

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Explorello, companionul tău în escapadele
urbane**

propusă de

George Giosanu

Sesiunea: *iulie, 2019*

Coordonator științific

Prof. Colab. Florin Olariu

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Explorerello, companionul tău în escapadele urbane

George Giosanu

Sesiunea: *iulie, 2019*

Coordonator științific

Prof. Colab. Florin Olariu

Avizat,
Îndrumător Lucrare de Licență

Titlul, Numele și prenumele _____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul(a) domiciliul în
..... născut(ă) la data de,
identificat prin CNP, absolvent(a) al(a) Universității „Alexandru Ioan Cuza”
din Iași, Facultatea de specializarea, promoția
....., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul
art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare
la plagiat, că lucrarea de licență cu titlul: _____

_____ elaborată sub îndrumarea dl. / d-na _____
_____, pe care urmează să o susțină în fața comisiei este
originală, îmi aparține și îmi asum conținutul său în întregime. De asemenea, declar că sunt de acord ca
lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității,
consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop. Am luat la
cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării
falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și
în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul
cercetării pe care am întreprins-o.

Data azi,

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Explorello, companionul tău în escapadele urbane”, 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
Motivație	6
Ce înseamnă Explorello?	7
Contribuții	9
Capitolul 1 – Descrierea problemei	11
Capitolul 2 – Abordări anterioare	12
Capitolul 3 – Descrierea soluției	15
3.1 Tehnologii folosite	15
3.2 Servicii integrate	17
3.3 Stocarea datelor	18
3.4 Fluxul aplicației	21
3.4.1. Navigarea	21
3.4.2. Autentificarea	22
3.4.3. Interacțiunea cu utilizatorul	24
3.4.4. Cursul aplicației	25
Concluziile lucrării	32
Concluzii	32
Direcții viitoare	32
Bibliografie	33

Introducere

Motivație

Călătoriile sunt unul dintre hobby-urile preferate ale secolului nostru, rezultat al dorinței omului de a se lăsa fascinat de mediul înconjurător, fie el urban sau natural. În viteza cu care noi ne trăim viața, căutăm să ne bucurăm de cât mai multe dintre peisajele și locurile de pe întregul pământ. Unul dintre țelurile noastre când vizităm un loc este să marcăm cât mai multe dintre aceste obiective turistice și culturale.

Acest fapt a dus la dorința multora de a rezolva problemele de eficiență în ce privește călătoritul. Multe aplicații au adus considerabile îmbunătățiri în ce privește facilitarea transportului, a cazării și nu în cele din urmă, a domeniului culinar.

Aplicații precum *Visit a City*, *Lonely Planet* și *Kayak* au confruntat problemele de mai sus, reușind să centralizeze date și servicii pentru utilizatori astfel încât au adus îmbunătățiri. A fost simplificat procesul de rezervare a unui loc de cazare, a unui bilet de avion sau a închirierii unei mașini.

Dintre aplicațiile care aduc un răspuns la întrebarea “Ce obiective turistice aș putea atinge în acest **city break**?”. *Visit a City* reușește să facă acest lucru într-o măsură mulțumitoare, concentrând informații relevante despre destinații de orice fel sub formă de itinerarii.

Un aspect pe care marile aplicații de traveling de pe piață nu reușesc să-l abordeze este o eficientizare mai severă în ce privește marcarea a câtor mai multe puncte turistice într-un timp cât mai scurt. Acest aspect îl abordează *Explorello*, companionul tău în escapadele urbane.

Ce înseamnă Exploreello?

Exploreello este o soluție a problemei pe care o întâmpinam atunci când călătoream perioade scurte de timp în orașe mari ale lumii, în care o proastă organizare a timpului, energiei și a banilor puteau ruina experiența, sau stârni ulterior regrete și o dorință costisitoare de a reveni acolo cât mai curând.



Companionul Exploreello, disponibil doar pe Android în fază inițială, este un spațiu online în care o comunitate de nomazi și persoane avide după experiențe de ordin cultural se întâlnesc și creează împreună o bibliotecă de *route* menite să simplifice călătoritul în perioade scurte de timp.

Utilizatorii vor putea crea rute în toate orașele lumii, vizitatori și localnici deopotrivă, menite să satisfacă obiectivele pe care călătorii și l-au propus pentru excursia respectivă. Rutele pot atinge puncte relevante ce privesc domenii specifice, cum ar fi cele culinare, culturale, artistice și multe altele.

Totodată, Exploreello este și un loc în care poți ține evidența călătoriilor tale, fiecare utilizator având posibilitatea de a-și crea un profil privat în care să aibă la dispoziție un istoric al rutelor parcurse și evenimentelor la care a participat pe parcursul excursiei. Mai mult decât toate cele mai de sus, Exploreello va fi o cale de a construi prietenii și a transforma excursiile de orice fel în oportunități de a-ți dezvolta networking-ul global ce va transforma călătoriile în experiențe ce îți vor schimba viața.

Tehnologia principală folosită în implementarea aplicației este React Native, un framework pus la dispoziție de către echipa *Facebook*. Motivația alegerii acestei tehnologii este relativ firească, aceasta fiind elasticitatea cross-platform pe care o pune la dispoziție. Deși nu am facilitat în totalitate de ea, concentrându-mă în fază inițială doar pe platforma Android, React Native lasă totuși o rapidă și convenabilă opțiune de a implementa și modulul pentru iOS cu foarte mici eforturi și costuri. Un alt motiv este limbajul de programare JavaScript utilizat pentru construirea aplicațiilor în React, ce face mai abordabilă participarea la implementare pentru dezvoltatorii de aplicații Web.

Contribuții

În realizarea acestui proiect a trebuit să încep prin a studia piața în ce privește domeniul adresat. În urma studierii pieței, observării a punctelor slabe dar și al celor forte ale aplicațiilor de top, am concretizat originalitatea soluției, retușat prototipul ideii principale și nu în cele din urmă, am învățat ce e de evitat și ce e inevitabil pentru aplicație pentru a avea succes.

După analiza ideilor principale, am schițat o arhitectură relativ minimalistă a datelor pe care aplicația urma să le stocheze și prelucreze, am analizat ce servicii sunt puse la dispoziție pentru a facilita anumite funcționalități, dar și alegerea tehnologiei principale de implementare.

Metoda de autentificare, ce a presupus integrarea a trei furnizori de servicii de logare și înregistrare diferiți, *Facebook*, *Google*, și *Firebase*, comună în aplicațiile mobile, dar necesară și extrem de convenabilă pentru orice utilizator.

În ce privește natura vizuală a proiectului, am ales o paletă de culori redusă, o interfață simplă, ce reușește să scoată în evidență funcționalitățile cheie ale aplicației, iar familia de fonturi *Montserrat* contribuie de asemenea la o experiență plăcută pentru utilizator, fiind lizibil și elegant. Logo-ul a fost creat de mine într-un tool online de design și desenat (*logomakr*), cu un stil menit să inducă un sentiment jucăuș și subtil informal.

Funcționalitățile pe care aplicația le pune la dispoziție, deși în aparență comune în cadrul aplicațiilor de traveling, sunt livrate într-un cadru diferit și dedicat comunității nomazilor și a amatorilor de city break-uri. O funcționalitate cheie este crearea și accesarea de rute auto-generate prin intermediul serviciului de hărți Google, ce permite turiștilor să elimine timpul pierdut cu sortarea punctelor pe care vor să le viziteze.

Un sistem de recenzii pentru rutele create există de asemenea, lucru ce influențează nu doar popularitatea și calitatea celor mai reușite dintre ele, ci pot să ajute la îmbunătățirea celor care nu au un succes răsunător.

Nu în cele din urmă, integrarea serviciului pus la dispoziție de Ticketmaster, ce permite utilizatorului să vadă în cadrul aplicației evenimente ce au loc pe perioada sejurului turistului.

Capitolul 1

Descrierea problemei

Înainte de a descrie problema, voi începe prin a prezenta contextul specific în care problema descrisă merită într-adevăr atenția cuvenită. Contextul este cel al călătoriei, mai exact, cel al călătoriilor scurte, în special city break-urile.

În multe din escapadele mele urbane m-am regăsit în situația de a nu ști cum să-mi gestionez resursele: timpul, banii și energia. Multitudinea aplicațiilor pe care le foloseam îmi induceau mai degrabă dezorientare și confuzie decât să mă ajute să pun la cale o excursie reușită. Strategia de a te pregăti anterior deplasării este demodată și elimină factorul cheie: spontaneitatea acesteia.

Nenumăratele itinerarii puse la dispoziție de majoritatea aplicațiilor devin copleșitoare și plastice, lipsindu-le un punct de vedere autentic în ce le privește alcătuirea. Acest punct de vedere fiind al unei comunități de amatori de aventuri urbane și nu numai. Un alt factor dezamăgitor este incertitudinea alegerii unor puncte de vizită, din cauza motivării din spatele promovării acelui obiectiv turistic.

Așadar implicarea utilizatorilor obișnuiți într-un cadru controlat lipsea, iar rezidenții locurilor vizitate nu aveau posibilitatea de a ajuta la construirea unor itinerarii mai personalizate în care să-și pună amprenta experiența unui localnic sau cea a unui călător experimentat.

Pe de altă parte, pentru excursiile în care aventurierul dorește să maximizeze spontaneitatea, (caz personal) există acea curiozitate și dorință de a grava ruta parcursă, locurile vizitate și evenimentele la care ai luat parte, funcționalități prezente în aplicație.

În concluzie, problema, deși specifică, a meritat confruntată, deoarece soluția oferă un cadru prietenos, personalizat și eficient de gestionare a călătoriilor scurte și cu rezultat maxim.

Capitolul 2

Abordări anterioare

Din mulțimea aplicațiilor ce se adresează acestui subiect, două dintre ele reușesc să aducă considerabile îmbunătățiri în ce privește experiența unei excursii contra-timp, însă nu întocmai într-un spirit de unitate, ci mai degrabă într-unul comercial și ușor concentrat pe alte particularități ale unei călătorii.

1. Lonely Planet



Fig. 1. Listă ghiduri (stânga) și un ghid detaliat (dreapta) în cadrul aplicației Lonely Planet

Această aplicație relativ populară în rândul nomazilor, a fost fondată de un cuplu englez în anul 1972, Maureen respectiv Tony Wheeler, în urma unor repetate călătorii prin Asia și Europa.

Inițial pornind cu producerea de ghiduri turistice în format fizic, Lonely Planet a ajuns în cele din urmă una dintre cele mai folosite aplicații de călătorit, punând la dispoziție o cantitate considerabilă de date pentru locații de pe întreaga planetă acestea abordând teme precum cultura, civilizația și atracții exotice (Fig. 1).

Deși Lonely Planet este un grozav partener de călătorii, nu are nici o funcționalitate ce poate rezolva problema ridicată mai sus. Deoarece crearea ghidurilor se face în cadrul companiei, utilizatorii nu au un cuvânt de spus. De asemenea popularitatea aplicației poate avea un efect profitabil pentru anumite puncte de vizitare comerciale ce apar în ghiduri, întrucât în istoria aplicației acest lucru a avut loc, stârnind efectul “Banana Pancake Trail”. Acest efect poate duce la implicarea business-urilor în alcătuirea unui ghid, fapt ce reduce nuanța și autenticitatea unui itinerariu.

2. Visit a City

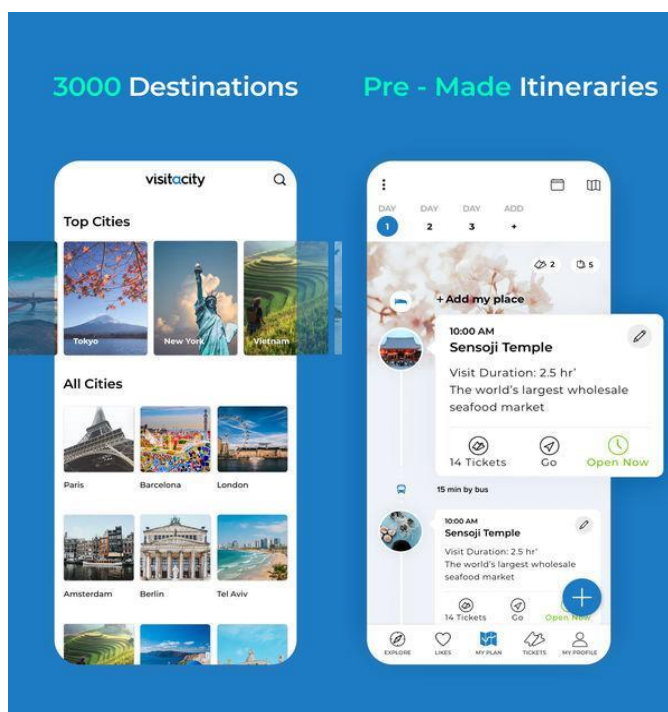


Fig. 2. Listă orașe (stânga) și un itinerariu detaliat (dreapta) în cadrul aplicației Visit a City

Visit a City, spre deosebire de Lonely Planet reușește să se adreseze problemei lipsei de varietăți în ce privește obiectivele unei călătorii. Această aplicație pune la dispoziție numeroase itinerarii, cu diferite tematici și expune un filtru pentru numărul de zile acordate excursiei (Fig. 2).

Deși rezolvă dilema tururilor și rutelor turistice oferind detaliate informații în ce privește ordinea vizitării locurilor, durata și alternativa transportului în public, este expusă aceluiași dezavantaj menționat mai devreme, și anume influența business-urilor în experiența celui ce călătorește.

De asemenea nu elimină dilema subiectivismului, întrucât toate itinerariile puse la dispoziție de către ei sunt la fel construite în cadrul companiei, influența userilor fiind limitată doar la review-uri pentru tururile plătite.

Capitolul 3

Descrierea soluției

Soluția problemei, luând în considerare cadrul în care a apărut, nu ar fi putut fi decât sub forma unei aplicații mobile, cu o multitudine de funcționalități menite nu doar să trateze problema, ci și să faciliteze alte aspecte ale călătoritului.

3.1. Tehnologii folosite

Principala tehnologie, și anume instrumentul de realizarea aplicației mobile Android, este React Native, un framework open-source creat de către *Facebook* ce permite folosirea atât a ReactJS-ului cât și a platformelor native de dezvoltare (Java, Kotlin sau Swift pentru iOS).

Sunt mai multe aspecte pentru care am făcut această alegere, unul dintre ele fiind familiaritatea cu JavaScript, opțiunea de a dezvolta simultan o aplicație Android și iOS fără a fi nevoie de a învăța două tehnologii noi și de asemenea avantajele ReactJS-ului și varietatea pachetelor dezvoltate de comunitatea dezvoltatorilor de aplicații în React respectiv React Native.

Modul în care funcționează React-Native este foarte simplu. Acesta are în componența sa trei thread-uri, unul pentru JavaScript, unul pentru codul nativ (Swift iOS, Java/Kotlin Android) și un thread “bridge” responsabil cu serializarea și deserializarea componentelor prin care celelalte două threaduri comunică.

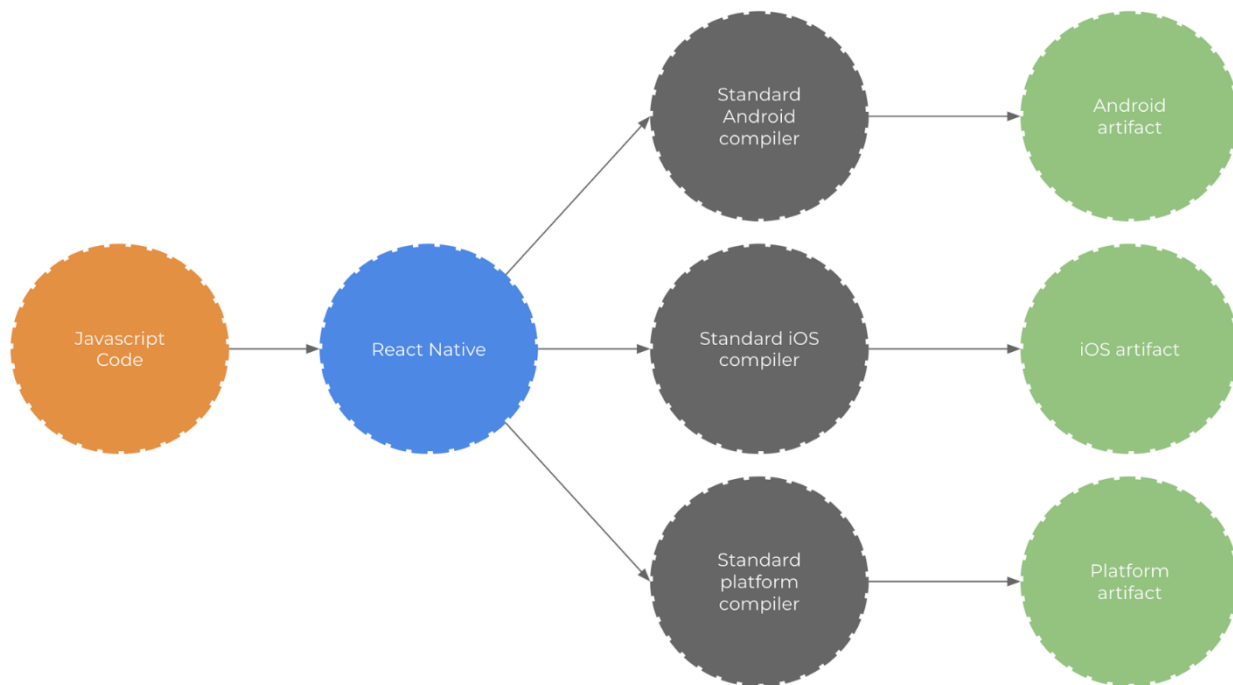


Fig 3. Diagramă simplificată ce descrie modul de funcționare a framework-ului React Native

În componența React-ului, intră JSX-ul (**JavaScrip eXtension**), o extensie menită să permită folosirea unui limbaj de tip Mark-up în cadrul scripturilor JavaScript și bine înțeles pentru partea de stilizare CSS-ul este tehnologia ce îndeplinește acest rol.

În cadrul React-ului am folosit importante pachete de gestionare a datelor și acțiunilor la nivel de aplicație a clientului precum:

- *react-redux*, pentru state-ul aplicației (separând acțiunile și datele, în *actions* respectiv *reducers*)
- *react-persist*, pentru menținerea datelor statice în cache (Ex: pentru a reține dacă userul s-a autentificat în momentul redeschiderii aplicației)
- *react-thunk*, middleware ce supraveghează expedierea acțiunilor din cadrul aplicației și atașând state-ul aplicației fiecărei acțiuni.

3.2. Servicii integrate

- Google Maps API.

Prin intermediul pachetelor *react-native-google-place-autocomplete*, *react-native-maps-directions*, *react-native-geolocation-service*, și *react-native-maps*, am integrat o mare parte din întreaga suită pe care Google o pune la dispoziție în ce privește hărțile, localizarea și calcularea rutelor.

- Firebase

Pentru stocarea datelor, și procesul de autentificare/înregistrare am ales serviciile puse la dispoziție de *Firebase* întrucât permite o gestionare de ansamblu a aplicației, indiferent de natura acesteia, Web sau Mobile. Pachetul *react-native-firebase* este cel ce gestionează această integrare și comunicarea cu API-ul.

- Ticketmaster

API-ul pus la dispoziție de către Ticketmaster permite funcționalitatea legată de informațiile despre evenimente, precum concerte, festivaluri culinare sau evenimente sportive.

- Facebook

Integrarea serviciilor Facebook, doar pentru autentificare în stadiul curent al aplicației, prin intermediul pachetului *react-native-fbsdk*. Pachetul permite utilizarea întregului API pus la dispoziție de către *Facebook*.

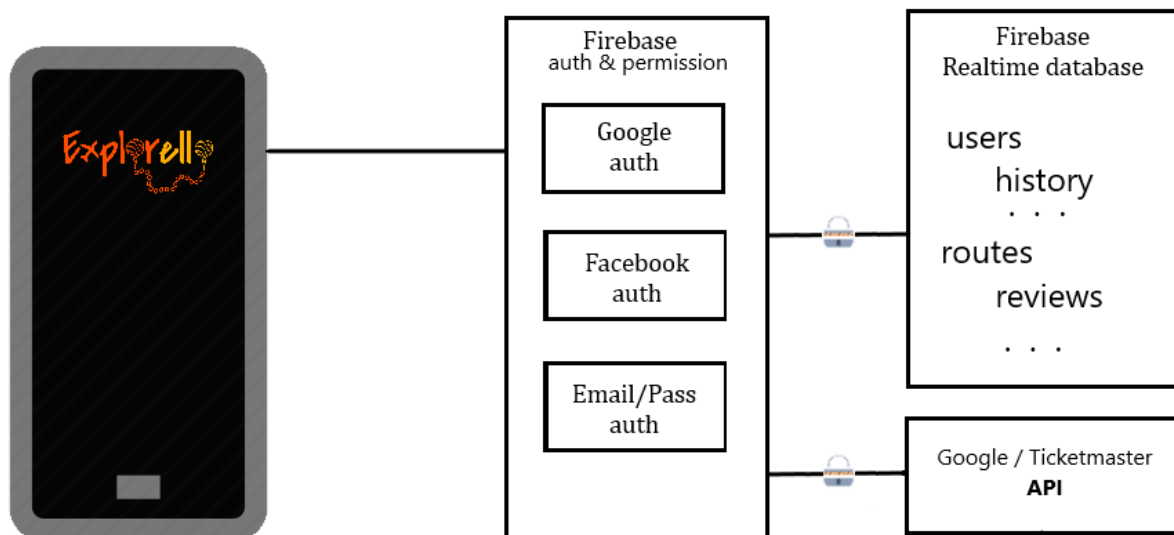


Fig 4. Diagramă schematică a aplicației în raport cu serviciile integrate

3.3. Stocarea datelor

- *Baza de date*

Modul de stocare a datelor în *Realtime Database* pus la dispoziție de către *Firebase*, este unul nerelațional, această bază de date fiind alcătuită dintr-un JSON, operațiile CRUD fiind furnizate de către API-ul aferent.

Evidența userilor nu face parte din *Realtime Database*, ci ține de modulul de autentificare din cadrul serviciilor *Firebase*. Userii sunt reținuți într-o tabelă separată, datele fiind relativ sumare, suficiente însă pentru identificarea user-ului și analiza activităților din cadrul autentificării și înregistrării (Fig. 5).

<div> Search by email address, phone number, or user UID <div> Add user </div> </div>				
Identifier	Providers	Created	Signed In	User UID ↑
ilovecitybreaks@gmail.com		Jun 25, 2019		7aBhAcYQ11ZAwjcdRX8K56JsVv...
test@test.test		Jun 20, 2019	Jun 20, 2019	2f0EhoTZkLRnVo3YPL2XGHmB0rx2

Fig 5. Un fragment al tabelii de evidență a userilor în *Firebase Authentication*

Echivalentul unui tabel al unei baze de date relaționale pentru Realtime Database sunt *cheile* de la primul nivel al JSON-ului ce reprezintă numele tabelului. În Figura 6 putem observa un exemplu, cheia “routes” fiind la nivelul 0, aceasta fiind identificatorul tabelului.

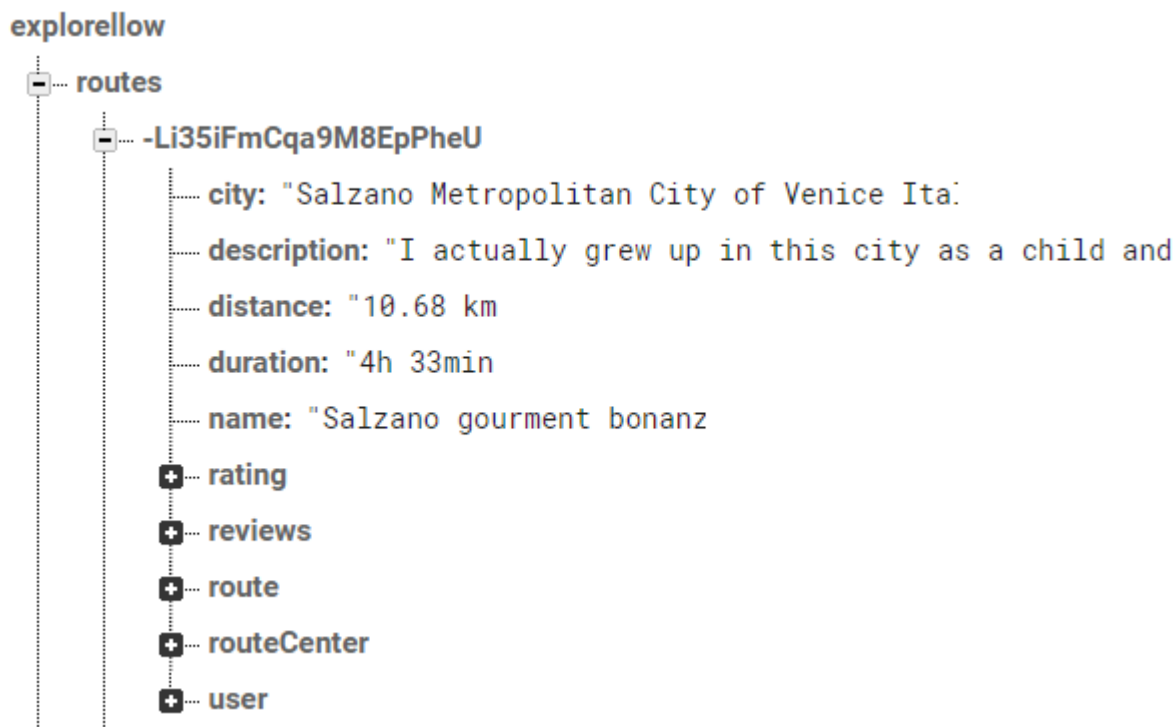


Fig 6. Tabela routes din Firebase Realtime Database, cu un entry

- *Datele la nivel de client*

Stocarea și gestionarea datelor la nivelul aplicației client este realizată prin intermediul pachetelor dedicate acestui aspect din cadrul spațiului de dezvoltare oferit de React. Redux integrat cu React prin dependențele *react-redux* și *redux*, se ocupă de gestionarea “state”-urilor aplicației, acesta fiind alcătuit din răspunsurile primite de la server-ul bazei de date sau API-uri, date din cache sau date create local și nestocate încă în baza de date.

Redux impune un set de principii ce duc la predictibilitate în ceea ce privește comportamentul *mutației* unei stări, când și cum poate fi ea alterată. Aceste principii descriu un mod de controlat de modificare a stării unei întregi aplicații. Modificările au loc doar prin emiterea unor *acțiuni*, ce descriu schimbarea stării, iar funcțiile pure numite *reductori* se ocupă cu marcarea schimbării descrise de acțiune și concretizând astfel o nouă stare.

Un alt rol important îl joacă *redux-persist*, acesta fiind responsabil cu persistența datelor la nivel de client, precum stocarea datelor în cache, în condițiile pierderii conexiunii la internet, sau reținerea anumitor acțiuni ale utilizatorului pentru a evita agasarea utilizatorului, precum logarea în aplicație.

Structura pe care am ales-o pentru gestionarea stării aplicației este relativ modulară, fiind alcătuită din mai mulți reductori, fiecare având ca responsabilitate sub-stări autentificarea sau cele administrează rutele și adăugarea lor (Fig. 7).

```
let store = createStore(combineReducers({
  login: persistedLogin,
  routes: persistedRoutes,
  addRoute: persistedAddRoute,
  events: persistedEvents,
  user: persitedUser
}), applyMiddleware(thunk));
```

Fig 7. Store-ul aplicației, alcătuit din mai mulți reductori.

3.4. Fluxul aplicației

3.4.1. Navigarea

Structurarea ordinii ecranelor într-o aplicație mobile este ușor asemănătoare cu cea din aplicațiile web, un ecran fiind echivalentul unei pagini web. Ce diferă însă este ordinea și modul de îmbinare a componentelor astfel încât fluxul aplicației să fie unul cursiv și intuitiv. De asemenea o modularitate în ce privește organizarea componentelor principale are ca rezultat și o mai ușoară structurare a stărilor și reductoarelor. În schema de mai jos putem observa stratificarea navigatoarelor și ecranelor (Fig. 11).

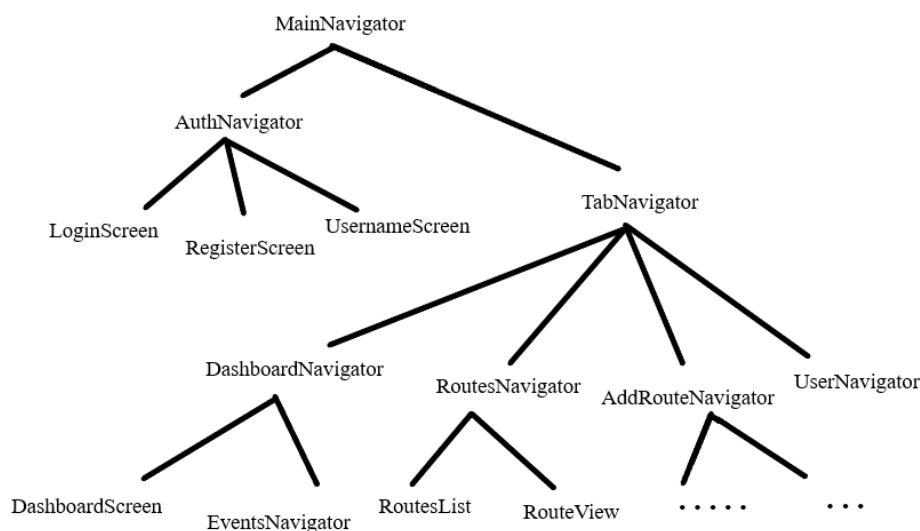


Fig 11. Structura ecranelor în vederea navigării prin aplicație

Navigarea între componente se face prin intermediul librăriei *react-navigation*, prin folosirea funcțiilor precum *createSwitchNavigator()* și *createBottomTabNavigator()* ce îmbină componente React și le atașează proprietăți ce le permite navigarea sau trimiterea de parametri de la un ecran la altul. Funcția de navigare atașată este *navigate()*, aceasta primind ca parametru numele ecranului și respectiv parametrul opțional în cazul în care componenta sursă are date de transmite celei destinație.

3.4.2. Autentificarea

Autentificarea este o caracteristică inevitabilă și obligatorie într-o aplicație ce are ca obiectiv potențarea utilizatorului. Însă deși prezentă în majoritatea programelor, aceasta poate fi sau nu descurajantă, în funcție de complexitatea unui formular de înregistrare sau logare. De asemenea aceasta, în cele mai multe cazuri, este primul contact al utilizatorului cu aplicația.

Un mod de a încuraja un potențial utilizator este să facilitezi metoda de înregistrare. Acest lucru se poate face în mai multe moduri, printre care punerea la dispoziție a autentificării prin intermediul altor servicii mai populare, precum *Facebook* sau *Google* (Fig. 8), sau prin reducerea numărului de câmpuri dintr-un formular de înscriere (Fig. 9).

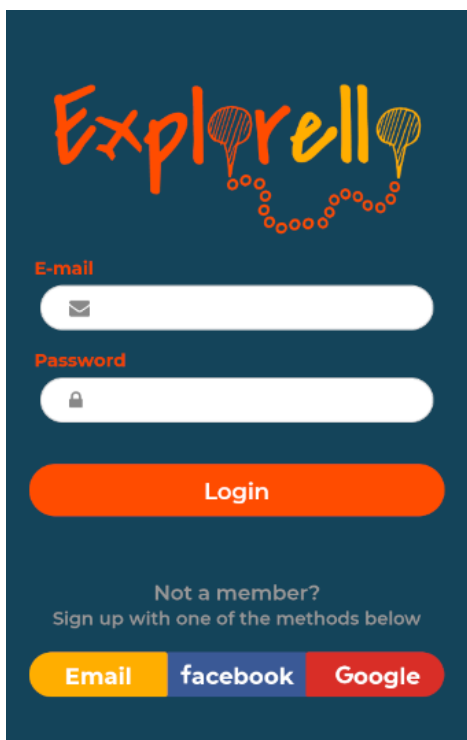


Fig 8. Ecranul de logare

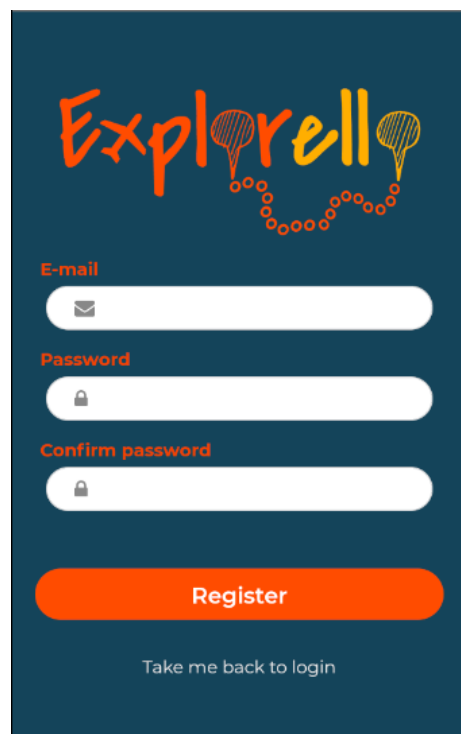


Fig 9. Ecranul de înregistrare

Înregistrarea, indiferent de modalitatea prin care utilizatorul alege să o facă, presupune și un mod de identificare mai personalizat în cadrul aplicației. Întrucât nu am dorit să aglomerez formularul de înscriere, în urma logării inițiale ale unui user, acesta va fi întâmpinat de ecranul pe care îl puteți observa în Fig. 10. Username-ul ales de user este salvat în Realtime Database, tabela *users*, identificatorul pentru fiecare user fiind cel prezent în tabela userilor înregistrați (Fig. 5).

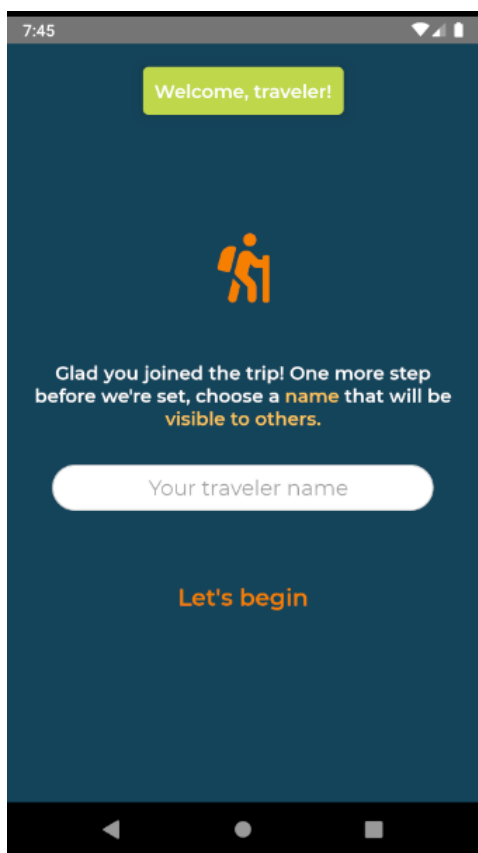


Fig 10. Ecranul de alegere a unui username

3.4.3. Interacțiunea cu utilizatorul

Pe parcursul utilizării aplicației, utilizatorul, în urma interacțiunii cu butoanele și componentele de tip *Touchable* relevante, va primi feedback în funcție de context prin intermediul unor *toast-uri*. Acestea au scopul de a confirma, ghida, sau corecta anumite acțiuni ale călătorului. Pachetul ce facilitează această caracteristică este *react-native-root-toast*. Aceste *toast-uri* sunt de 3 tipuri și semnifică:

1. **Succesul operației.** Conține o informație ce îi confirmă utilizatorului că operațiunea s-a îndeplinit cu succes.
2. **Eșecul.** Acesta conține o eroare, fie procurată de Firebase, fie la nivel de client și înștiințează utilizatorul că operațiunea a dat greș și furnizează eventuale recomandări.
3. **Ghidarea prin indicii.** În cursul folosirii unei funcționalități, scopul acestuia este de a conștientiza utilizatorul de toate uneltele de care dispune.

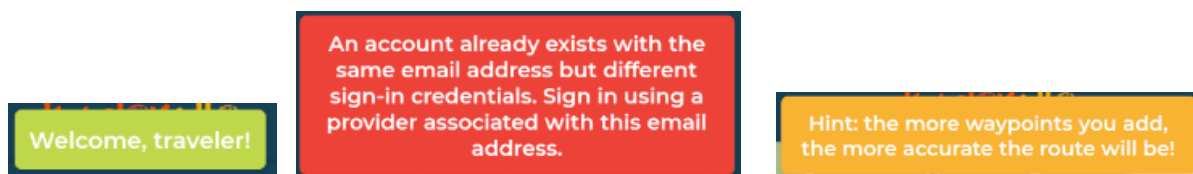


Fig 11. Cele 3 tipuri de notificări Succes, Eșec și Indiciu (de la stânga la dreapta)

3.4.4. Cursul aplicației

În urma autentificării, utilizatorul este întâmpinat de ecranul principal, un ecran ce abordează un stil simplist de punere la dispoziție a funcționalităților aplicației. Înainte de a încărca pagina, utilizatorului i se cere accesul la locație. În urma consimțământului utilizatorului, pagina de aterizare prezintă orașul curent în care se află dispozitivul utilizatorului, informație procurată prin intermediul serviciului de geo-localizare de la Google Maps API.

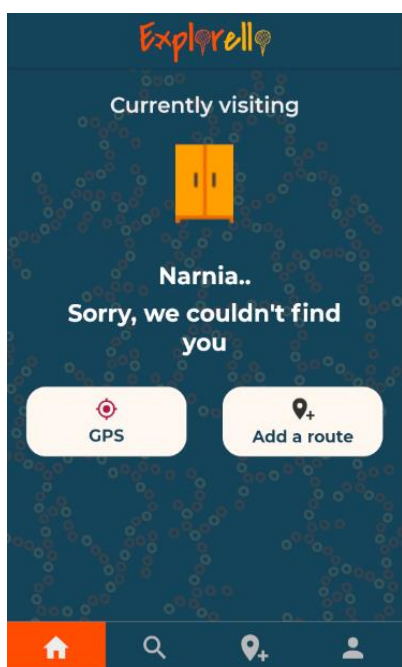


Fig 13. Utilizator nelocalizat

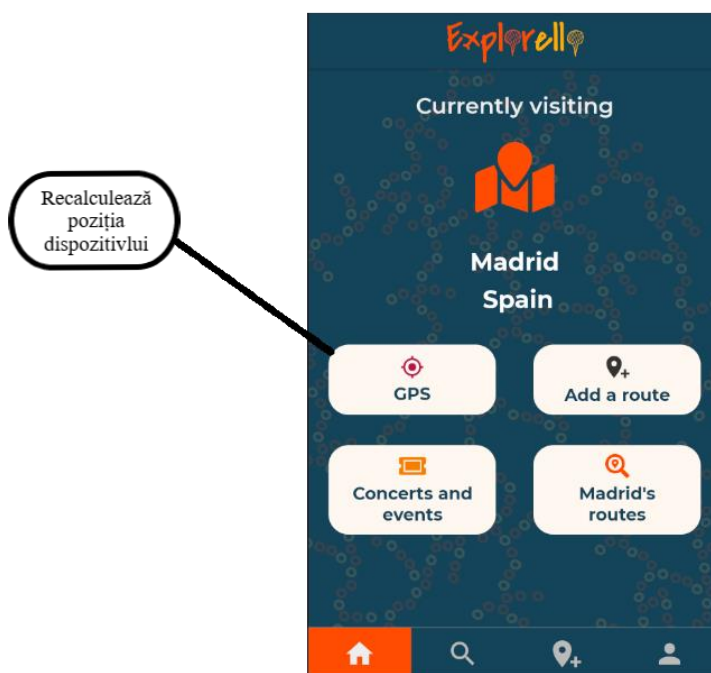


Fig 12. Dashboard

Apelarea API-ului pentru obținerea și parsarea răspunsului cu detaliile locației utilizatorului este precedată de stocarea aceste informații, apărând în istoricul utilizatorului. Utilizatorul va avea opțiunea de a șterge mențiunea din cadrul profilului său.

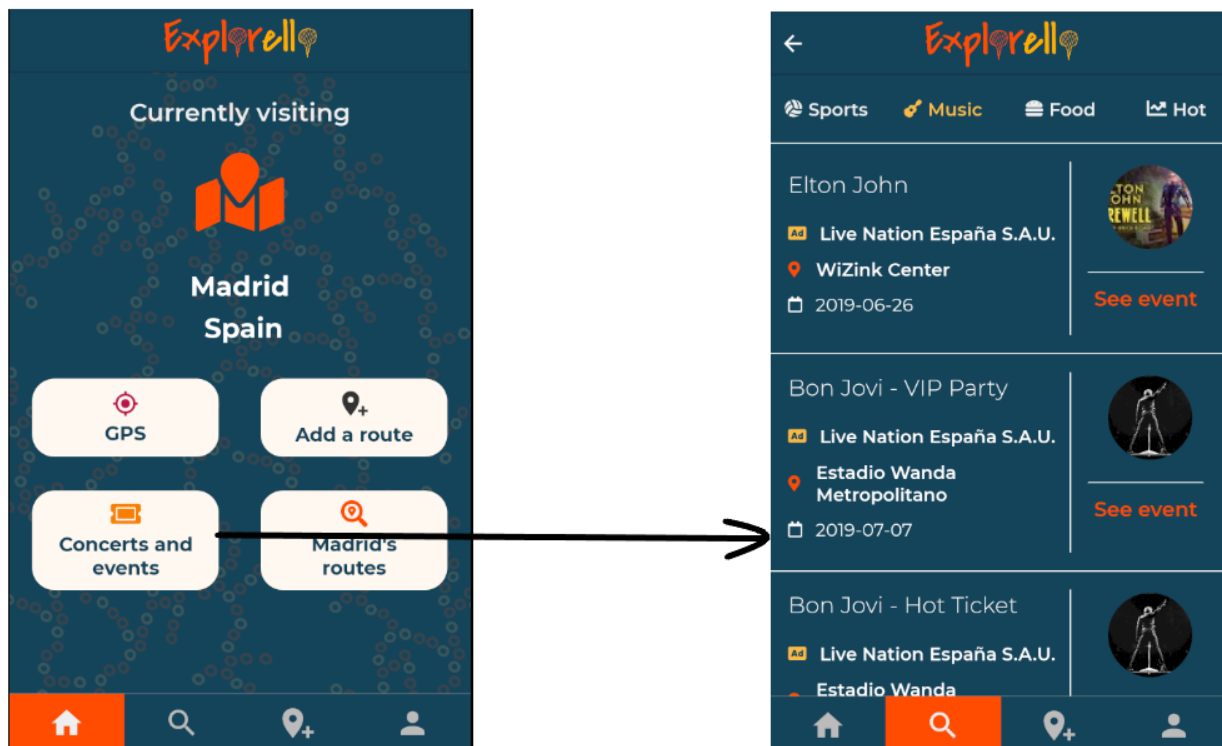


Fig 14. Lista de evenimente (dreapta)

Cele 4 funcționalități puse la dispoziție în dashboard sunt următoarele:

1. **GPS** – Are scopul de a recalcula poziția utilizatorului, în cazul unei erori sau actualizări ale locației (Exemplu: După un zbor) (Fig. 12)
2. **Concerts and events** – conduce utilizatorul la listarea evenimentelor din orașul în care se află, prin intermediul API-ului Ticketmaster. Intervalul de timp setat implicit este până la o săptămână din momentul actual. De asemenea utilizatorul poate să filtreze tipul evenimentelor în funcție de preferințele lui. Analog locației, un utilizator poate adăuga la istoric participarea la evenimente (Fig. 14).

3. **Add a route** – Conduce la funcționalitatea principală a aplicației și anume adăugarea unei rute. Cadrul de navigare pentru crearea itinerariilor începe prin pagina de întâmpinare care pune la dispoziție două opțiuni, cea de crearea unei rute manual, sau a unei rute create dinamic în timp ce utilizatorul este în mișcare (Fig. 16).
4. **City's route** – Navighează către pagina unde se listează rutele pentru locația curentă (Fig. 15). În cazul în care acestea nu există, utilizatorul va putea opta pentru generarea unei rute în funcție de anumiți parametri și anume tipurile obiectivelor turistice pe care acesta dorește să le marcheze. Listarea rutelor, prezentă și în tab-ul de navigare din josul ecranului, are în componența sa o bară de căutare după numele orașului în care dorești să găsești sau adaugi o rută.

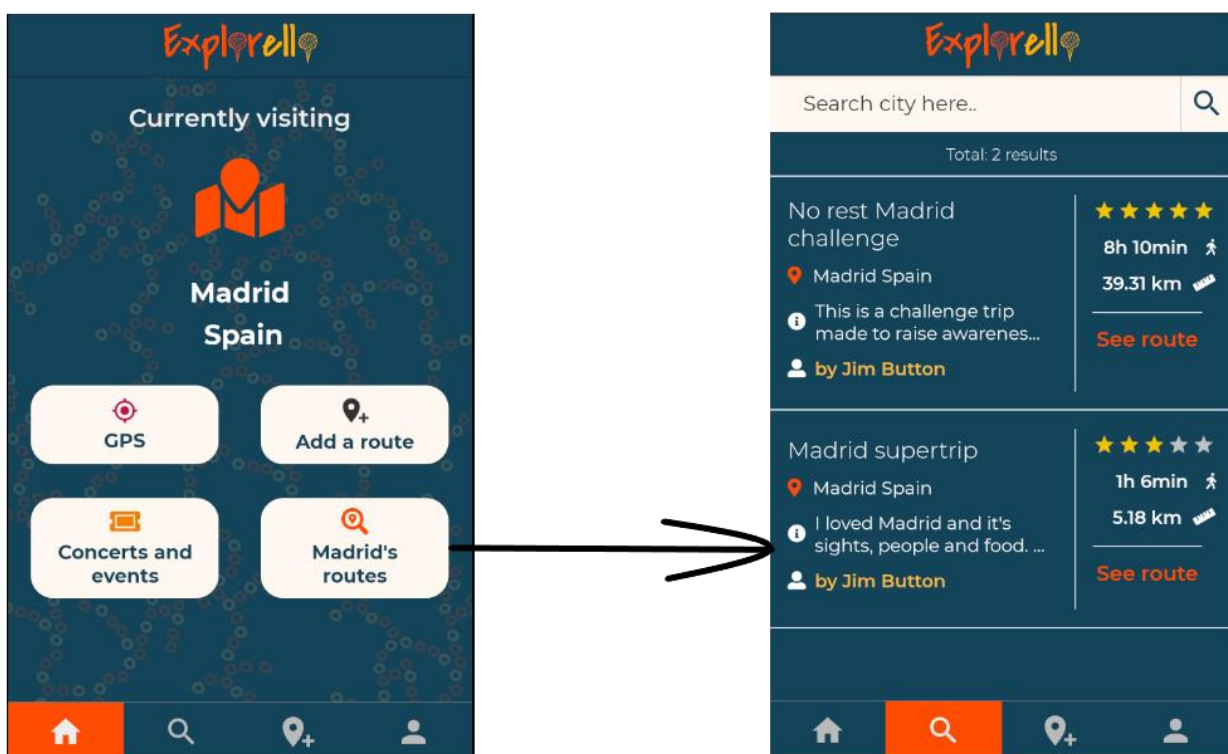


Fig 15. Listare de rute pentru locația curentă (dreapta)



Fig 16. Ecranul de întâmpinare pentru crearea unei rute (dreapta)

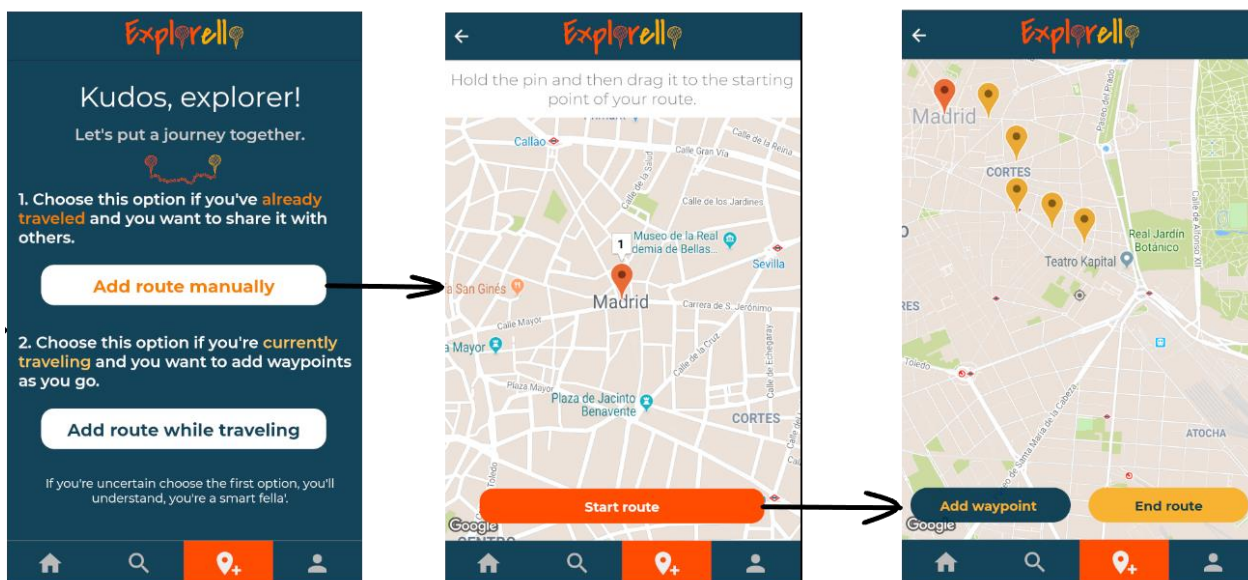


Fig 17. Ecranul de întâmpinare pentru crearea unei rute (dreapta)

Ambele modalități de adăugare, deși similare ca proces de creare, au scopuri diferite. Primul este de a crea o rută customizată, în care obiectivul creatorului este să aleagă o tematică și să o respecte. Cea de-a doua opțiune este adresată călătorilor ce doresc să țină o evidență a rutei parcurse de ei pe timpul unui sejur. Aceasta poate fi publică sau privată, după preferința utilizatorului.

Metoda de adăugare a unei rute este simplă și intuitivă pentru cel ce dorește să o creeze, punctul de start poate fi fie punctul central al orașului creat, fie locația curentă a utilizatorului. Acul roșu ce indică startul poate fi mutat în punctul de începere dorit. Odată setat, se poate trece mai departe, urmând să fie adăugate pe rând celelalte ace.

Concretizarea rutei se face în ecranul prezentat mai jos, unde se pot adăuga detalii privind ruta, eventuale lămuriri sau indicii. Numele rutei poate sugera scopul rutei sau tematica. Datele generate legate de distanță și durată sunt furnizate de Google Directions API (Fig. 18).



Fig 18. Ultimul pas din adăugarea unei rute

Vizualizarea unei rute, creată de tine sau de comunitate, se poate face fie din listarea rutelor în funcție de oraș, fie din cea ale rutelor din istoricul personal. (Fig. 19)

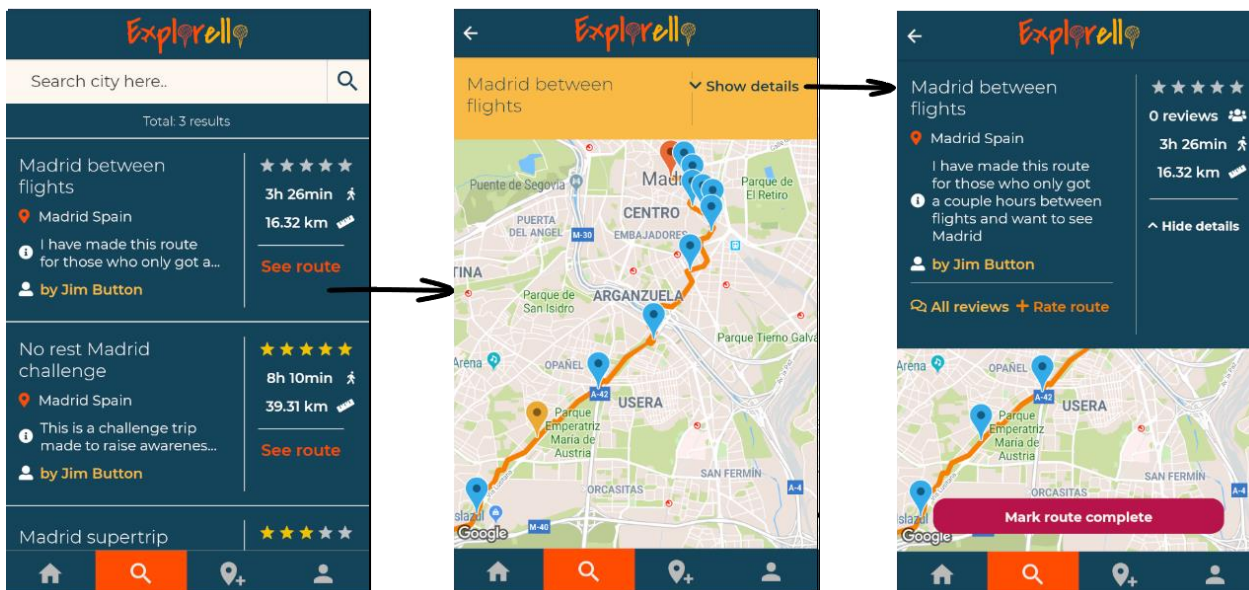


Fig 19. Detaliile unei rute

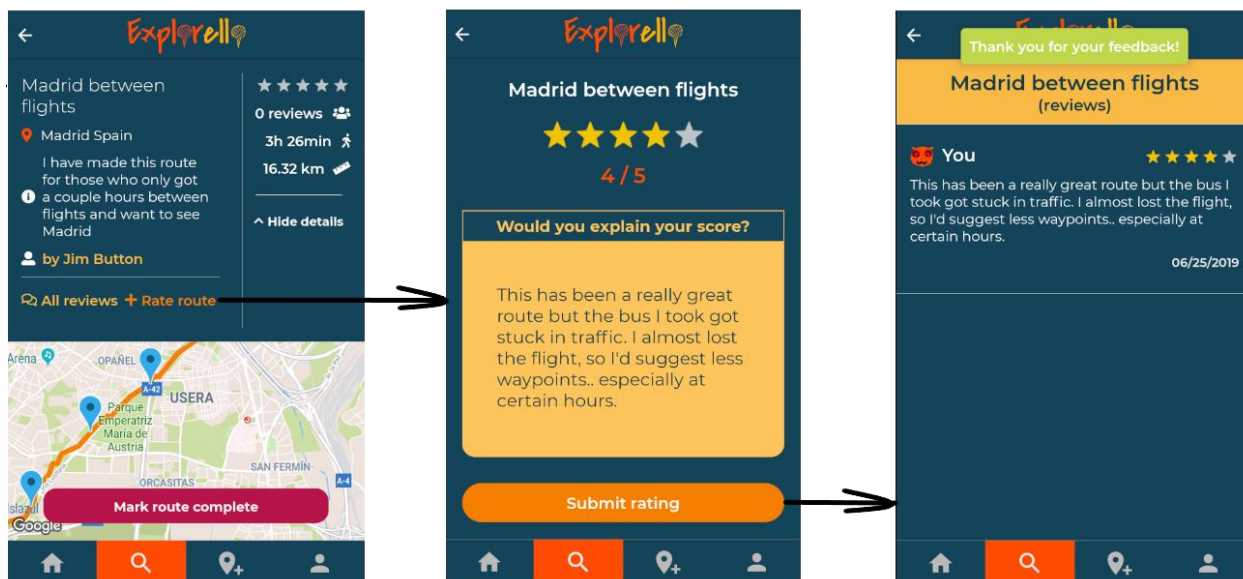


Fig 20. Adăugarea (mijloc) și listarea de recenzii (dreapta)

Componenta ce prezintă detalii despre o rută, afișează modalitățile prin care un utilizator poate interacționa cu o rută existentă. Acesta poate analiza harta, poate vedea lista recenziilor, adăuga unul la rândul său (Fig. 20) și nu în cele din urmă poate completa o rută. Completarea unei rute marchează ruta în istoricul utilizatorului și este îndemnat să lase o recenzie.



Fig 21. Profilul utilizatorului

Ultimul ecran din bara de jos este cel al profilului utilizatorului, acesta având posibilitatea să-și observe activitatea și să-și adauge orașul curent ca fiind *acasă* (Fig. 21). Recenziile unei rute dintr-o locație în care utilizatorul este rezident este evidențiat altfel și are o pondere mai mare. De asemenea, pentru a se deloga, utilizatorul va apăsa butonul din josul paginii. Aceasta va șterge de asemenea toate datele stocate în cache.

Concluziile lucrării

4.1. Concluzii

Această aplicație se adresează unei probleme destul de specifice din contextul călătoritului, iar în urma unui sondaj de opinie și analizării unui eșantion de indivizi interesați de astfel de activități, majoritară a fost reacția pozitivă în raport cu conceptul aplicației.

Explorello se adresează mai multor aspecte. În primul rând, la importanța contribuției nomazilor la planificarea unor itinerarii menite să se adreseze unor anumite categorii de oameni. În al doilea rând este o platformă de auto-gestionare și ținerea evidenței a activităților ce au loc pe parcursul unui sejur, un aspect organizatoric esențial pentru cei care călătoresc cu regularitate.

În concluzie, această aplicație este un partener de călătorie simplu, prietenos și foarte folositor. Poate fi o metodă de a apropia persoanele din comunitatea nomadă, astfel reușind să fie nu doar o unealtă avantajoasă, ci și o platformă socială ce îndeamnă la sinergie.

4.2. Direcții viitoare

Având în vedere actualele funcționalități și avantajele pe care o populare masivă a bazei de date le-ar aduce, rutele ar deveni din ce în ce mai precise și practice, utilizatorii voi fi clasificați în funcție de nivelul de contribuție din cadrul aplicației, iar în cele din urmă un sistem de *vânătoare de comori* ar putea fi implementat.

De asemenea un modul mai complex în ce privește aspectul evenimentelor precum concertele sau competițiile sportive ar putea fi implementat, permițând utilizatorului să procure bilete prin intermediul utilizatorilor ce sunt rezidenți în orașul respectiv evenimentului.

Bibliografie

1. React-Native: <https://facebook.github.io/react-native/>
2. Redux: <https://redux.js.org/introduction/motivation>
3. Firebase Realtime Database: <https://firebase.google.com/docs/database>
4. Redux thunk: <https://github.com/reduxjs/redux-thunk/>
5. React Diagram: <https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8>
(Fig 3)
6. Google Maps API: <https://developers.google.com/maps/documentation/>
7. Google Directions API: <https://developers.google.com/maps/documentation/directions/start>
8. Google Geolocation API: <https://developers.google.com/maps/documentation/geolocation/intro>
9. Google Places API: <https://developers.google.com/places/web-service/intro>
10. react-native-maps: <https://github.com/react-native-community/react-native-maps>
11. react-native-google-places-autocomplete: <https://github.com/FaridSafi/react-native-google-places-autocomplete>
12. rnfirebase: <https://rnfirebase.io/>
13. react-native-fbsdk: <https://github.com/facebook/react-native-fbsdk>
14. react-native-geolocation-service: <https://github.com/Agontuk/react-native-geolocation-service>
15. react-native-google-signin <https://github.com/react-native-community/react-native-google-signin>
16. react-native-image-sequence: <https://www.npmjs.com/package/react-native-image-sequence>
17. react-native-maps: <https://github.com/react-native-community/react-native-maps>
18. react-native-maps-directions: <https://www.npmjs.com/package/react-native-google-maps-directions>
19. react-native-modal: <https://github.com/react-native-community/react-native-modal>
20. react-native-root-toast: <https://github.com/magicismight/react-native-root-toast>
21. react-native-vector-icons: <https://github.com/oblador/react-native-vector-icons>
22. React Navigation: <https://reactnavigation.org/>
23. Google Functions: <https://cloud.google.com/functions/>
24. Ticketmaster Developer: <https://developer.ticketmaster.com/products-and-docs/apis/getting-started/>
25. Lonely Planet: <https://www.lonelyplanet.com/about/story>
26. Visit a City: <https://www.visitacity.com/>
27. Redux-Persist: <https://github.com/rt2zz/redux-persist>