

Предмет: Мобилни информациски системи

**Документација за мобилна апликација**

**TravelPoint**

**Професор:** **Изработиле:**

Д-р Петре Ламески Душко Аврамоски, 191143

Славко Анакиевски, 191173

Ана Дрогришка, 191024

Содржина

[1. Вовед 3](#_Toc145884310)

[2. Користени технологии 3](#_Toc145884311)

[3. Архитектура 3](#_Toc145884312)

[3.1 База на податоци 3](#_Toc145884313)

[3.2 Frontend/ Backend 4](#_Toc145884314)

[3.2.1 Assets folder 4](#_Toc145884315)

[3.2.2 Theme file 5](#_Toc145884316)

[3.2.3 Core folder 5](#_Toc145884317)

[3.2.4 Features folder 6](#_Toc145884318)

[3.2.5 Firebase\_options file 11](#_Toc145884319)

[3.2.6 Injection\_container file 11](#_Toc145884320)

[3.2.7 Main file 11](#_Toc145884321)

[3.2.8 Останати фајлови 11](#_Toc145884322)

[4. Кориснички сценарија 11](#_Toc145884323)

[4.1 Форма за регистрација 11](#_Toc145884324)

[4.2 Форма за најава 12](#_Toc145884325)

[4.3 Поглед ‘Explore’ 12](#_Toc145884326)

[4.4 Поглед ‘Home’ 14](#_Toc145884327)

[4.5 Поглед ‘Near me’ 15](#_Toc145884328)

[4.6 Информации за корисник 16](#_Toc145884329)

[4.7 Верификација на е-маил 16](#_Toc145884330)

# Вовед

Се стремиме да креираме апликација која ќе му овозможи на крајниот корисник, лесно, брзо и ефективно да ja пронајде најдобра локација во светот според однапред внесени негови побарувања зависно за живеење (преселба) или за годишно патување на одмор. Нашата идеја се сведува на апликација базирана на географска карта, со податоци за локации како болници, градинки, училишта, седишта на компании, плажи, кафичи, маркети и многу други институции и установи, кои што ќе ги искористиме за да им ја овозможиме најдобрата понуда на корисниците на нашата апликација за идеално место за патување или препорачано место за живеење. TravelPOINT е мобилна апликација која им помага на луѓето да откријат нови и интересни места за патувања. Вклучува функција за пребарување, која им овозможува на корисниците да бараат држави, градови, општини или рачно внесени коориднати како дестинации, како и приказ на карта со локацијата на различни места од интерес. Апликацијата ќе вклучува и прегледи и оценки од корисниците. Со користење на апликација за наоѓање на најдобрите локации за патување, патниците можат лесно да истражуваат нови дестинации и да ги најдат совршените места за посета.

Целта на овој документ е полесно да се разбере целиот код, тек и кои се главни функционалности на апликацијата преку прикажани различни кориснички сценарија.

# Користени технологии

За развојот на оваа мобилна апликација се користени технологии кои што се моментално многу актуелни и се користат секојдневно во многу ИТ компании.

За градење на сите функционалности на апликација, користевме Flutter што претставува крос-платформски SDK (Software Development Kit) што може да се користи за градење на апликации како и за frontend, така и за backend апликации, во различни програмски јазици како Dart, Java, C/C++ и многу други. Нашата апликација најголем процент е развивана во Dart. Flutter претставува моќен framework, каде што неговите алатки го олеснуваат градењето на висококвалитетни мобилни апликации за кој било намена. За чување на нашите податоци користевме Firebase, што претставува множество на cloud услуги и платформи за развој на апликации обезбедени од Google. Се состои од бази на податоци, сервиси, автентикација и интеграција на различни апликации вклучувајќи Android и iOS. Земање на вредностите на локации се врши преку Google API.

# Архитектура

## База на податоци

Во нашата база на податоци се наоѓа само една табела, тоа е табелата за Authentication. Во табелата Authentication се сместуваат податоците на корисниците при нивна регистрација/ најава, како меил, пасворд, корисничко име, тип на провајдер преку кој што е најавен, датум на креирање на профилот, датум на последна најава од страна на корисникот и слично. Користени се голем број на готови библиотеки кои ги нуди Firebase за автентикација, со прилагодувања за нашите потреби. Користени се два методи на регистрација и најава: стандардниот начин преку email и password, а друга можност е најава и регистрација преку Google. Овозможена е и email верификација, каде што кога корисникот ќе се најави преку email/password метод, има можност да му се испрати конформациски меил, со цел да се верификува неговата регистрирана меил адреса. Направен е меил темплејт, повторно со помош на сервисите кои ги нуди Firebase, каде што се испраќа меил до корисникот со линк за верификација. Сите останати податоци се црпат од надворешни API, затоа во нашата база има податоци само за корисникот.

## Frontend/ Backend

За подобра работа и организираност, користевме Bloc шаблон, кој претставува шаблон за управување со состојби што вообичаено се користи во Flutter апликациите. Тоа помага да се управува со состојбата на апликацијата и го олеснува градењето сложени апликации. Овој шаблон е дизајниран да ја оддели бизнис логиката на апликацијата од интерфејсот. Со тоа, се олеснува одржувањето и менувањето на кодот, бидејќи промените во деловната логика не влијаат на презентацискиот слој. Во овој модел бизнис логиката на апликацијата е сместена во Блок класа. Таа класа прима влезови во форма на настани (промени во состојбата на апликацијата кои треба да се обработат) и произведува излези во форма на состојби (моментална состојба на апликацијата). Презентацискиот слој прима ажурирања и промени во состојбата на апликацијата. Ваквата интеракција обично се прави со помош на BlocBuilder, кој го обновува интерфејсот како одговор на промените во состојбата на апликацијата. Во продолжение следи целата организираност на кодот.

### Assets folder

Во овој фолдер се сместени фајлови кои се распоредени низ апликацијата и се достапни при извршување. Во нашиот случај тука е сместена слика од логото на апликацијата кое понатаму е видливо во неколку погледи.

### Theme file

Во овој фајл се дефинирани сите основни стилизирања на апликацијата кои подоцна се користат во UI компонентите. Тука се сместени примарни бои, фонтови, облик на копчиња и слично. Дефинирањето се извршува на едно место со цел полесно и побрзо менување на стилови и подобра организираност, со тоа што се се наоѓа на едно место.

### Core folder

Тука се сместени различни константи, грешки, типови, use-cases, валидатори на форми и widgets, кои претставуваат елементи за интеракција во корисничкиот интерфејс.

**А) constants folder**

*constants.dart* - Дефинирани константи (пр. API\_KEY), за поедноставно користење и лесно пристапување до нив во други делови од кодот.

**Б) errors folder**

*1. exception.dart*

Исклучоци кои може да се фрлат во текот на извршување на апликацијата и справување со нив.

*2. failure.dart*

Слично на исклучоците, генерирање на грешка при неуспешно внесување на податоци за корисник или неуспешен повик до API.

**В) type folder**

*type\_def.dart* - фајл каде се дефинирани различни типови на категории за избор, кои се користат во понатамошниот тек, конкретно делот со прикажување на места на мапа.

**Г) usecase folder**

**Д) utils folder**

*form\_validators.dart* – фајл каде што се дефинирани валидатори за форми, како на пример валидатор за задолжително поле.

**Ѓ) widgets folder**

фолдер со *widgets*, кои претставуваат готови елементи, прилагодени за нашите потреби, за интеракција во кориснички интерфејс и секој од нив претставува по еден елемент. Тука се наоѓаат: bottom bar , drawer bar и top bar, како елементи за навигација, потоа форми, прозорци со некакви известувања, sidebar за детални информации и слично. Сите овие може да се реискористуваат во различни погледи низ апликацијата (views).

### Features folder

Овој “главен“ дел е поделен на features и секое од нив во себе содржи три слоеви: data, domain и presentation layer. Всушност тука е применет Bloc шаблонот кој што беше споменат претходно. Имаме два главни features: map и authentication.

1. **Authentication feature**
2. *Data layer –* за интеракција и преземање податоци од различни извори (конкретно базата на податоци)

***Datasource/auth\_remote\_data\_source.dart*** – тука се одвива комуникацијата со база и тука има неколку функции:

* *getCurrentAuthFirebaseUser()* – го враќа моментално најавениот корисник, доколку нема таков фрла исклучок
* *getCurrentUser()* – враќа модел од моментално најавен корисник
* *loginUser()* – функција за најава на корисник
* *signUpUser()* – функција за регистрација на корисник
* *getUsernames()* – ги враќа корисничките имиња на сите корисници
* *loginWithGoogle()* – функција за најава преку Google
* *logoutUser()* – функција за одјава на моментално најавен корисник
* *sendEmailVerification()* – функција за испраќање на мејл за верификација
* *updateUser()* – додавање или ажурирање на нов корисник
* *getEmailVerifiedFlag()* – функција која враќа резултат дали меилот е верификуван или не

***models/user\_model.dart –*** дефинирано е од кои полиња се состои user моделот

***repository/auth\_repository\_impl.dart –*** имплементација на функции поврзани со автентикација на корисник

1. *Domain layer –* врска меѓу data и presentation слојот

***entity/user.dart –*** дефиниран ентитет за корисник

***repository/auth\_repository.dart –*** функции веќе дефинирани во data layer repository се проследуваат кон презентациски слој

***usecase/:*** Тука се земени предвид различни кориснички сценарија и за секое од нив имаме соодветно справување со нив.

* *check\_email\_verify\_user.dart*
* *email\_verify\_user.dart*
* *get\_current\_user.dart*
* *login\_user\_with\_google.dart*
* *login\_user.dart*
* *logout\_user.dart*
* *signup\_user.dart*

1. *Presentation layer*

***Bloc folder****:*

* *auth\_bloc.dart* – фајл одговорен за ракување (справување) со влезните настани, за производство на нови состојби врз основа на настаните и моменталната состојба на апликацијата
* *auth\_event.dart* – настани кои се влез во Bloc. Тие претставуваат дејства или појави кои може да предизвикаат промена во состојбата. Такви настани се: *SignupAuthEvent,*

*LoginWithGoogleAuthEvent,*

*LogoutUserAuthEvent,*

*CheckEmailVerifyUserAuthEvent,*

*EmailVerifyUserAuthEvent,*

*GetCurrentUserAuthEvent,*

*LoginAuthEvent.*

Сите наследуваат од AuthEvent и се самоопишувачки.

* *auth\_state.dart* – се претставува моменталната состојба на апликацијата во било која дадена временска точка. Има повеќе состојби кои може да се јават кај еден корисник и се дефинирани во овој фајл и наследуваат од AuthState:

*InitialAuthState,*

*LoadingAuthState,*

*ResultAuthState,*

*ErrorAuthState,*

*CheckEmailVerifyState,*

*CurrentUserState,*

*UserActiveOnAppState.*

Сите овие состојби се самоопишувачки.

***Views folder:*** Во овој фолдер се сместени сите погледи поврзани со некакво прикажување или внесување податоци за корисник. Има неколку погледи:

* *home\_page.dart –* почетен поглед на корисникот откако ќе направи најава/регистрација, каде што е прикажана мапа, top bar, bottom bar и drawer.
* *login\_page.dart –* поглед каде што има форма за најава на корисник, со можност за најава на стандарден email/password начин и преку Google и можност за навигирање до погледот регистрација на корисник
* *signup\_page.dart –* поглед каде што е прикажана форма за регистрација на корисник и можност за навигирање до погледот за најава на корисник
* *userInfo\_page.dart –* поглед каде што се прикажани информации за најавениот корисник како: корисничко име, мејл, датум на креирање и модификација, дали мејлот е верификуван и можност за верификација на мејл доколку тоа не е направено.
* *auth\_page.dart –* поглед каде што се навигира помеѓу погледот за најава и погледот за регистрација

1. **Map feature –** на ист начин како и кај Authentication feature имаме поделба на 3 слоеви:
2. *Data layer -* за интеракција и преземање податоци од различни извори (Google API)

***Datasource/map\_remote\_data\_source.dart –*** тука има комуникација со API и има неколку функции:

* getCurrentLocation() – бара дозвола и ја добива моменталната локација на корисникот
* getNearbyPlaces() – дава места во внесен радиус, од избрана категорија
* getPredictionsFromAutoComplete – autocomplete при внесување на места
* getPlaceFromPlaceId() – враќа детални информации за место со дадено ID
* getDistanceForNearbyPlaces() – враќа растојание на места во близина

***models folder:***

* distance\_matrix\_response.dart – дефинирано од кои полиња се состои матрицата од растојанија
* nearby\_places\_response.dart -
* place\_model.dart – полиња од кои се состои моделот за место

***repository/map\_repository\_impl.dart -***

1. *Domain layer –* врска меѓу data и presentation слојот

**Repository/map\_repository.dart –**

**Usecase:** Различни кориснички сценарија и справување со секое од нив соодветно

* *get\_distance\_nearby\_places.dart*
* *get\_nearby\_places.dart*
* *get\_place\_details.dart*
* *get\_search\_autocomplete\_prediction.dart*
* *get\_user\_current\_location.dart*

1. *Presentation layer*

***Bloc folder:***

* *map\_bloc.dart* – фајл одговорен за ракување (справување) со влезните настани, за производство на нови состојби врз основа на настаните и моменталната состојба на апликацијата
* *map\_event.dart* – настани кои се влез во Bloc. Тие претставуваат дејства или појави кои може да предизвикаат промена во состојбата. Такви настани се:

*GetCurrentLocationEvent,*

*GetCurrentLocationNearbyPlacesEvent,*

*GetChosenLocationNearbyPlacesEvent,*

*ClearMarkersEvent,*

*GetDistanceForNearbyPlacesEvent,*

*GetPredictionsFromAutocompleteEvent,*

*GetPlaceFromPlaceIdEvent,*

*ResetStateToInitialEvent*

Сите наследуваат од MapEvent и се самоопишувачки.

* *map\_state.dart* – се претставува моменталната состојба на апликацијата во било која дадена временска точка. Има повеќе состојби кои може да се јават кај еден корисник и се дефинирани во овој фајл и наследуваат од MapState:

*InitialMapState,*

*LoadingMapState,*

*ResultMapState,*

*ErrorMapState,*

*ResultDistanceMatrixState.*

Сите овие состојби се самоопишувачки.

***Views folder:*** Во овој фолдер се сместени сите погледи поврзани со некакво прикажување или внесување податоци за приказ на податоци од или на мапа. Има неколку погледи:

* *details\_sidebar\_marker.dart –* во овој поглед имаме приказ на drawer каде што се прикажани детални информации за места во близина кои сме ги пребарувале.
* *get\_nearby\_places\_form.dart –* во овој поглед кој се појавува со клик на копче, се прикажува форма каде што најпрво селектираме кои типови на места сакаме да бидат прикажани, а потоа во наредното поле внесуваме радиус во метри за тоа на колкава најголема оддалеченост да бидат тие места од нас. Тука имаме и валидатори кои доколку не е внесена никаква вредност или е внесен радиус поголем од 50 000, соодветно да се појави порака за предупредување.
* *map\_page.dart –* во зависност од тоа што сме одбрале во bottom bar, не’ води во различен поглед. Првиот поглед ‘*Explore’* ни дава можност да пребаруваме места на мапа со опција search, а потоа не’ води до погледот *get\_nearby\_places\_form,* вториот поглед e ‘Find home’ е обичен поглед кој се состои само од мапа и третиот е ‘Near me’ поглед каде што постои опција за лоцирање на моментална локација на корисник и пребарување на места близу до него повторно преку формата *get\_nearby\_places\_form.*
* *marker\_info.dart –* поглед кој што се појавува на некој од маркерите прикажани на мапа при пребарување на места. Овој поглед содржи информации за тоа место како име, рејтинг, категорија и слично.

### Firebase\_options file

Конфигурациски фајл каде се внесени Firebase Options поврзани со базата на податоци за повеќе оперативни системи како macOS, iOS и android. Останатите платформи не се конфигурирани за користење.

### Injection\_container file

Во овој фајл, кодот иницијализира и регистрира зависности користејќи го пакетот GetIt, контејнер за инјектирање на зависност кој ви овозможува регистрирање и преземање истанци од обејкти.

### Main file

Во овој фајл се дефинира кој ќе биде почетен поглед при отварање на апликација. Доколку не е извршена најава/регистрација тогаш се отвара поглед за најава, доколку корисникот е најавен се отвара HomePage погледот со мапата и останатите функционалности.

### Останати фајлови

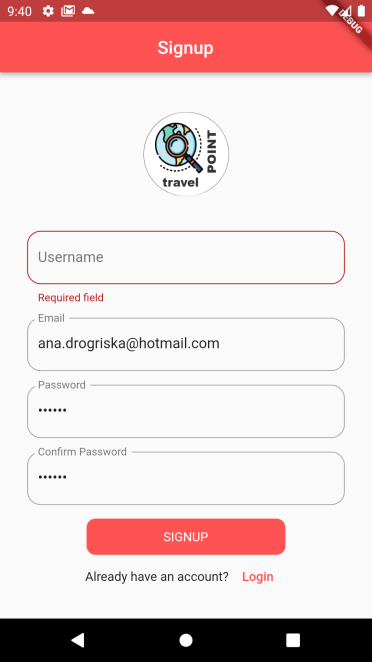
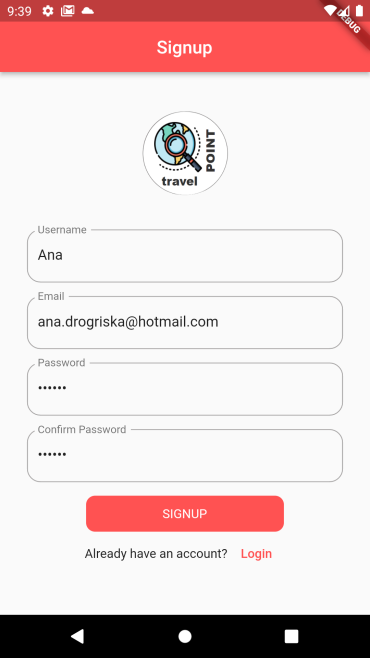
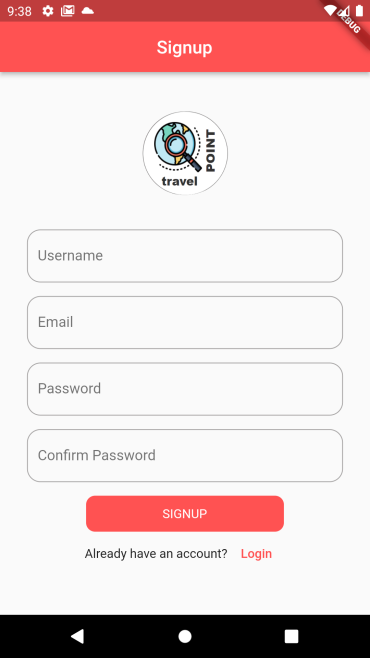
***Pubspec.yaml –*** на секој pub пакет му требаат метаподатоци за да може да ги специфицира своите зависности. Pub пакетите што се споделуваат со други, исто така, треба да обезбедат некои други информации за корисниците да можат да ги откријат. Тие метаподатоци се запишани во оваа датотека на јазикот YAML.

# Кориснички сценарија

Визуелно прикажување на сите сценарија кои се очекувани и достапни на корисничкиот интерфејс се покриени во оваа секција.

## Форма за регистрација

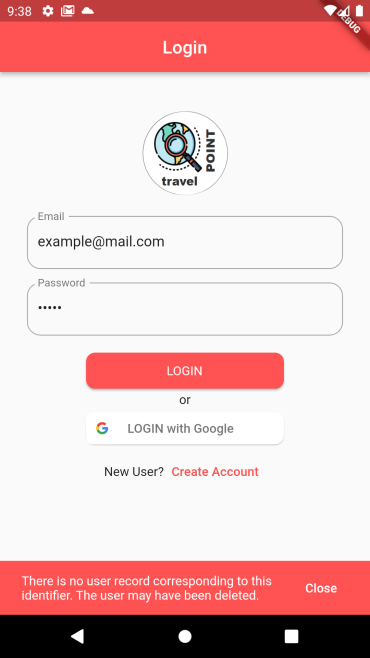
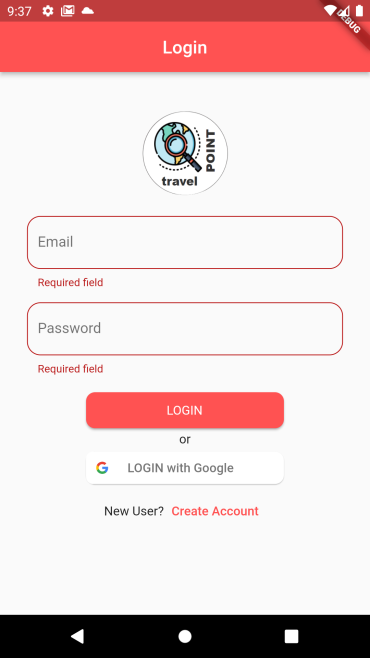
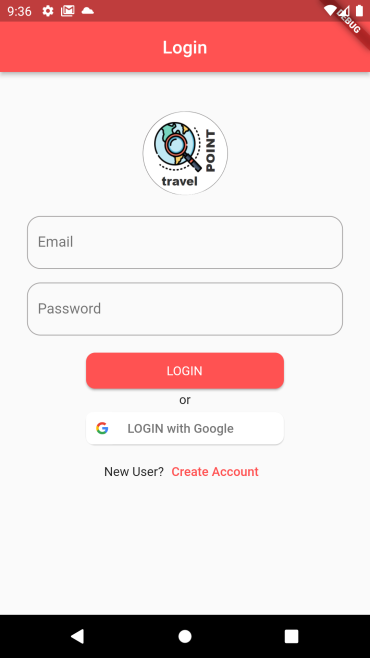
Изглед на празна форма (Слика 1), целосно пополнета форма (Слика 2) и појава на валидациска грешка доколку некое поле не е пополнато (Слика 3)



*Слика 1,2,3*

## Форма за најава

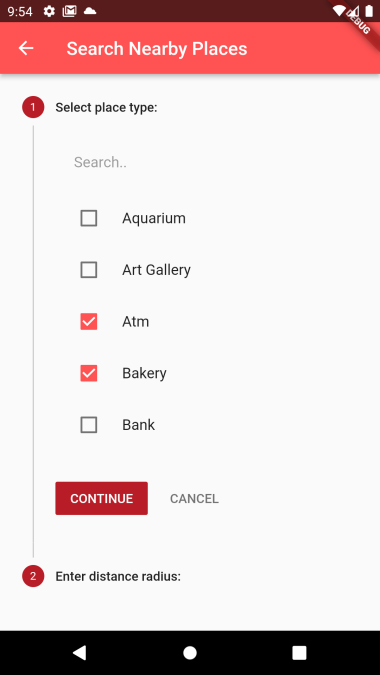
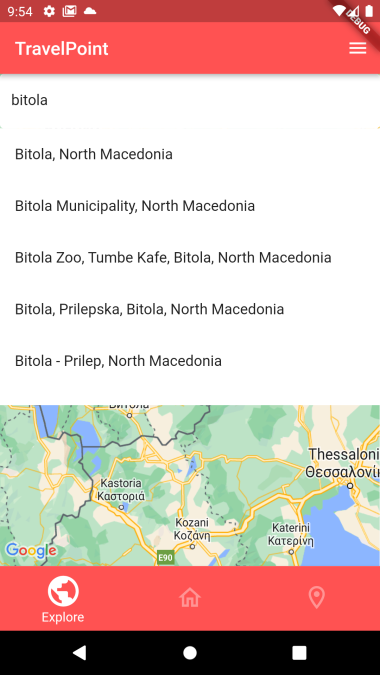
Изглед на празна форма (Слика 4), изглед при појава на валидациска грешка за непополнато задолжително поле (Слика 5) и излед при внесени мејл кој досега не е регистриран (Слика 6).



*Слика 4,5,6*

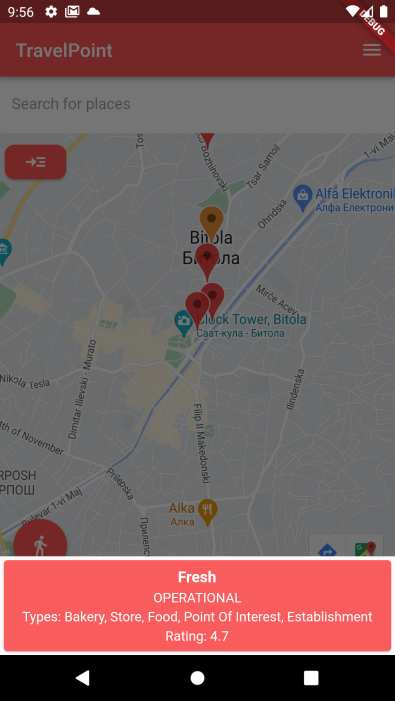
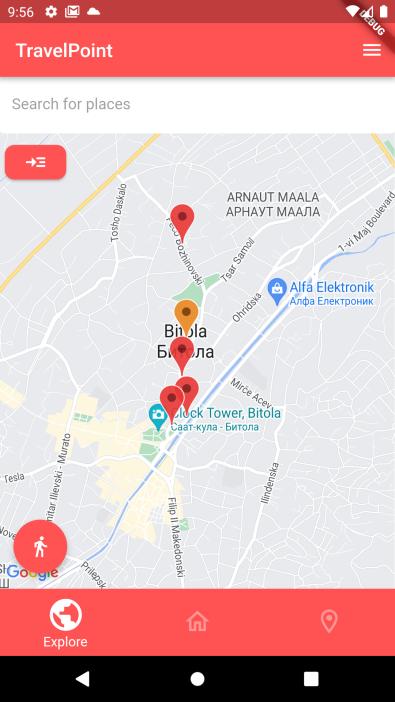
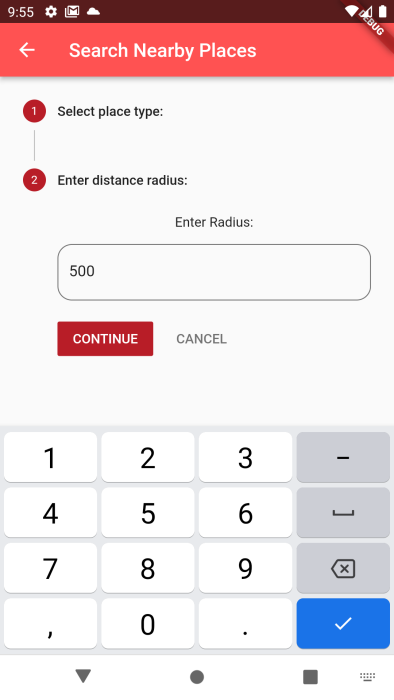
## Поглед ‘Explore’

При успешна најава/регистрација на корисникот, се пренасочува кон погледот ‘Explore’ (Слика 7). При клик на search bar-от се појавува опција за внесување на место (пр. град) (Слика 8), по внесувањето на место се појавува форма за избирање на тип на место (задолжително да се избере барем едно) (Слика 9), а потоа се внесува радиус во метри (Слика 10), што претставува најголемо можно растојание на кое може да се наоѓаат бараните места од претходно внесената локација.

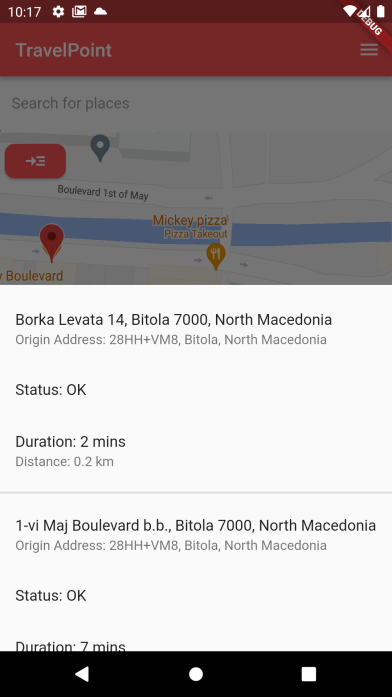
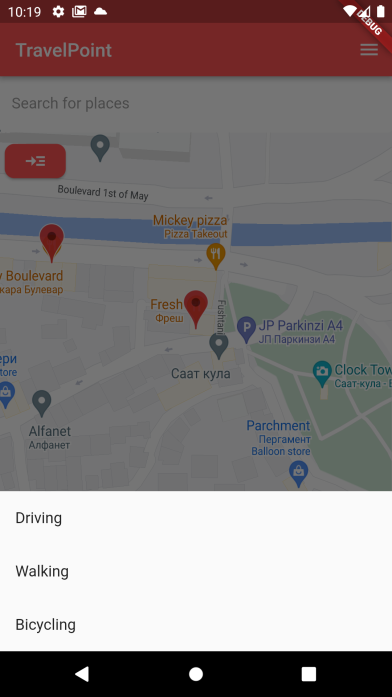
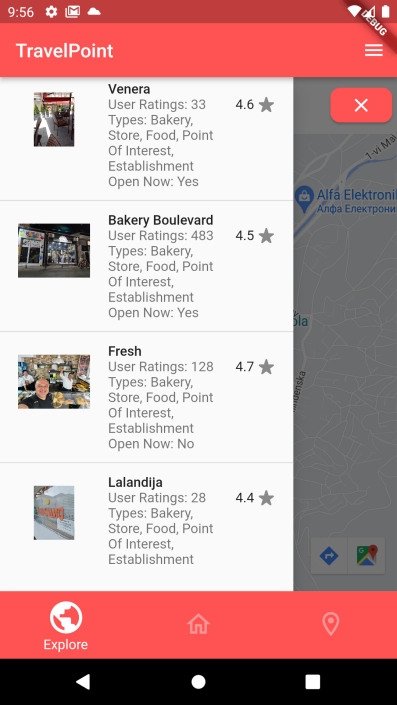


*Слика 7,8,9*

По внесување на овие параметри на мапа се прикажуваат сите места од избраните категории кои се наоѓаат во дадениот радиус (Слика 11). Со клик на на Drawer-от кој се наоѓа во левиот горен агол се прикажуваат детални информации за секое од тие места (Слика 12). Доколку кликнеме на некој од маркерите на мапата во долниот дел од екранот ни се прикажува детално инфо за тоа место (Слика 13). При клик на човечето во долен лев агол, може да се избере една од трите опции за начин на патување до одредедено место (Слика 14). Кога ќе избереме една од тие опции ни дава рута до тоа место и времетраење на истото (Слика 15).



*Слика 10,11,12*



*Слика 13,14,15*

Друга опција која ни се нуди е опцијата за отворање на пребаруваното место на Google Maps со клик на иконите во долен десен агол и добивање насоки за движење на мапа.

## Поглед ‘Home’

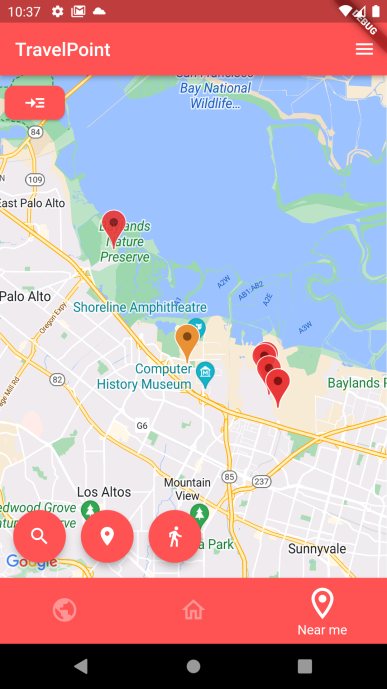
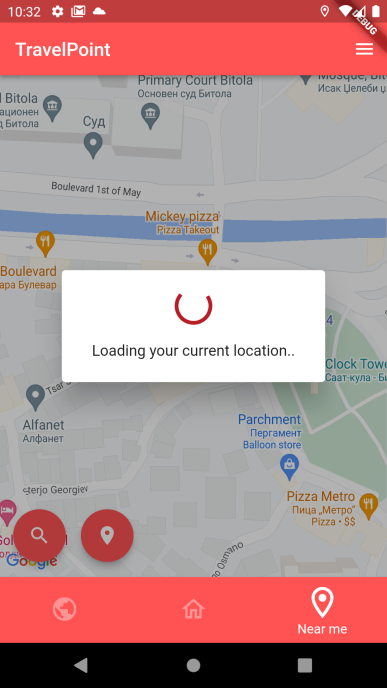
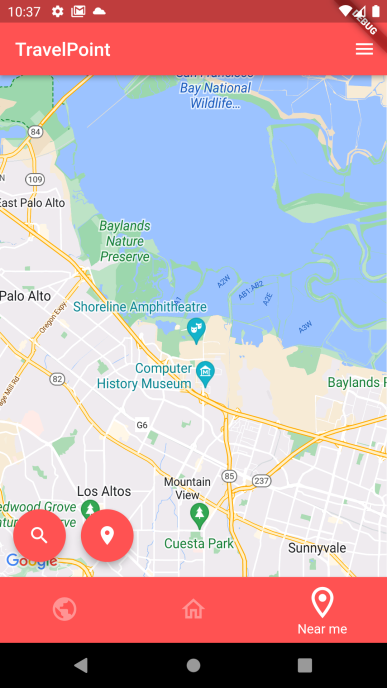
При клик на делот Home во bottom bar-от се прикажува обична мапа со можност за разгледување места на самата мапата, без некои додатни функционалности (Слика 16).



*Слика 16*

## Поглед ‘Near me’

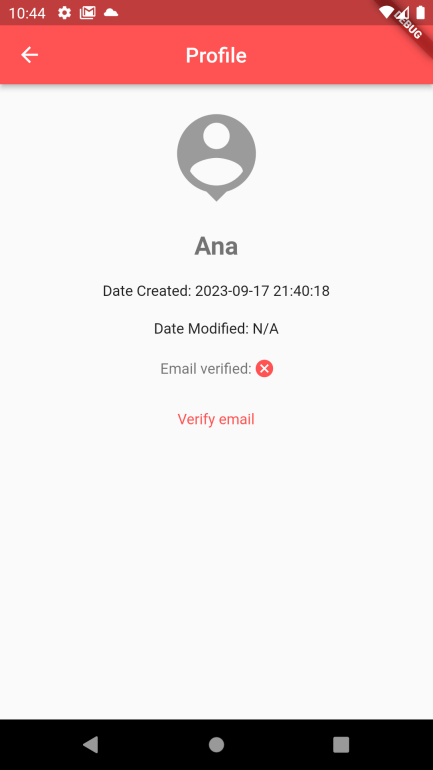
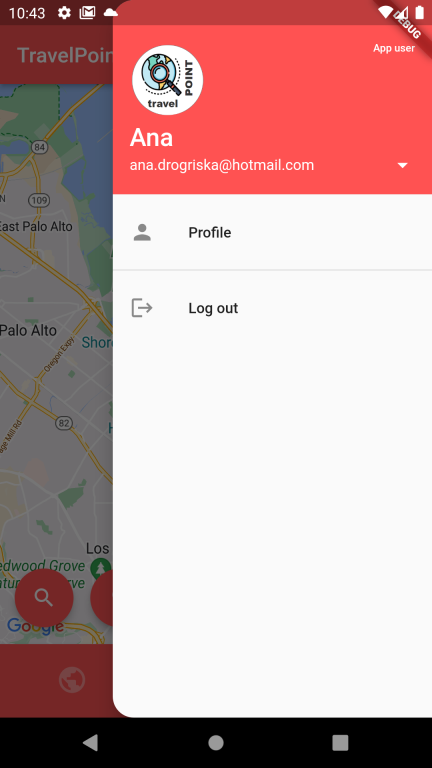
При клик на ‘Near Me’ делот во bottom bar-от се прикажува поглед со мапа (Слика 17), каде што се пристапува до моменталната локација на корисникот (Слика 18) и преку истата форма за внесување на тип на место и радиус во погледот ‘Explore’ се прикажуваат сите избрани типови на места со максимален радиус од тој што е внесен од моменталната локација на корисникот (Слика 19). Тука повторно имаме Drawer со детални информации од сите добиени места, детални информации при клик на некој од прикажаните маркери и можност за добивање на насоки и времетраење до пристигнување до дадено место.



*Слика 17,18,19*

## Информации за корисник

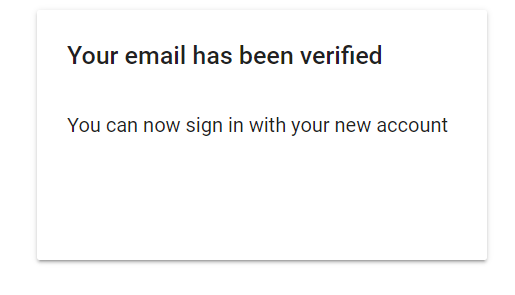
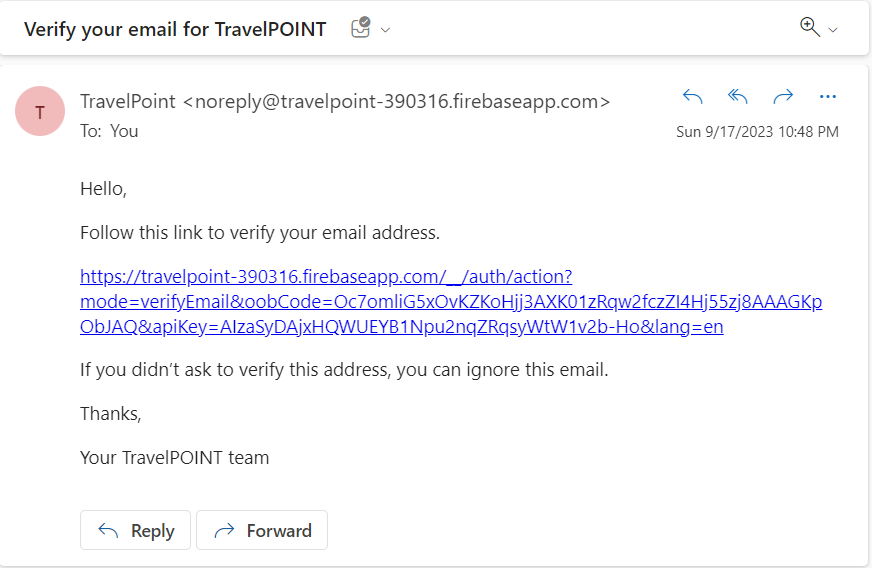
При клик на трите линии во горен десен агол се прикажува Drawer каде што се прикажани корисничкото име и мејлот на корисникот, а под него копчиња кои водат до Profile Info погледот и Logout на корисникот (Слика 20). При клик на Log out се прави одјава на корисникот и го води кон Login Page. При клик на Profile се прикажуваат информации за корисникот како корисничко име, датум на креирање, ажурирање на профилот и информација дали мејлот е верификуван (Слика 21).



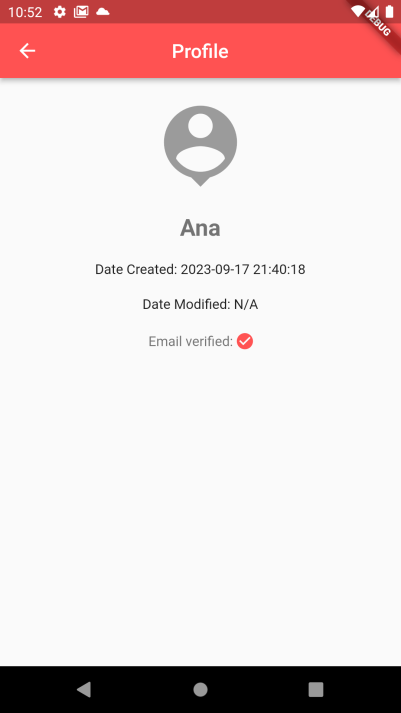
*Слика 20,21*

## Верификација на е-маил

Доколку мејлот не е верификуван постои опција за негова верификација. Со клик на опцијата Verify email во Profile погледот, се испраќа мејл на внесениот мејл на корисникот (Слика 22) и при клик на линкот кој се наоѓа во мејлот се добива порака за направена успешна верификација (Слика 23). Откако ќе се изврши оваа акција, во делот ‘Profile’ се прикажува дека мејлот е верификуван (Слика 24).



*Слика 22,23*



*Слика 24*