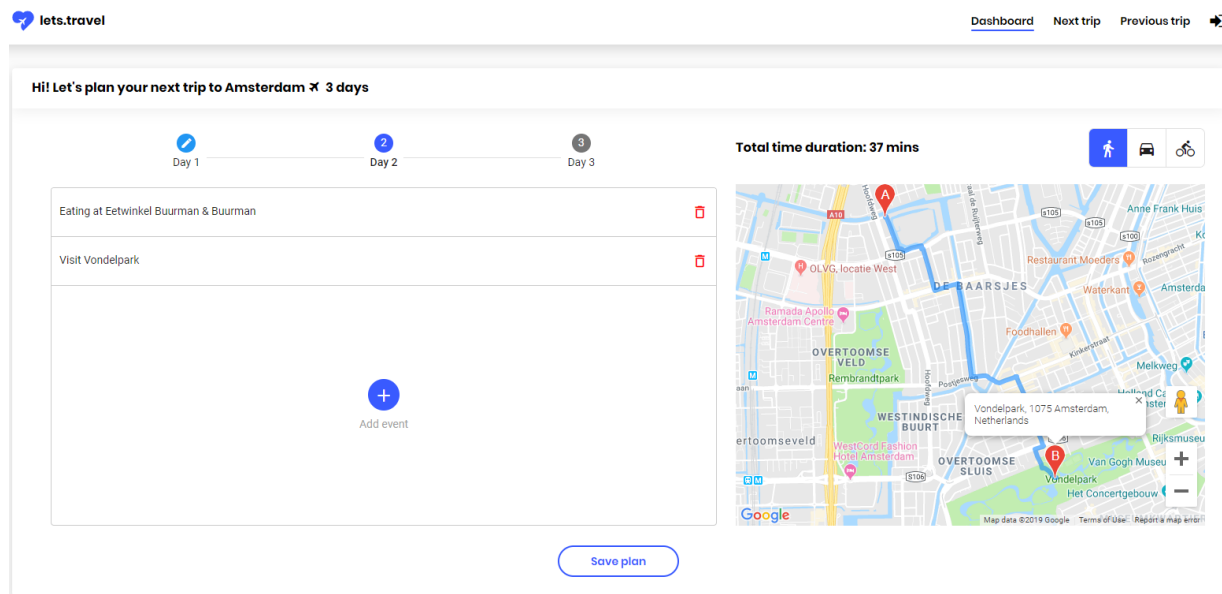


Figura 21: Pagina de planificare a călătoriei

În pagina de planificare utilizatorul poate începe să își organizeze excursia. La început acesta poate vizualiza harta cu locația și câte o listă pentru fiecare zi.

Dacă utilizatorul apasă butonul *Add event*, se deschide un modal în care poate naviga prin tab-uri alegând o activitate pe care și-o dorește. Pentru locurile de tip *food*, *outdoors*, *shops* în partea dreaptă sunt recomandările provenite din API-ul de la Foursquare discutat în detaliu în

Anexa 6. Utilizatorul are posibilitatea și să adauge un zbor sau un hotel la care are rezervare sau o altă activitate.

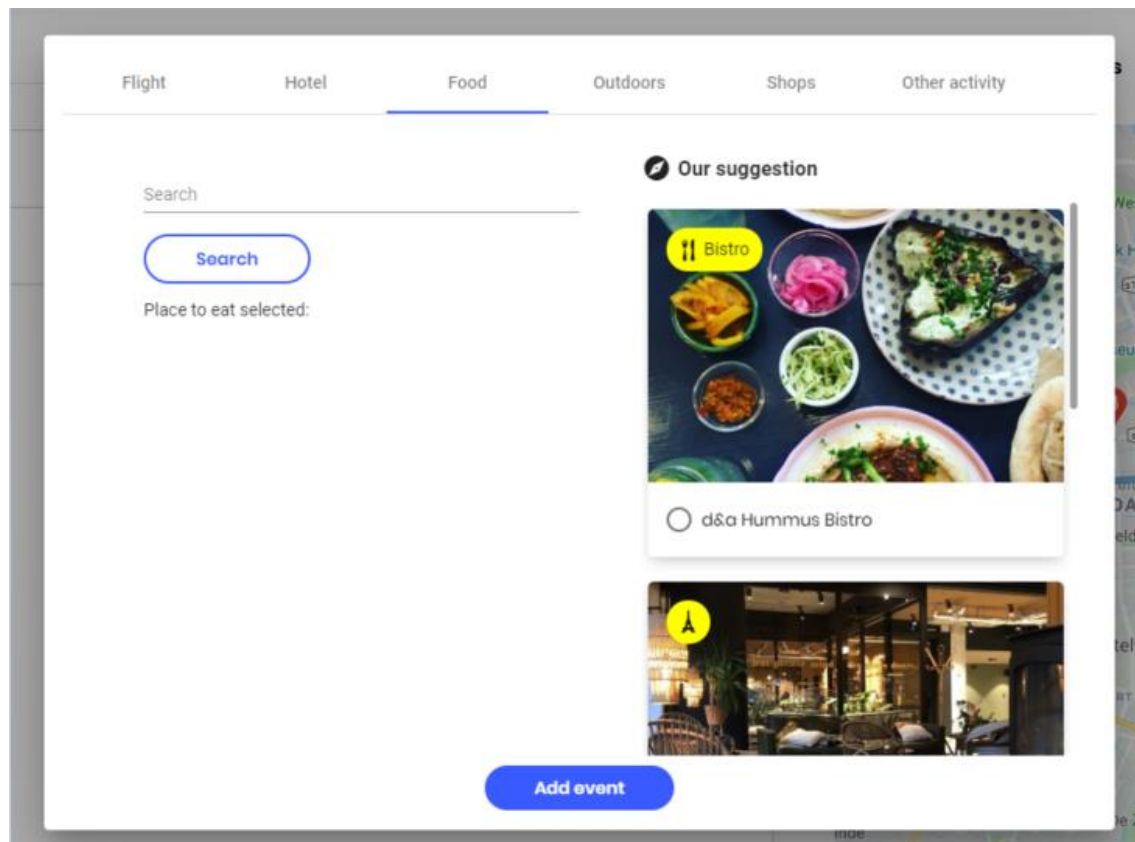


Figura 22: Modal pentru adăugare activitate

În funcție de parametrii trimiși pentru locație și tip, se face un http request la Foursquare API, exemplificat în Figura 23.

```
getPlace(city, section, limit) {  
  let url = 'https://api.foursquare.com/v2/venues/explore?' + this.key + '&limit=' + limit + '&near=' + city + '&section=' + section;  
  let place = this.http.get(url);  
  return place;  
}
```

Figura 23: Request pentru recomandări de localuri în funcție de secțiune

Pentru a obține datele cu recomandări, apelez metoda din Figura 24 cu parametrii specifici și fac subscribe la request-ul de mai sus. Când primesc datele, parsez răspunsul ce vine în format JSON exemplificat în Anexa 6. Apoi îmi creez obiectele pentru fiecare secțiune în exploreResult

ce este un obiect care conține pentru fiecare secțiune informații cum ar fi numele, id-ul, categoria din care face parte, adresa și coordonatele.

```
116   this.placesService.getPlace(this.data.location, section, 4).subscribe((result) => {
117       let exploreResult = Object(result).response.groups[0].items;
118
119       for (let i = 0; i < exploreResult.length; i++) {
120           switch (section) {
121               case "food": {
122                   this.exploreResults.food.values.push({
123                       name: exploreResult[i].venue.name,
124                       id: exploreResult[i].venue.id,
125                       category: exploreResult[i].venue.categories[0],
126                       lat: exploreResult[i].venue.location.lat,
127                       lng: exploreResult[i].venue.location.lng,
128                       address: exploreResult[i].venue.location.address
129                   });
130                   break;
131               }
```

Figura 24: Subscribe la request-ul pentru recomandari la Foursquare API

După crearea obiectului cu recomandări, pentru fiecare locație se face un request pentru poze, exemplificat în Figura 25. Apoi salvez url-urile corespunzătoare fiecărei locații într-un dicționar, având ca cheie id-ul locației și ca valoare url-ul.

```
let id = exploreResult[i].venue.id;
this.placesService.getPhotos(id).subscribe((result) => {
    let url = Object(result).response.photos.items[0].prefix + '640' + Object(result).response.photos.items[0].suffix;
    this.photosUrl.set(id, url);
});
```

Figura 25: Subscribe la request-ul pentru poze de la Foursquare API

După obținerea tuturor datelor oferite de API-ul de la Foursquare, recomandările se afișează pe pagină. Cum se poate observa și în Figura 26 de mai jos în structura clasică de HTML iar cu ajutorul sintaxei din Angular adăugăm logica. De asemenea, fiecare opțiune are un radio button pentru a avea posibilitatea de a selecta localul dorit. În plus, în partea dreaptă este calculat și timpul total al duratei de deplasare a ultimului traseu afișat.


```

54 <div class="explore">
55   <h3> <i class="material-icons">explore</i> Our suggestion</h3>
56   <ul *ngIf="exploreResults.food.values.length > 0" class="places-list">
57     <mat-radio-group aria-label="Select an option" [(ngModel)]="foodPlace">
58       <li *ngFor="let place of exploreResults.food.values; let i = index" class="place">
59         <div class="img-container">
60           
61         </div>
62         <div class="category">
63           
64           <p class="text">{{place.category.name}}</p>
65         </div>
66         <div class="select">
67           <mat-radio-button [value]="place.name">{{place.name}}</mat-radio-button>
68         </div>
69       </li>
70     </mat-radio-group>
71   </ul>
72 </div>

```

Figura 26: Structura HTML pentru afișarea recomandărilor de localuri

În partea dreapta din Figura 21, utilizatorul are posibilitatea sa facă o căutare a unui local după un text dat și dacă se găsește va fi afișat sub input o poza cu acesta, câteva detalii și va fi selectat automat. Pentru realizarea căutării unui local se face un subscribe la request-ul din Figura 27.

```

getPlacebySearch(city, text) {
  let url = 'https://api.foursquare.com/v2/venues/search?' + this.key + '&limit=1' + '&near=' + city + '&query=' + text;
  let place = this.http.get(url);
  return place;
}

```

Figura 27: Request pentru căutarea unui local în funcție de un string dat

După adăugarea unui eveniment acesta se va adăuga în lista. Elementele din lista cu evenimente din acea zi pot fi mutate sau șterse, fiind practic un vector cu evenimente.

În partea din dreapta din pagina de planificare, harta este integrată cu ajutorul API-ului de la Google Maps și când avem cel puțin două locații în lista, se va genera un traseu cu ajutorul componentei *agm-direction* ce folosește Directions API, descris în Anexa 8. Aceasta are nevoie ca date de intrare coordonate pentru locul din care pleacă și destinația, în plus poți adăuga și locații de legătura. Dacă apeși pe orice indicator din traseu, se va afișa adresa completă a locului, acestu lucru fiind vizibil în Figura 29.

În figura 28, se poate observa cum arată structura HTML pentru componenta ce integrează harta de la Google Maps și de asemenea, logica adăugată pentru indicatoare și trasee.

```

<agm-map [zoom]="zoom" [latitude]="lat" [longitude]="lng">
  <ng-container *ngIf="locationList.length < 2">
    <agm-marker *ngFor="let location of locationList" [latitude]="location.lat" [longitude]="location.lng">
    </agm-marker>
  </ng-container>
  <ng-container *ngFor="let days of coords; let i = index">
    <ng-container *ngIf="days.length > 1">
      <agm-direction [origin]="days[0]" [destination]="days[days.length - 1]" [waypoints]="waypoints[i]"
        [travelMode]="travelMode" (onResponse)="onResponse($event)">
      </agm-direction>
    </ng-container>
  </ng-container>
</agm-map>

```

Figura 28: Structura HTML pentru harta cu trasee

Dacă avem mai puțin de o locație în ziua respectivă, va apărea doar un indicator. În plus, utilizatorul poate schimba modul de calcul al traseului prin butoanele din dreapta sus, inițial fiind setat pe modul de mers pe jos.

În figura 29 de mai jos, avem exemplificat cum arată situația în care avem într-o zi un singur obiectiv iar într-o altă zi un traseu între doua puncte. Modul de calcul fiind pentru mersul pe jos.

Pe lângă acestea, în partea stângă de sus este afișată o estimare pentru timpul total al ultimului traseu selectat.

Total time duration: 68 mins



Figura 29: Traseele pe hartă

Când utilizatorul termină de planificat, acesta are posibilitatea de a salva planul prin apăsarea butonului *Save plan*. La salvare acestuia se apelează metoda *savePlan* din Figura 30, în care se apelează *createPlan* unde trimite ca parametrii obiectul cu evenimente și cheia călătoriei.

```
114     savePlan() {  
115         this.firebaseService.createPlan(this.allEvents, this.key);  
116         this.router.navigate(['/timeline', this.key]);  
117     }
```

Figura 30: Metoda savePlan()

3.3.5 Timeline cu planul final

După salvarea unui plan suntem redirecționați către timeline-ul cu programul final din Figura 31. Pentru acesta am folosit pachetul *angular-mgl-timeline* ce oferă ca componentă un timeline vertical pentru proiectele de Angular 2+.

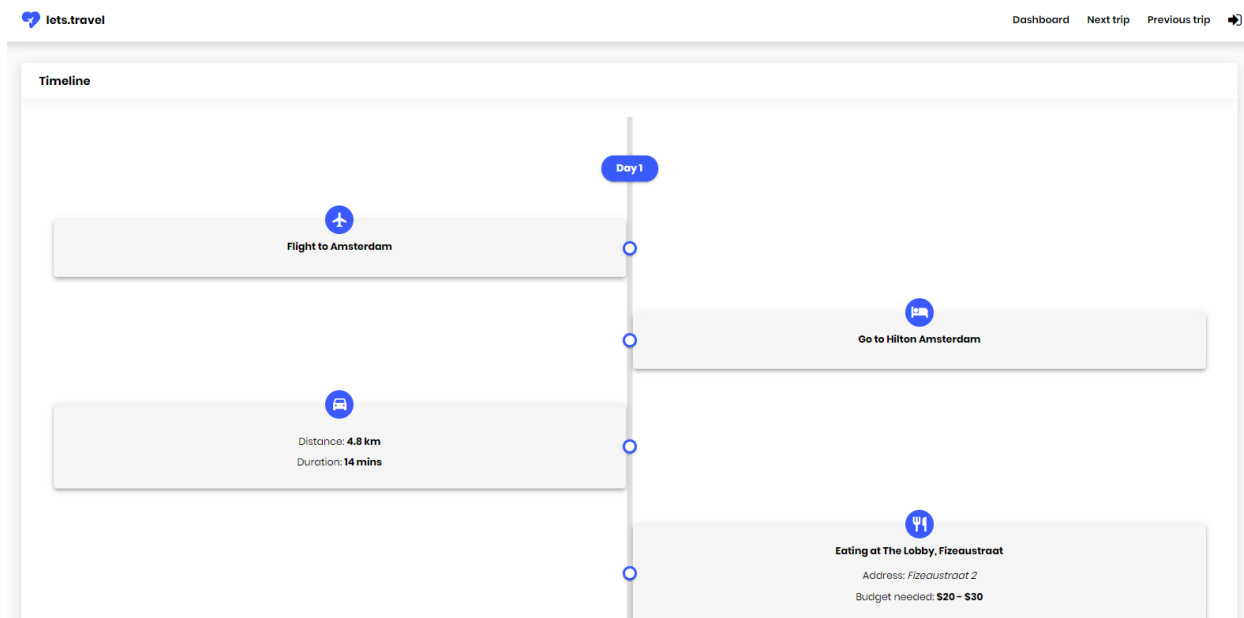


Figura 31: Timeline cu programul final

Fiecare activitate este reprezentată sub forma unui card cu iconiță specifică tipului de activitate iar zilele sunt evidențiate. De asemenea, distanța dintre două locații este afișată tot sub formă de card, cu iconiță pentru specificarea modului de deplasare și detalii despre distanță și durata acesteia. În plus, pentru localurile din secțiunea *food*, este afișată o estimare de buget pentru o masă per persoană. Acest buget este în funcție de datele din API-ul de la Foursquare care oferă pentru locații o categorie de preț de la 1 la 4, cu un interval exemplificat în Figura 32.

```

54     getPrice(tier: number) {
55         switch(tier) {
56             case 1:
57                 return '< $10';
58             case 2:
59                 return '$10 - $20';
60             case 3:
61                 return '$20 - $30';
62             case 4:
63                 return '> $30';
64             default:
65                 return;
66         }
67     }

```

Figura: 32 Categoriile de preț în funcție de datele Foursquare

Pentru aducerea datelor specifice călătoriei, la inițializarea componentei de timeline, se face un request la baza de date de la Firebase pentru toate călătoriile utilizatorului curent. După cum se poate observa și în Figura 33, după ce se primesc toate planurile pentru călătorii se caută călătoria cu id-ul corespunzător primit ca parametru în url-ul aplicației.

```

37     ngOnInit() {
38         let copyThis = this;
39         let tripKey = this.route.snapshot.paramMap.get("plan");
40         firebase.auth().onAuthStateChanged((user) => {
41             copyThis.firebaseService.getPlans(user.uid).subscribe(plans => {
42                 copyThis.plans = plans;
43                 copyThis.plans.forEach(plan => {
44                     if (plan.tripId === tripKey) {
45                         copyThis.events = Object(plan).events;
46                         this.getDistances(copyThis.events);
47                         this.showTimeline = true;
48                     }
49                 });
50             });
51         });
52     }

```

Figura 33: Metodă în care se aduc date despre planul specific călătoriei

3.3.6 Interfața aplicației

Interfața aplicației fiind un aspect important pentru utilizator, de a fi cât mai simplu și ușor de folosit dar în același timp și cu concepte moderne de design. Pentru a îndeplini aceste aspecte și aplicația să fie cât mai consistentă la nivel de interfață, am folosit componentele de la Angular Material.

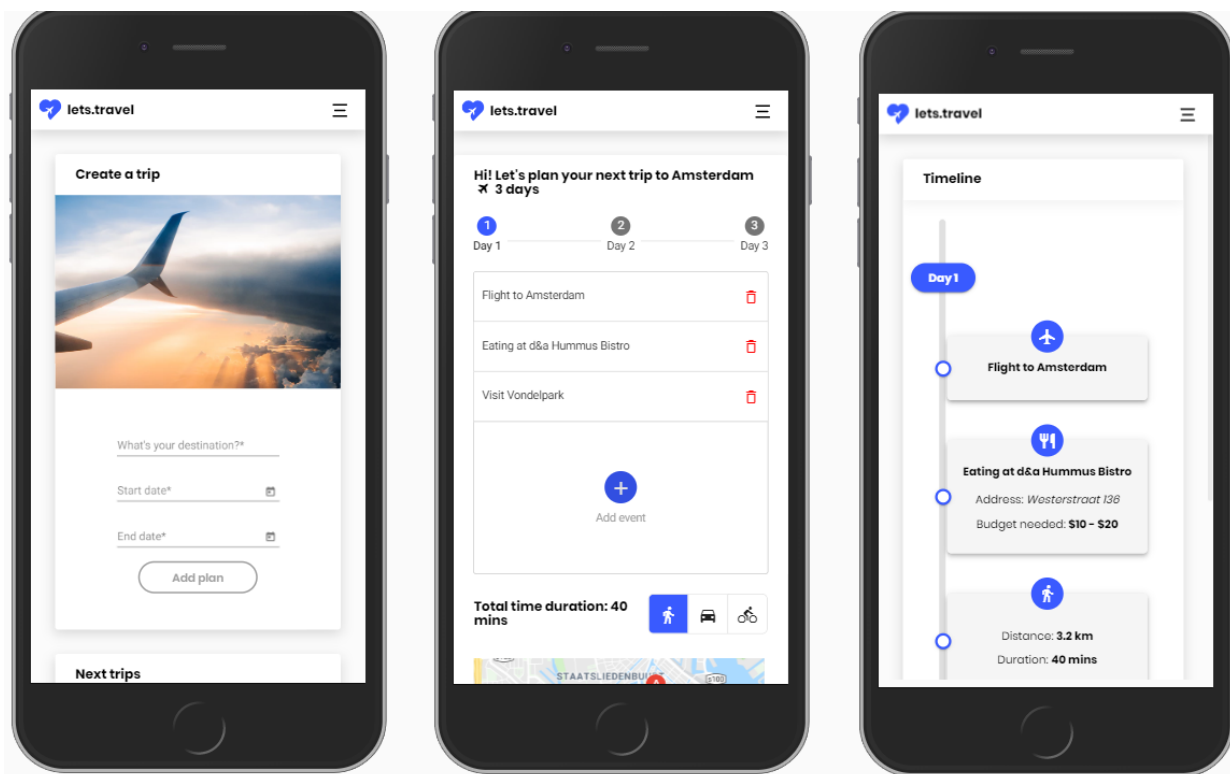


Figura 34: Accesibilitatea pe telefoane

Un alt lucru important este accesibilitatea aplicației și pe mobil. Se observă în Figura 34 de mai sus că aplicația este *responsive* și atractivă pe mobile.

CONCLUZIILE LUCRĂRII

1. Concluzii generale

Scopul aplicației de a aduce împreună un mediu de planificare a călătoriilor și de asemenea sa ofere recomandări de activități a fost îndeplinit, utilizând tehnologii moderne și API-uri potrivite.

Aplicația prezentată în lucrare oferă un mediu în care poți să îți planifici excursiile rapid, vine în ajutorul tău cu activități și recomandări. În plus, utilizatorul își poate vizualiza traseele și știe cât timp va petrece pe drum. La finalul planificării, aplicația oferă o privire de ansamblu asupra întregii călătorii dar pe care îl poate vizualiza oricând de pe pagina de dashboard. De asemenea, aplicația este accesibilă atât pe desktop cât și pe mobil.

2. Direcții de viitor

În viitor ca funcționalitate ar putea fi adăugate și recomandări de zboruri, hoteluri. Pe lângă acestea, să ai posibilitatea să faci rezervări, să poți cumpăra bilete direct din platformă.

O altă idee, ar fi ca aplicația să genereze idei de planuri zilnice și să recomande un buget total pentru toată călătoria în funcție și de câți călători participă. În plus, ar fi o idee ca planurile fiecăruia să fie salvate și să vină ca recomandare altei persoane care călătorește în același loc.

În final să acumuleze toate funcționalitățile de călătorii existente acum pe internet, ca utilizatorul să nu mai fie nevoit să folosească 5 aplicații în loc de una pentru a își organiza propria călătorie.

BIBLIOGRAFIE

1. Pagina Angular <https://angular.io/>
2. Pagina Angular Material <https://material.angular.io/>
3. Pagina Material Design <https://material.io/design/>
4. Pagina Firebase <https://firebase.google.com/>
5. Firebase Realtime database <https://firebase.google.com/products/realtime-database>
6. Documentație Google Maps <https://developers.google.com/maps/documentation/>
7. Directions API <https://developers.google.com/maps/documentation/directions/start>
8. Places autocomplete API <https://developers.google.com/places/web-service/autocomplete>
9. Places Details API <https://developers.google.com/places/web-service/details>
10. Foursquare API <https://developer.foursquare.com/places-api>
11. Venue recommendations <https://developer.foursquare.com/docs/api/venues/explore>
12. Foursquare API - place details <https://developer.foursquare.com/docs/api/venues/details>
13. Angular Google Maps Direction <https://www.npmjs.com/package/agm-direction>
14. Angular Google Maps <https://www.npmjs.com/package/@agm/core>
15. Foursquare Places API <https://developer.foursquare.com/docs/api>
16. Angular mgl timeline package <https://www.npmjs.com/package/angular-mgl-timeline>
17. Material icons <https://material.io/tools/icons/?style=baseline>
18. Google Poppins Fonts <https://fonts.google.com/specimen/Poppins>

Link-uri poze

- [1] <https://forkingtasty.com/wp-content/uploads/2016/09/Gogle-Trips-App.jpg>
- [2] <https://www.google.com/travel/>
- [3] <https://cdn-image.travelandleisure.com/sites/default/files/1488385348/tripit-airport-timing-app-TRIPIT0317.jpg>
- [4] <https://www.inspirock.com/>
- [5] <https://angular.io/guide/architecture>
- [6] <https://www.ficode.co.uk/wp-content/uploads/2017/08/firebaseblog.png>
- [7] <https://foursquare.com/developers/explore#req=venues%2Fexplore%3Fnear%3DIasi%26section%3Dfood>
- [8] <https://foursquare.com/developers/explore#req=venues%2F5380c350498e59fb622f245e%3F>

ANEXE

1. Angular

Aplicația este dezvoltată la nivel de client cu ajutorul framework-ului Angular versiunea 7.2.0. Angular este un *framework* pentru a construi aplicații la nivel de client cu *HTML* și *Typescript*. În Fig. 8 de mai jos sunt reprezentate relațiile între conceptele principale din acest *framework*. De asemenea, este evidențiat faptul că un *view* este definit printr-o componentă și un *template* iar *injector*-ul furnizează servicii unei componente, cum ar fi serviciul de rutare în care ai posibilitatea de a îți defini navigarea între pagini.

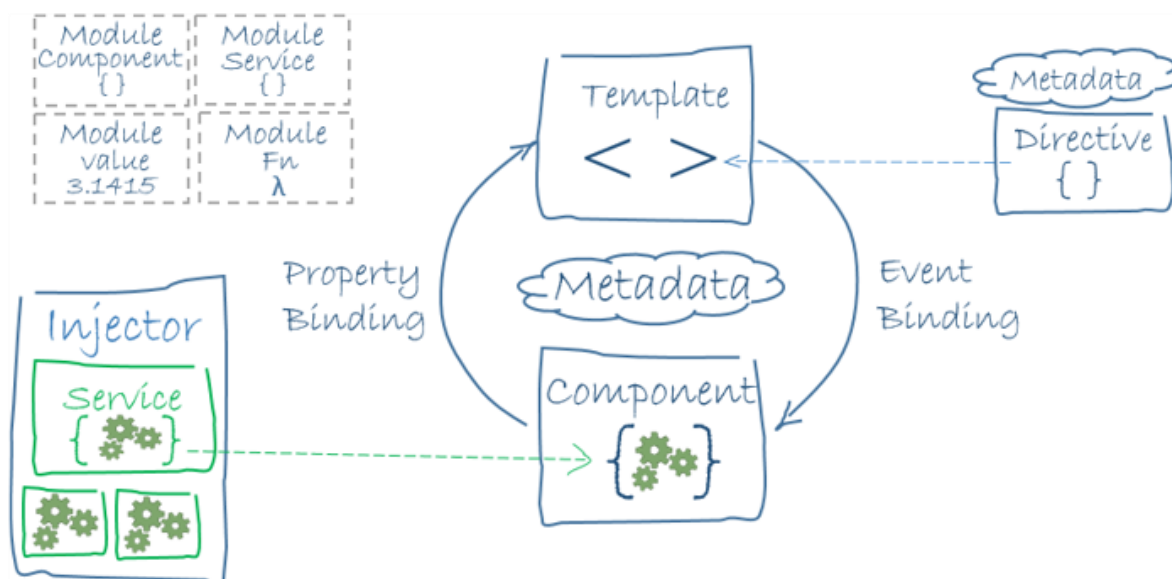


Figura 35: Relațiile între conceptele principale din Angular [5]

2. Angular Material

Pentru a dezvolta mai ușor aplicația și a avea consistența la nivel de interfață, am folosit Angular Material, ce este o librărie pentru Angular ce conține componente deja dezvoltate având ca design Google Material UI. Angular Material ajută ca aplicațiile să fie atractive, consistente și fluide. De

asemenea, componentele sunt bine testate și construite de echipa de la Angular pentru a se integra perfect cu Angular framework.

3. Firebase

Google Firebase ce este o platforma cu ajutorul căreia poți dezvolta rapid aplicații. Din funcționalitățile oferite am folosit autentificarea (Firebase Authentication), baza de date (Realtime Database) și mediul de hosting.

4. Firebase Authentication

Autentificarea de la Firebase oferă un sistem de autentificare securizat și simplu, îmbunătățind experiența de logare a utilizatorului. Acesta furnizează servicii backend, SDK-uri ușor de folosit și suportă autentificarea utilizând email și parolă sau prin furnizori ca Google, Facebook sau Twitter etc.

5. Firebase Realtime Database

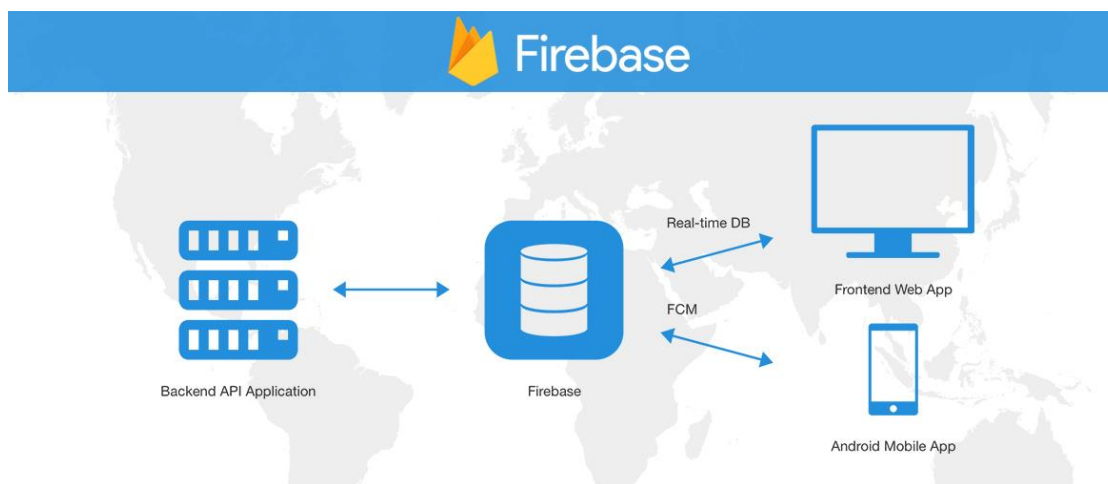


Figura 36: Firebase Realtime Database [6]

Stocarea și sincronizarea datelor se realizează cu baza de date NoSQL de pe cloud. Datele sunt sincronizate pentru toți clienții în timp real și rămân disponibile și când aplicația este offline. Datele sunt stocate în format JSON și datele se actualizează automat și instant. Pașii pentru folosirea bazei de date de la Firebase:

1. Integrarea SDK-ului pentru Firebase Realtime Database

2. Creează o referință pentru baza de date, cum ar fi "users/user:1234/phone_number"
3. Setează date și așteaptă modificări

6. Foursquare Places API

Foursquare Places API oferă detalii precise despre localuri/atracții dintr-un loc anume. De asemenea, poți obține recomandări de locații în funcție de secțiune, zona etc. În Figura 36 putem observa cum arată un răspuns pentru recomandări în zona Iași, categoria mâncare.

```
"type": "Recommended Places",
"name": "recommended",
- "items": [
  - "0": {
    + "reasons": { ... },
    - "venue": {
      "id": "5380c350498e59fb622f245e",
      "name": "VIVO Fusion Food Bar",
      - "contact": {
        "phone": "0332417727",
        "formattedPhone": "0332 417 727",
        "facebook": "229934233809483",
        "facebookUsername": "vivofood",
        "facebookName": "Vivo - Fusion Food Bar Iasi"
      },
    - "location": {
      "address": "Str. Zimbrului nr. 5",
      "lat": 47.15580561116792,
      "lng": 27.591429536864222,
      - "labeledLatLngs": [
        - "0": {
          "label": "display",
          "lat": 47.15580561116792,
          "lng": 27.591429536864222
        }
      ],
      "postalCode": "700047",
      "cc": "RO",
      "city": "Iași",
      "state": "Iași",
      "country": "România".
    }
  }
]
```

Figura 36: Răspuns la o solicitare pentru recomandări în zona Iași, secțiunea de mâncare [7]

Din cadrul acestui API, am mai folosit solicitările pentru căutarea unui local după un text dat, obținerea unei poze pentru localuri și obținerea detaliilor despre preț pentru localurile cu

mâncare. Detaliile despre preț și categoria localul se poate observa cum vine ca răspuns în Figura 37 de mai jos.

```
- "categories": [  
  - {  
    "id": "4bf58dd8d48988d16c941735",  
    "name": "Burger Joint",  
    "pluralName": "Burger Joints",  
    "shortName": "Burgers",  
    - "icon": {  
      "prefix": "https://ss3.4sqi.net/img/categories_v2/food/burger_",  
      "suffix": ".png"  
    },  
    "primary": true  
  },  
],  
  "verified": false,  
+ "stats": { ... },  
- "price": {  
  "tier": 1,  
  "message": "Cheap",  
  "currency": "$"  
},
```

Figura 37: Răspuns la o solicitare pentru detalii despre un anumit local

7. Google Maps API

Am folosit Google Maps API, pentru vizualizarea locațiilor pe hartă iar pentru integrarea în proiect am folosit componenta *agm-map* de la Angular Google Maps. Iar pentru trasee în funcție de modul ales de deplasare, am folosit Directions API pe care l-am integrat cu ajutorul directivei *agm-direction* pentru *agm/core*. Cu ajutorul acesteia am creat trasee și detalii pentru acestea, cum ar fi timpul de deplasare și distanța.

```
licenta-letstravel> npm install @agm/core --save
```

Figura 38: Comanda pentru instalarea pachetului pentru angular google maps

Tot din serviciile oferite de Google, am folosit și Google Places API pentru funcționalitatea de căutare cu autocomplete oferită de ei și pentru integrarea acestuia în proiect am folosit librăria *google-maps-autocomplete*.

8. Google Directions API

Directions API este un serviciu ce calculează direcțiile dintre locații. Cu ajutorul acestui serviciu poți căuta diferite modalități de transport cum ar fi mers pe jos, cu mașina, cu bicicleta etc. Realizează trasee specificând zona din care pleacă, destinația și eventual locații intermediare. API-ul returnează cel mai eficient traseu dar în plus și detalii despre distanță și timpul estimativ.

9. Google Places API

Places API este un serviciu ce returnează informații despre locuri iar dintre serviciile oferite în proiect am folosit Place Photos și Place Autocomplete.

Place Photos furnizează accesul la milioane de poze corespunzătoare locației pe care o oferi ca parametru.

Place Autocomplete este un API ce automat completează numele și adresa locului pe care utilizatorul îl scrie.