



**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ –
СОФИЯ**

ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

КУРСОВ ПРОЕКТ

Дисциплина: „Програмиране за мобилни устройства”

Тема: „Български турист“

Изготвил:

Мартин Стоименов (121221049), Група: 44, III курс, КСИ

Ръководител:

Доц. д-р Невен Николов

Съдържание

1. Увод	4
2. Анализ на съществуващи разработки	4
100 places	4
Top 100 sites of Bulgaria	5
Опознай.bg	6
Yettel - туристическа книжка	6
Изводи	7
3. Проектиране	7
Функционални изисквания	7
Нефункционални изисквания	8
Обхват на приложението	8
Use Case диаграма	9
Диаграма на компонентите	9
Съхранение на данните	10
Отдалечена база данни Firebase	10
Локална база данни SQL Lite	10
Използвана технология	11
Използвани библиотеки	11
Структура на проекта	12
Бизнес-логика (java code)	12
UI	21
Входни данни	22
4. Реализация	22
Използване на горно и долно меню	22
Използване на случайна снимка	24
Използване на картата	24
Сензор за пръстов отпечатък	25
Адаптери	25
Callback интерфейси	25
Връзка с базата данни	25
Модели	26
NTO100	26

Place.....	26
User.....	27
5. Потребителски интерфейс и ръководство	27
Вход	27
Регистрация.....	29
Меню „Помощ“	29
Начална страница	30
Страница „Моите места“	31
Добави място	32
Детайли	32
100 НТО.....	33
Детайли на национален туристически обект	33
Профил	34
В близост.....	35
6. Бъдещо развитие на проекта.....	35
В близост.....	35
Функция “запомни ме”	35
Автоматично вземане на информация от гугъл.....	35
Посещаване на мястото посредством снимка.....	36
Потребителска обратна връзка.....	36
Приложение за iOS.....	36
7. Заключение.....	36
8. Използвана литература.....	37

1. Увод

Всеки българин, който обича страната си, иска да обиколи най-красивите кътчета в нея. Планирането, обаче, понякога е доста трудно, когато искаме да обиколим много дестинации в един ден. Трябва да мислим кое място след кое да бъде, за да не пропуснем някоя спираща дъха дестинация.

С приложението "Български турист" потребителят може да въведе всички свои дестинации, които иска да посети, както и да добавя в „любими места“ тези, които би посетил най-скоро.

Освен свои дестинации, потребителят може да вижда и списък със 100-те национални туристически обекта и да добавя всеки един от тях в своя списък за посещение.

А в допълнение, приложението има и създателен характер между потребителите, като всеки посетен туристически обект носи по 5 точки, а всеки друг – по 2 точки, като по този начин се насърчава посещаването на повече места!

Функцията за пръстов отпечатък предразполага към по лесно и бързо верифициране на потребителя, за да не губи време с въвеждането на имейл и парола, а да се наслади на своето преживяване с едни от най-красивите места в България!

2. Анализ на съществуващи разработки

100 places

Какво представлява?

Представлява електронна туристическа книжка за 100-те национални туристически обекта, с възможност за печати.

Предимства:

Има възможност за слагане на електронен печат за всеки обект, проверявайки локацията.

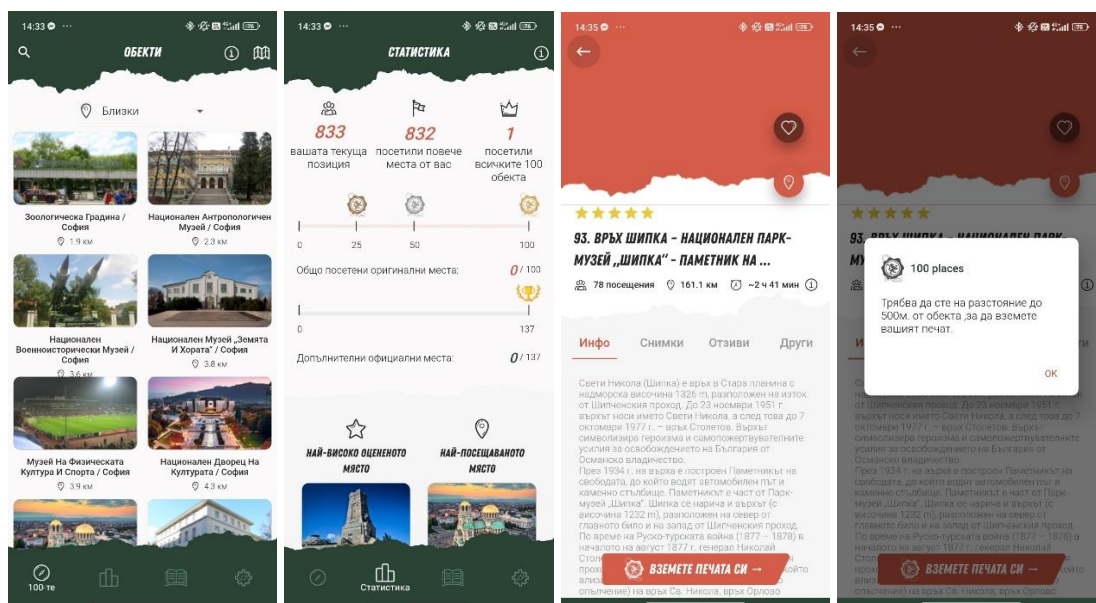
Открива близките до текущата локация национални обекти.

Има създателен характер между потребителите.

Недостатъци:

За разлика от моето приложение, обаче, може да се използва само за 100-те национални обекта и няма възможност за планиране на обекти, които потребителят желае.

Няма възможност да се пази информацията за вход с пръстов отпечатък.



Top 100 sites of Bulgaria

Какво представлява?

Представя информация за всеки от 100-те национални туристически обекта.

Предимства:

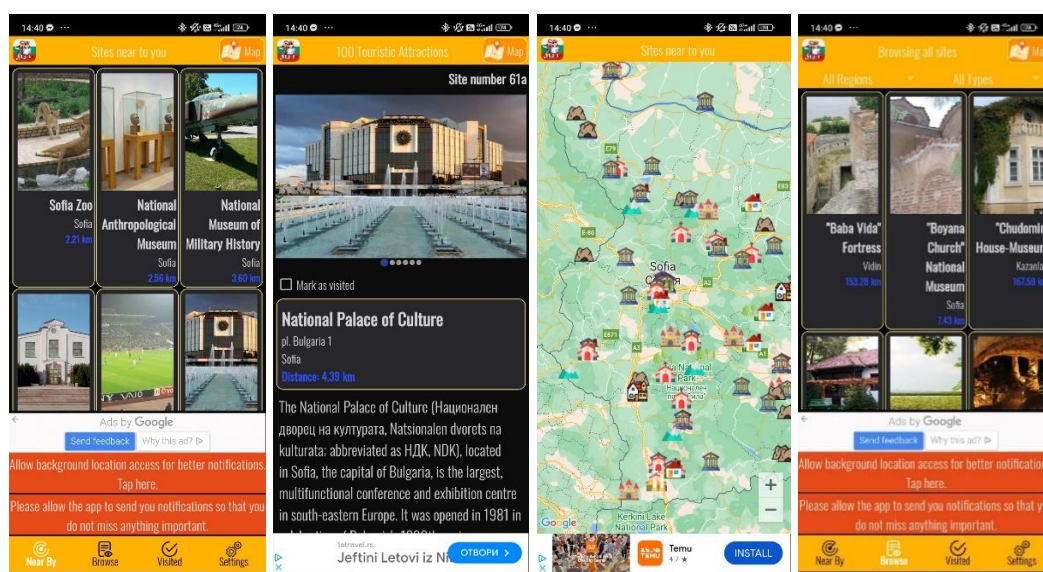
Предоставя подробна информация за обекта, като разстояние, история, работно време, цени и т.н.

Недостатъци:

Няма възможност да се добавят и планират други дестинации.

Няма възможност да се слагат електронни печати при посещение на обектите.

Няма възможност да се пази информацията за вход с пръстов отпечатък.



Опознай.bg

Какво представлява?

Представя информация за около 10000 дестинации в България, разделени в различни категории като култура, природа и други.

Предимства:

Предоставя подробна информация за почти всички обекти на територията на България, не само за 100-те национални туристически обекта.

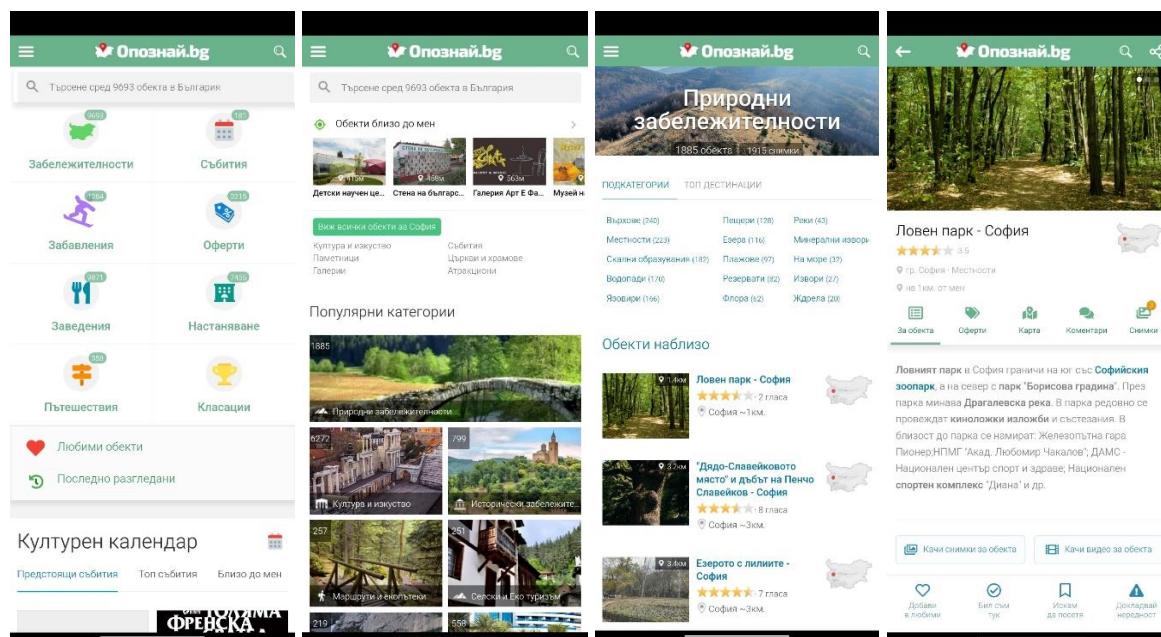
Недостатъци:

Не може потребителят сам да си добавя и планира дестинации.

Не могат да се слагат печати и да се отбелязват местата като посетени.

Няма възможност да се пази информацията за вход с пръстов отпечатък.

Няма създателен елемент.



Yettel - туристическа книжка

Какво представлява?

Телекомуникационният оператор Yettel имат в своето приложение възможност за персонална туристическа книжка със 100-те национални туристически обекта, дори за лица, които не са техни клиенти. Приложението предоставя възможност за персонално посещаване на обектите и слагането на дигитален печат, посредством QR код.

Предимства:

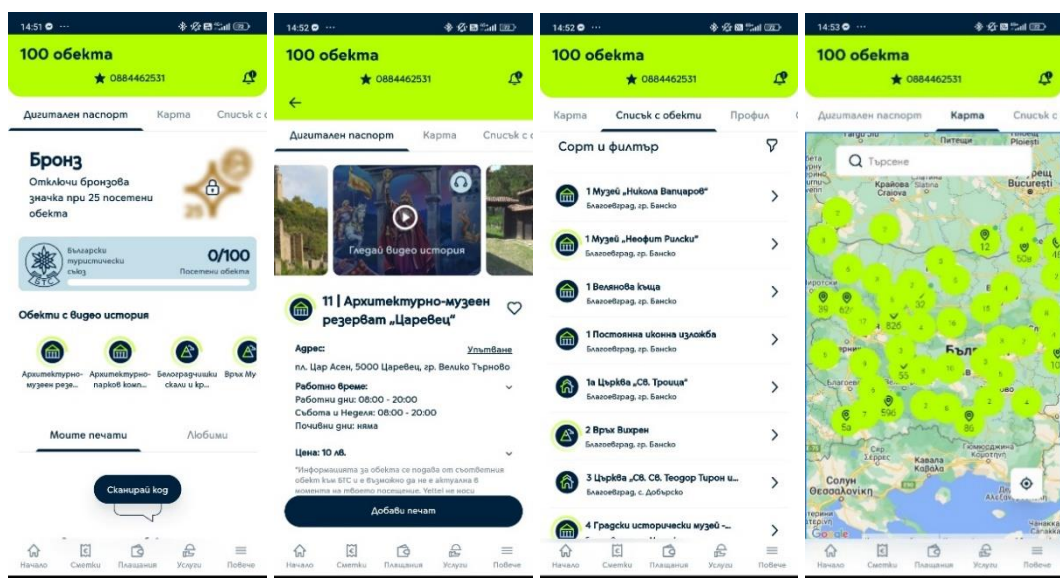
За всеки туристически обект е възможно да се слага печат посредством QR код.

В приложението на yettel за някои обекти има и видео история.

Недостатъци:

Не позволява добавянето и планирането на лични пътешествия.

Няма възможност да се пази информацията за вход с пръстов отпечатък.



Изводи

Лично аз и мои познати използваме приложението на yettel за печати за 100-те национални туристически обекта и след кратко проучване смятам, че то е най-използваното от изброените, но липсата на възможност за списък с лични дестинации лишава потребителите от лесно планиране на разходките.

След него бих наредил опознай.bg, който се използва от много хора за предварително проучване на маршрути до даден обект и информация за него.

3. Проектиране

Функционални изисквания

№	Изискване	Приоритет
1	Приложението трябва да може да позволява регистрацията, редактиране и изтриване на потребителски профили.	висок
2	Приложението да позволява вход с потребителски профил, използвайки входни данни или пръстов отпечатък.	висок
3	Приложението да има създателен характер, като известява потребителят за неговия резултат: посетени обекти, точки, ниво, класация спрямо останалите потребители.	среден

4	Приложението да разполага с помощно меню, демонстриращо работата му.	нисък
5	Приложението да показва списък със 100-те национални туристически обекта.	висок
6	Всеки туристически обект да може да се отваря в детайли и да позволява добавянето му в личния списък на потребителя.	висок
7	Приложението да поддържа личен списък с дестинации за посещение на потребителя, като в него могат да се добавят и изтриват записи.	висок
8	Приложението да разполага с функция „в близост“, като известява потребителите за места от списъка, в близост до локацията на тяхното устройство.	среден
9	Приложението да позволява добавянето на места от личния списък с дестинации в списък с любими.	нисък

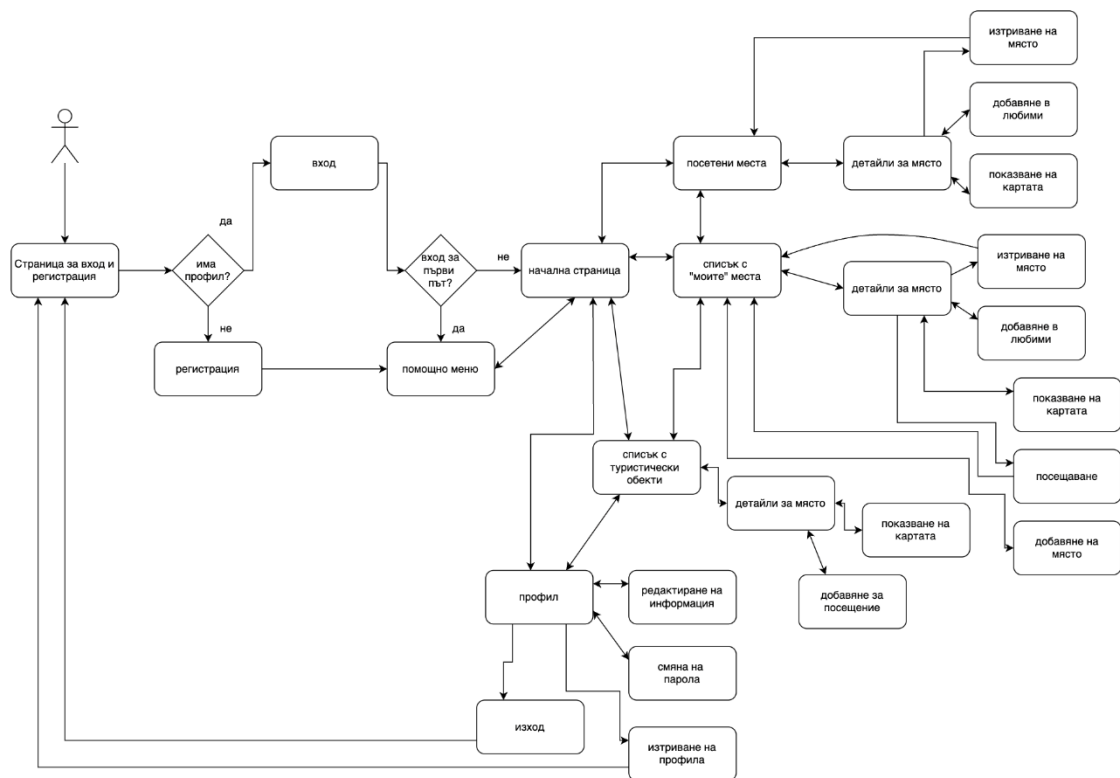
Нефункционални изисквания

№	Изискване	Приоритет
1	Приложението използва топли, светли цветове, с цел задържане вниманието на потребителя.	висок
2	Темата на приложението зависи от темата на устройството – тъмна или светла.	среден
3	Приложението е предвидено за български граждани, така че трябва да съдържа само текст на български език, за по-лесната му употреба.	висок
4	Времето за необходимите изчисления в приложението да бъде не повече от 20 секунди (пример: при изчисляване на разстоянието до дадена дестинация)	висок
5	Приложението да е достъпно за устройсва с Android OS	среден

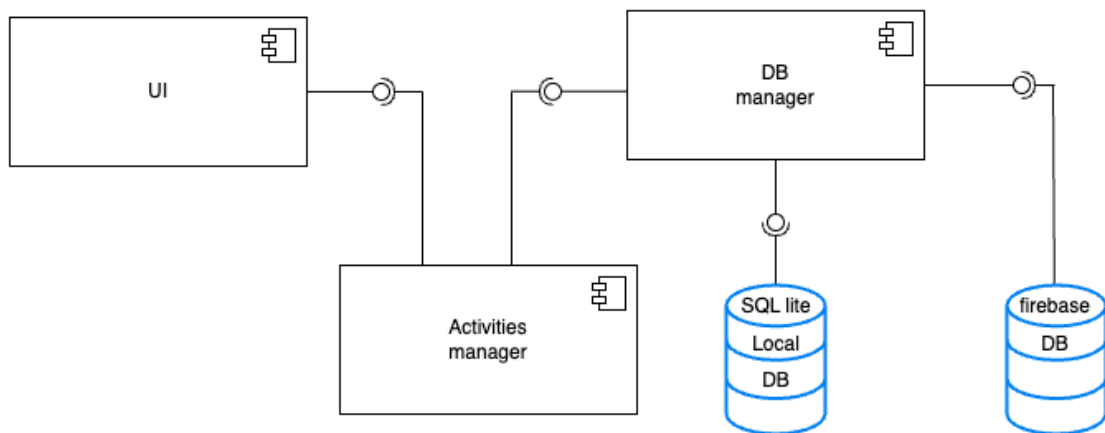
Обхват на приложението

Приложението „Български турист“ е предвидено изцяло за български или чуждестранни граждани, пребиваващи на територията на република България. Потенциалните потребители са всички, които искат да посетят едни от най-красивите места в нашата страна. Приложението е изцяло на български език, за да е удобно за всякакви възрастови групи и за да се съхрани един от най-големите символи на България.

Use Case guazpama



Диаграма на компонентите

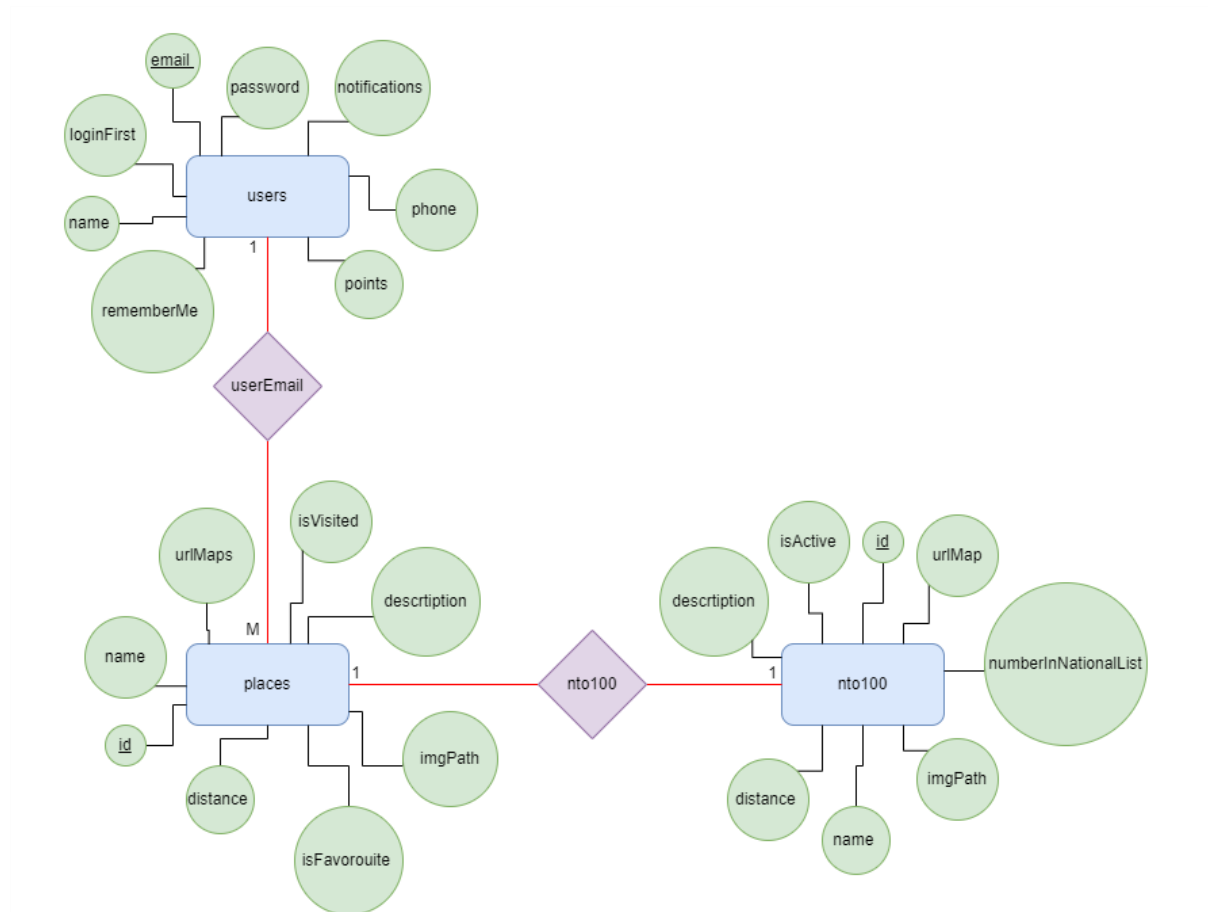


Съхранение на данните

Отдалечена база данни *Firestore*

Приложението използва отдалечена база данни *Firestore*, за да съхранява информацията за потребители и техните дестинации. В нея се съхранява и списъкът със 100-те национални туристически обекта.

Базата данни съдържа три основни таблици (колекции): *users*, *places*, *nto100*. Ето и подробна E-R диаграма:



Всеки потребител може да има много на брой лични дестинации, като всяка от тях може да бъде туристически обект, което налага и връзката между *places* и *nto100*. Връзката между тях е 1 към 1, тъй като веднъж добавен национален обект в списъка на даден потребител, той не може да се добавя повече, тъй като вече съществува там.

Локална база данни *SQLite*

Приложението разполага и с локална база данни *SQLite*, за да съхранява входните данни на потребителите, желаещи да използват функцията за вход с пръстов отпечатък. Тя съдържа една единствена таблица с две полета за имейл и парола.

```
@Override public void onCreate(SQLiteDatabase db) {  
    String CREATE_EMAIL_TABLE = "CREATE TABLE " + TABLE_EMAILS
```

```
+ "(" + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
+ KEY_PH_NO + " TEXT" + ")";
    db.execSQL(CREATE_EMAIL_TABLE);
}
```

Използвана технология

Приложението е разработено на Android Studio, поради лесната му употреба и съществуващите инструменти, които улесняват създаването и тестването на приложението. Използва java за описание на бизнес логиката и xml за визуалната част(UI).

Приложението е предвидено за android устройства.

Използвани библиотеки

- com.google.android.gms.location, android.location – за използване на местоположението на устройството и на картата

-android.gms.maps – за използване на картата.

- com.google.android.material – създаване на меню с елементи (items) за навигация между страниците.

- com.google.firebase.firestore – за връзка с отдалечена база данни Firebase на Google.

-java.net, android.net – за изпълняване на URL-и и обръщания към външни системи.

androidx.biometric – за използване на биометричните данни на устройството, като пръстов отпечатък.

bumptech.glide.Glide – за зареждане на снимка от URL.

com.google.android.gms.tasks – асинхронно изпълнение на задачи.

android.database.sqlite – локална база данни на SQL Lite.

java.security – приложението го използва за хеширане на паролата по алгоритъм SHA-256.

androidx.cardview.widget – позволява групирането на няколко елемента в едно цяло и използването на темата на устройството – тъмна или светла.

Структура на проекта

Бизнес-логика (java code)

Activities package

Този пакет съдържа всички Страници(Активитита), управляващи layout-тите на UI.

- MainActivity (activity_main layout) – логин страницата, верификацията с пръстов отпечатък и запазването на имейлите в локалната база данни при разрешаване на функцията „Вход с пръстов отпечатък“ от страна на потребителя. Използвани библиотеки: androidx.biometric.BiometricPrompt, bumptechnology.glide.Glide, com.google.firebase.firestore, java.net

- getPrompt – взимат се биометричните данни на потребителя от устройството. Ако удостоверяването е правилно, се взимат всички имейли от локалната база данни SQL Lite и ако те са повече от един, управлението се предава на метода chooseEmailToLogin за избиране на входни данни. Ако входните данни запазени в локалната база данни са само за един потребител (само за един потребител е активна функцията „Вход с пръстов отпечатък“ на това устройство), директно се преминава към тяхната проверка и валидация с помощта на метода checkForExistingUser.
- chooseEmailToLogin – при разрешена функция „Вход с пръстов отпечатък“ за повече от един профил на едно устройство, се предизвиква изскачащ прозорец rorup_emails, който има списъчен компонент в себе си. На този компонент се зарежда списъкът с данните, взети от локалната база данни за вход. При кратко натискане върху някой от елементите на списъка, се прави проверка за съществуващ потребител с избраните входни данни, посредством метода checkForExistingUser. При по-дълго задържане върху даден елемент от списъка с входните данни, се появява нов изскачащ прозорец, който пита потребителя дали иска да премахне входните данни за избрания имейл адрес. В случай на положителен отговор, входните данни за избрания имейл се изтриват от текущия списък и от локалната база данни.
- notifyUser – показва изскачащо TOCT съобщение с подадения като параметър текст.
- checkForExistingUser – проверка в отдалечената база данни Firebase дали данните за вход са валидни и дали потребител с такива данни съществува, посредством метода checkForExistingUser от помощния клас QueryLocator. Ако намери съществуващ потребител, прави проверка дали въведените входни данни са запазени в локалната база данни. Ако не съществуват локално, потребителят се подканя да ги добави. Ако пък вече са добавени, се прави проверка дали потребителят влиза за пръв път в своя профил. Това става ясно от полето loginFirst на user записа, което се променя на false, след първото влизане.
- activateFingerPrintForEmail – при въвеждане на данните за вход и изключена функция за „Вход с пръстов отпечатък“ потребителят се подканва да включи функцията за дадения профил, посредством изскачащ прозорец. Ако потребителят избере да добави входните данни, те се добавят в локалната база данни SQL Lite.

- `savedEmailExists` – проверява дали за даден профил вече е активна функцията за „вход с пръстов отпечатък“ – дали входните данни съществуват в локалната база данни.
- `navigateToHomeActivity` – навигиране към активитите за начална страница.
- `showPassword` – показва паролата с реално въведените символи.
- `redirectToHelp` – отваря активитите съдържащо страницата „Помощ“.
- `checkIsFirstLogin` – проверява дали потребителят се логва за пръв път в приложението и се препраща към менюто „Помощ“ ако това е така. В противен случай се редиректва към началната страница.
- `TextWatcher` – следи за промяна на текста в текстовите полета, за да блокира или позволи бутона за „Вход“.
- `updateButtonState` – разрешава и забранява бутона за „Вход“, когато не са въведени коректни входни данни или текстовите полета са празни.
- `setImageRandomly` – задава снимката на логин страницата на случаен принцип, търсейки в Google по следния низ: „български обекти“.
- `setupKeyListener`, `moveViewsUp` – при отваряне на клавиатурата, текстовите полета, заедно с бутоните се качват нагоре и покриват снимката, за да може потребителят да вижда необходимата информация за вход.
- `resetViewsPosition` – връща позицията на полетата и бутоните обратно на мястото им и показва снимката.

- `SignUpView(activity_sign_up layout)` – потребителят бива подканен да въведе данните си за регистрация в системата. „Подслушва се“ промяната на текстовите полета и се прави проверка на данните с регулярни изрази. Прави се проверка и за съществуващи имейл адрес или телефон в базата данни. Използвани библиотеки:

`com.google.android.gms.tasks`, `com.google.firebase.firestore`

- `checkForExistingEmail`, `checkForExistingPhoneNumber` – извикват се при натискане на бутона за потвърждаване на регистрацията. От отдалечената база данни се извличат записи с въведения имейл адрес или телефон. Ако такива съществуват, потребителят бива предупреден, че не може да създаде профил с вече съществуващи данни в системата.
- `navigateToHelpActivity` – веднага след регистрацията потребителят бива препратен към менюто за „Помощ“, за да се запознае със структурата и работата на приложението.
- `updateSignUpButtonState` – разрешава бутона, в зависимост от въведената информация. Следи се дали всички полета са попълнени и дали спазват шаблона зададен с регулярни изрази във `verifyFields` метод.
- `verifyFields` - проверява синтаксиса на въведената информация. Прави проверка и дали паролата и нейното потвърждение са еднакви.

- `CodeVerificationActivity (activity_code_verification layout)` – този клас беше предвиден да изпраща верификационен код на въведения имейл адрес при регистрация, чрез който потребителят да го потвърждава. В процеса на работа се оказа, че да се изпращат имейли е необходим имейл сървър. Опитах се да използвам вече съществуващи, но системите ме отхвърлиха.

- generateRandomCode – генерира 6-цифрен код на случаен принцип
- sendEmail – изпраща код по имейл адрес
- generateHtmlBody – генерира тялото на имейла

-HomeActivity (activity_home layout) – представлява бизнес логиката на началната страница, където потребителят вижда своите точки, посетени места, ниво и класация. Ако не е дадено разрешение за достъп на приложението до локацията, се прави обръщение към потребителят да го позволи. Използвани библиотеки: android.material.

- onRequestPermissionsResult – прави обръщение към потребителят да разреши достъп на приложението до локацията на устройството.
- onActivityResult – когато управлението се върне от списъка с посетени дестинации обратно към началната страница, този метод прекратява progress dialog-a.
- displayPointsAndLevel – променя текста на текстовите полета с новоизчислените данни.
- getLinkedVisitedPlaces – използва метода getVisitedPlaces от помощния клас QueryLocator, за да извлече всички посетени от текущия потребител дестинации. Методът се извиква при отваряне на страницата и служи за обновяване на информацията. В зависимост от съдържанието на списъка с посетени обекти се изчисляват точките и нивото на потребителя и опреснената информация се запазва в отдалечената база данни Firebase.
- updatePointsForUser – обновява информацията за точките в отдалечената база данни.
- calculatePoints – обхожда списъка с посетени места на текущия потребител и в зависимост от това, дали мястото има свързан към себе си запис на национален обект или не, прибавя към общия брой точки съответно 2 или 5.
- calculateLevel – дели целочислено броя точки на 100 и прибавя 1, за да изчисли нивото. Условието е на всеки 100 точки да се вдига ниво.
- displayCountOfPlaces – обновява текста на текстовото поле textViewPlacesCount, което визуализира броя посетени места.
- fetchTopUsers – извлича всички потребители, сортирани по брой точки и използва метода populateTopUsers, за да визуализира първите трима на началната страница.
- populateTopUsers – визуализира списъка с тримата потребители с най-висок брой точки. Итерираща върху списъка с потребители, сортирани по брой точки, за да установи позицията на текущия потребител.

- HelpActivity (activity_help2 layout) – това е менюто „Помощ“. Отваря се при първо влизане, след регистрация на потребител. Също така се отваря от горното навигационно меню, чрез въпросителната икона.

- onBackPressed – забранява връщането назад от текущото активити. Ако се натисне стрелката „Назад“ на устройството, потребителят ще излезне от приложението. Единственият начин да се затвори менюто е да се мине през всички слайдове.
- updateContent – сменя текста в текстовото поле за информация и снимката на слайда. По този начин се използва едно view, на което му се променя съдържанието.
- onNextButtonClick – предизвиква се при натискане на бутона „Напред“, променя индекса на страницата нагоре и обновява текста и снимката посредством метода

updateContent. Управлява текста на бутоните, ако е последна страница от менюто препраща обратно към началната страница на приложението.

- onBackButtonClick – предизвиква се при натискане на бутона „Назад“, променя индекса на страницата надолу и обновява текста и снимката посредством метода updateContent. Ако е първа страница от менюто, бутона „Назад“ се скрива.

- PlaceListView (activity_place_list_view layout) – използва се за всички списъци с места в приложението, в зависимост от случая на употреба: случай 1: визуализира „моите непосетени места“ на текущия потребител, извлечени от базата данни; случай 2: показва всички 100 национални туристически обекта, извлечени от базата данни; случай 3: показва всички посетени места на текущия потребител, извлечени от базата данни. Бъдещото развитие на приложението предвижда и случай 4, в който списъкът ще се използва, за да се изброят всички места в близост до устройството на дадения потребител. Използвани библиотеки: com.google.android.material

- listMyPlaces – извлича всички непосетени места на текущия потребител от отдалечената база данни Firebase, посредством метода getMyUnvisitedPlaces от помощния клас QueryLocator и използва PlaceAdapter класа, за да отдели всеки един елемент на списъка като отделна единица. Селектира „моите места“ в навигационното меню.
- listNto100 - извлича всички национални туристически обекти от отдалечената база данни Firebase, посредством метода getNto100 от помощния клас QueryLocator и използва NTOAdapter класа, за да отдели всеки един елемент на списъка като отделна единица. При избиране на един обект, се препраща към активитито PlaceView, където се отваря избраното място в детайли. Селектира „100 НТО“ в навигационното меню.
- listVisitedPlaces - извлича всички посетени места на текущия потребител от отдалечената база данни Firebase, посредством метода getVisitedPlaces от помощния клас QueryLocator и използва PlaceAdapter класа, за да отдели всеки един елемент на списъка като отделна единица. Избира началната страница като селектиран елемент на навигационното меню.
- sortPlaces – бъдещата реализация и развитие на приложението предвижда възможността за сортиране на списъците с дестинации.
- navigationMenu – в зависимост от случая, селектира дадения елемент в навигационното меню.

- PlaceView (activity_place_view) - използва се за за детайли на всяко място в приложението, в зависимост от случая на употреба: случай 1: визуализира детайли на „моите непосетени места“ на текущия потребител, извлечени от базата данни; случай 2: показва детайли на избран национален туристически обект, извлечен от базата данни; случай 3: показва детайли на посетено място на текущия потребител, извлечено от базата данни. Бъдещото развитие на приложението предвижда и случай 4, в който компонента ще се използва, за да се изобразят детайлите на дестинация в близост до устройството на дадения потребител. Използвани библиотеки: com.google.android.material

- `onOptionsItemSelected` – прихваща връщането със стрелка назад към списъка, за да прекрати `progress dialog`-а в списъка, активира се при отваряне на детайлите на мястото.
- `showMyPlace` – активира се при случай 1, извлича избраното място по ID от отдалечената база данни и задава стойност на текстовите полета с извлечената информация. Имплементира се логиката за всички бутони на `View`-то, като изтриване на място, посещаване, отваряне в картите, добавяне в любими.
- `deletePlace` – стартира изтриването на дестинацията. Препраща към метода `acceptDeleting`.
- `acceptDeleting` – визуализира се изскачащ прозорец, който пита потребителя за решението му да изтрие дестинацията. При положителен отговор, мястото се изтрива от отдалечената база данни посредством метода `deletePlace` от помощния клас `QueryLocator` и потребителят се препраща към страницата с непосетените му места.
- `changeFavourite` – при натискане на бутона с формата на сърце, се променя `isFavourite` полето на даденото място, обновява се в отдалечената база данни и се променя иконата за добавяне в любими.
- `setHeartImg` – променя състоянието на иконата на бутона за добавяне на място в любими, като следи предишното и състояние: ако е била запълнена, маха запълването, ако не е била – я запълва.
- `show100ontos` – активира се при случай 2, извлича избраното туристическо място по ID от отдалечената база данни и задава стойност на текстовите полета с извлечената информация. Имплементира се логиката за всички бутони на `View`-то, като добавяне за посещение, отваряне в картите. Прави се проверка дали мястото вече не е добавено за посещение в списъка с местата на дадения потребител и ако е така, се скрива този бутон.
- `formatDistance` – преобразува целочислената стойност на изчисленото разстояние в текстов низ.
- `refreshDistance` – предупреждава потребителя да си включи локацията, ако не го е направил. Задава таймаут 20 секунди, за да изчисли разстоянието от текущото местоположение до дестинацията, която е отворена. След изтичането му, калкулирането се прекратява и дистанцията не се оповестява на потребителя. Използва `calculateDistance` метод от помощния клас `Helper`, за да изчисли разстоянието до избраната дестинация, в зависимост от типа на обекта: дали е национален или потребителски.
- `navigationMenu` – в зависимост от случая селектира дадения елемент в навигационното меню.
- `visitPlace` – променя статуса `isVisited` на избраната дестинация на `true`, ако между потребителя и мястото, разстоянието е по малко от 500 метра. Извършва се опресняване на данните в отдалечената база данни.

-`AddPlaceActivity` (`activity_add_place` layout) – Текущият потребител добавя нова дестинация в списъка си за посещения. Използвани библиотеки: `android.gms.location`, `android.gms.maps`, `android.material`, `firebase.firestore`, `java.net`

- `onMapReady` – Зарежда картата на страницата. Ако местоположението на устройството е включено, картата се позиционира на текущото местоположение, в

противен случай са зададени координати да се позиционира в центъра на София. Този метод задава и „слушател на събития“ (listener), за да отрази, когато потребителят маркира точка на картата.

- `onMapClick` – слага маркер на избраната точка на картата и взема линка за тази точка.
- `getImage` – Извиква се при натискане на бутона „Добави“. Търси на случаен принцип снимка от гугъл, базирана на името на добавеното място. Извършва се HTTP обръщение към външна система. Върнатият URL за снимка се използва за новосъздадената дестинация.
- `addNewPlace` – Извлича списъка със 100-те национални туристически обекта от базата данни и проверява по име, дали новосъздадената дестинация съществува в този списък. Ако съществува, значи мястото е национален обект и се прави връзка между новосъздадения обект, и съществуващия. След което се извиква `addPlaceToMyPlaces` от помощния клас `QueryLocator`, за да се добави новото място в базата данни. Използва се референцията от базата данни за текущия потребител, за да се свърже новосъздаденото място с него.
- `navigateToNewPlace` – след като мястото е създадено успешно, потребителят се препраща към списъка с всичките му обекти.
- `onOptionsItemSelected` – когато потребителят се върне назад, без да е създал записа, трябва да се прекрати изпълнението на `progress dialog`-а, който е активиран преди зареждане на текущата страница.

-`ProfileActivity(activity_profile layout)` – Отговаря за управлението на потребителския профил. Имплементира логиката зад бутоните за редактиране на име и телефон, смяна на паролата, изтриване на профила и изход. Използвани библиотеки: `com.google.android.material`.

- `getUserInfo` – извлича информацията за текущия потребител, с помощта на метода `getUserByEmail` от помощния клас `QueryLocator`.

-`ChangePasswordActivity (activity_change_password layout)` – описва логиката на страницата за смяна на потребителската парола. Използвани библиотеки: `android.material`,

- `checkForExistingUser` – Извиква се при натискане на бутона за потвърждаване на новата парола. Използва метода `checkForExistingUser` от помощния клас `QueryLocator`, за да извлече всички потребители с имейла на текущия потребител и старата му парола. Ако потребителят не съществува, значи е въвел грешна парола. Ако потребителят съществува, новата парола се хешира и се извиква универсалният метод `updateUserSingleField` от помощния клас `QueryLocator`, за да се обнови паролата в отдалечената база данни. Освен в отдалечената база данни, паролата се обновява и в локалната база `SQL Lite`, където е запазена информацията за вход на това устройство. След това потребителят се препраща към страницата „Профил“.
- `updateAcceptButtonState` – извиква се при промяна на текста в текстовите полета. Отговаря за блокирането и разблокирането на бутона за потвърждаване на новата парола. За да е наличен бутон, всички полета трябва да са попълнени и да отговарят на регулярните изрази.

- `verifyNewPassword` – проверява дали нововъведената парола отговаря на регулярен израз, задължаващ потребителя да въведе поне 8 символа, съдържащи големи, малки букви и цифри. Сравнява също дали новата парола и потвърждението ѝ са еднакви. Ако паролата не отговаря на някои от тези изисквания, до всяко текстово поле се показва грешка.

-`DeleteProfileActivity` (`activity_delete_profile_layout`) – Страница за изтриване на текущия профил. Потребителят е необходимо да въведе своята парола и да се съгласи, че е разбрал предупреждението за изтриване. Зарежда се от страницата „Профил“. Използвани библиотеки: `android.material`

- `checkForExistingUser` – Използва метода `checkForExistingUser` от помощния клас `QueryLocator`, за да извлече всички потребители с текущия имейл адрес и въведената парола. Ако потребителят не съществува, се показва грешка на текстовото поле за въвеждане на паролата. Ако потребителят съществува и е приел предупреждението, профилът се изтрива от отдалечената база данни `Firebase`, заедно с всичките свързани към него места за посещение. Изтриват се и потребителските данни за вход, запазени на текущото устройство. След успешното изтриване, потребителят се препраща към страницата за вход.

- `NearestActivity` (`activity_nearest_layout`) – бъдещото развитие на проекта предстои да съдържа логика за откриване на местата в близост до локацията на устройството и да ги показва списъчно. Към този момент съдържа единствено снимка и текст, гласящ „Очаквайте скоро!“.

Adapters package

Съдържа адаптерите за представяне на елементите на списъците с места като отделни елементи.

-`NTOAdapter` (`list_item_place_layout`) – преобразува всеки елемент на списъка със 100-те национални туристически обекта в отделна единица и задава стойностите на полетата в отделната единица.

-`PlaceAdapter` (`list_item_place_layout`) - преобразува всеки елемент на списъка с местата за посещение на текущия потребител в отделна единица и задава стойностите на полетата в отделната единица, като прави проверка в базата данни за `isFavourite` и `nto100`, за да зададе правилно и техните стойности за всяко място. Имплементира логиката за добавяне в любими, като обновява иконата и информацията в отдалечената база данни.

Callbacks package

Използва се за връщане на резултат от вложен клас, например при извличане на данни от базата данни или от другаде

- `LocationCallback` – връща текущото местоположение. Използва се в метода `getCurrentLocation` в помощния клас `Helper`.

- NTO100Callback – връща списък с националните обекти. Използва се в метода getNto100 в помощния клас QueryLocator.
- PlacesCallback – връща списък с обектите на даден потребител. Използва се в getAllMyPlaces, getMyUnvisitedPlaces и getVisitedPlaces в помощния клас QueryLocator.
- SingleNTO100Callback – връща единичен запис на национален обект. Използва се в getNTO100ById в помощния клас QueryLocator.
- SinglePlaceCallback – връща единичен запис на дестинация на даден потребител. Използва се в getMyPlaceById в помощния клас QueryLocator.
- SingleUserCallback – връща единичен запис на потребител. Използва се в getUserByEmail и checkForExistingUser в помощния клас QueryLocator.

DataBase package

Описва използването на отдалечена база данни Firebase и на локална база данни SQL Lite. Използвани библиотеки: android.database.sqlite.

- LocalDatabase – описва таблицата за съхранение на входни данни, локално на устройството.

- onCreate – създава се таблицата TABLE_EMAILS, която да съхранява записите на входни данни .
- addEmail – добавяне на нов потребител в локалната база данни.
- updatePassword – обновяване на паролата на съществуващи входни данни, по подаден имейл адрес.
- getAllEmails – връща списък с всички имейли от локалната таблица.
- deleteEmail – изтрива входните данни по подаден имейл адрес.
- getHashedPassword – взима паролата на входните данни по подаден имейл адрес.

-QueryLocator – Служи за връзка с отдалечена база данни Firebase на Google. Съдържа основните CRUD заявки, използвани в приложението. Използвани библиотеки: com.google.android.gms.tasks, com.google.firebase.firestore.

- getUserByEmail – търси потребител в базата данни по подаден имейл адрес.
- checkForExistingUser проверява за съществуващ потребител по подадени имейл адрес и парола.
- updateUserSingleField – абстрактен метод, използван за обновяване на полетата на потребителя, по подадена текстова стойност.
- updateUserSingleField - абстрактен метод, използван за обновяване на полетата на потребителя, по подадена булева стойност.
- getAllMyPlaces – връща всички места на текущия потребител – посетени и непосетени.
- getMyUnvisitedPlaces – връща непосетените места на даден потребител,
- updateFavouriteStatus – обновява полето isFavourite на подадено място.
- getNto100 – връща списъка с всички активни национални обекти.
- getUserRef – връща референция към потребителски запис.

- `getPlaceRef` – връща референция към запис на дестинация.
- `getNTORef` – връща референция към запис на национален обект.
- `getVisitedPlaces` – връща всички посетени места на дадения потребител.
- `getNTO100ById` – връща национален обект, по подадено ID.
- `getMyPlaceById` – връща дестинация по подадено ID.
- `addANewPlace` – добавя нов запис на дестинация, създаден от национален обект.
- `registerUser` – добавя нов потребител в базата данни.
- `removeIsFirstLoginStatus` – премахва флага за първо влизане на даден потребител.
- `addPlaceToMyPlaces` – добавя нов запис на дестинация.
- `updatePlaceDistance` – обновява разстоянието до дадено място в базата данни.
- `updatePlaceVisitation` – обновява полето `isVisited` за дадено място.
- `deleteUser` – изтрива потребителски профил от системата.
- `deleteAllLinkedPlaces` – изтрива всички свързани записи с референция към даден потребител, по подаден имейл. Итерира върху списък с места и ги изтрива едно по едно посредством метода `deletePlace`.
- `deletePlace` – изтрива място по подадено ID.

Helpers package

Този пакет съдържа помощните класове в приложението.

-Helper – съдържа помощни методи, използвани в страниците на приложението.

Използвани библиотеки: `android.location`, `android.net`, `com.google.android.gms.location`, `com.google.firebase.firestore`.

- `checkIsNTO` – проверка дали дадено място е свързано към национален туристически обект.
- `createPlaceFromNTO` – създава място от национален обект, свързано към потребител.
- `showOnMap` – построява се URL, който се отваря в уеб браузър. Ако устройството има приложението Карти, уеб браузъра препраща към него.
- `isLocationEnabled` – проверява дали локацията на устройството е включена.
- `calculateDistance` – изчислява разстоянието между текущото местоположение и подаденото място.
- `getCurrentLocation` – взема текущото местоположение на устройството и го връща посредством `LocationCallback`.

- Navigation – имплементира логиката за пренасочване между страниците от навигационните менюта.

- `bottomNavigation` – прави проверка кой елемент от долното меню е избран и пренасочва към съответната страница, посредством методи.
- `topMenu` – прави проверка кой елемент от горното меню е избран и пренасочва към съответната страница.
- `navigateToHelp` – навигира към менюто „Помощ“.
- `navigateToHomeActivity` – навигира към началната страница.

- `navigateToPlaceListView` – навигира към списъка с места, като се подава за кой случай става въпрос. Както по-нагоре се споменава, случаят определя дали ще се показват „моите непосетени места“ на текущия потребител, 100-те национални туристически обекта, или „моите посетени места“ на текущия потребител.
- `navigateToProfileActivity` – навигира към страницата с информация за профила.
- `navigateToNearestActivity` – навигира към списъка с места в близост.

- `PasswordHasher` – помощен клас с един единствен метод да хешира подаден низ. Използва се за хеширане на паролата. Използвани библиотеки: `java.security`.

Models package

Съдържа моделите, използвани в приложението и отдалечената база данни:

- `NTO100` – модел за националните туристически обекти.
- `Place` – модел за потребителските места за посещение.
- `User` – модел за потребителските профили.

UI

drawable resources

Съдържа основните ресурси, използвани в приложението: снимки, векторни изображения, шаблонни фигури и т.н.

- `bulgarian_flag` – трибагреникът на България. Използва се, за да идентифицира всички национални обекти.
- `error_image` – Снимка, която се показва при невъзможност за зареждане на оригиналните снимки.
- `flag_black_border` – Отново българското знаме, но с черна рамка около него, за да се отличава белия цвят, при използване върху бял фон.
- `ic_favourite_border` – иконата „незапълнено сърце“. Използва се, когато дадено място не е добавено в любими.
- `ic_favourite_filled` – иконата „запълнено сърце“. Използва се, когато дадено място е добавено в любими.
- `placeholder_image` – използва се докато зареди оригиналната снимка.
- `rounded_buttons` – определя фигурата на използваните в приложението оранжеви, зелени, червени, светлозелени и жълти бутони.
- `sand_clock` – снимка, използвана за места в близост, за да информира потребителя за скорошното пускане на новата функционалност.
- `slide1-8` – снимките, използвани в менюто „помощ“.

layout resources

Тук е цялата визия на страниците. Подробно описание на логиката, която стои зад всеки `layout` се намира в описанието на **Activities package** в описанието на бизнес логиката на приложението.

menu resources

Съдържа визуалната част на горното меню(top_menu) и долното меню(bottom_navigation_menu).

mipmap resources

Съдържа иконата на android приложението.

Values resources

Съдържа всички цветове, използвани текстови низове, теми и други в приложението.

Входни данни

Потребителите регистрират потребителски профили в системата със следните полета: име, имейл адрес, телефонен номер и парола.

Локално се пазят само имейл адресите и техните пароли, при включена от потребителя функция „вход с пръстов отпечатък“.

Потребителите могат да добавят дестинации със следните полета: име на мястото и местоположение, което избират от картата. Останалите полета се попълват автоматично: разстоянието се смята автоматично, спрямо текущата локация, а снимката се взима също директно от облака на гугъл. „Посетено“ и „Любими“ са булеви променливи, които по подразбиране са false.

Потребител може да присвоява и туристически обект към своя списък, като се създаде копие на избрания туристически обект и се свърже с референция на текущия потребител.

4. Реализация

Използване на горно и долно меню

Страниците, които използват горното и долното меню, имплементират логиката за селектиране на елементите им.

Горното меню се използва от HomeActivity, PlaceListView, NearestActivity, ProfileActivity и отговаря за отваряне на менюто „Помощ“:

```
private BottomNavigationView topMenu;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profile);

    topMenu = findViewById(R.id.top_menu);
    topMenu.setSelectedItemId(R.id.space); //да не е
                                           селектирано нищо при
```

```

                                                                    отваряне на страницата
Navigation navigation = new Navigation(email,
    ProfileActivity.this);
navigation.topMenu(topMenu);
    ...
}

```

Долното меню служи за навигиране между основните страници на приложението и се използва от абсолютно всички страници в приложението, без тези за вход(MainActivity), регистрация(SignUpActivity) и помощното меню(HelpActivity). Това улеснява навигирането на потребителя между страниците и ускорява работата му с приложението:

```

private BottomNavigationView bottomNavigationView;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profile);

    bottomNavigationView =
        findViewById(R.id.bottom_navigation);

    bottomNavigationView.setSelectedItemId
        (R.id.action_profile);

    // Set up bottom navigation view
    Navigation navigation = new Navigation(email,
        ProfileActivity.this);
    navigation.bottomNavigation(bottomNavigationView);
}

```

И двете менюта използват помощния клас Navigation, който следи за селектиран елемент и използва метод за навигация към страницата, свързана с този елемент.

```

public Navigation(String email, Context context){
    this.email = email;
    this.context = context;
}

public void bottomNavigation(BottomNavigationView
bottomNavigationView){
    bottomNavigationView.setOnNavigationItemSelectedListener(
        item -> {
            if(item.getItemId()== R.id.action_home){
                navigateToHomeActivity();
                return true;
            } else if(item.getItemId()==R.id.action_my_places){
                navigateToPlaceListView(1);
                return true;
            }
        }
    );
}

```

```

        ...
    });
}

```

Навигацията между страниците се извършва посредством Intent:

```

Intent placeViewIntent = new Intent(AddPlaceActivity.this,
PlaceListView.class);
placeViewIntent.putExtra("caseNumber", 1);
placeViewIntent.putExtra("email", email);
startActivity(placeViewIntent);
finishAffinity(); //забранява връщането към това активити с
стрлката „назад“

```

Използване на случайна снимка

Търсенето на случайна снимка по даден низ, се извършва чрез обръщение към външна система с помощта на инструмента Programmable Search Engine на Google:

```

URL url = new
URL("https://www.googleapis.com/customsearch/v1?key=" +
    "AIzaSyBSrFIVWfPGscGFskb3s3tl1crSYL5lq9A" + "&cx=" +
    "8662ccaade9c34971" + "&searchType=image&q=" + name);

// Open a connection to the URL
URLConnection connection = (URLConnection)
url.openConnection();
connection.setRequestMethod("GET");

```

Използване на картата

Използването на вградена карта в страницата налага употребата на командите, предоставени от библиотеката com.google.android.gms.maps.

```

GoogleMap mMap = googleMap

...
mMap.setOnMapClickListener(this); - „подслушва“ за кликване
върху картата
...
mMap.addMarker(new
    MarkerOptions().position(currentLocation).title("Marker
    at Current Location")); //поставя маркер на локация
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
    currentLocation, 15f)); //фокусира картата и я приближава

```

Използването на картата и навигацията налага потребителят да даде разрешение за достъп на приложението до локацията на устройството:


```
ActivityCompat.requestPermissions(this, new  
String[]{Manifest.permission.ACCESS_FINE_LOCATION},  
MY_PERMISSIONS_REQUEST_LOCATION);
```

Сензор за пръстов отпечатък

Подканянето на потребителя да се верифицира с пръстов отпечатък и използването на биометричните данни на устройството използват команди от библиотеката `androidx.biometric`:

```
BiometricPrompt.PromptInfo promptInfo = new  
BiometricPrompt.PromptInfo.Builder()  
    .setTitle("Моля потвърдете!")  
    .setDescription("задължителна ауторизация!")  
    .setNegativeButtonText("откажи")  
    .build();  
getPrompt().authenticate(promptInfo);
```

Адаптери

Използването на адаптер дава възможност всеки елемент в даден списък да бъде отделна единица. В случая се използва в списъците за показване на дестинации:

```
PlaceAdapter adapter = new PlaceAdapter(PlaceListView.this,  
filteredPlaces, email, caseNumber);  
placeListView.setAdapter(adapter);  
...
```

Callback интерфейси

Имплементират се във вложените класове в заявките към отдалечената база данни `Firebase`, като позволяват връщането на резултатът от заявката, посредством метода `onPlacesLoaded` и т.н.:

```
public interface PlacesCallback {  
    void onPlacesLoaded(List<Place> places);  
    void onError(Exception e);  
}
```

Връзка с базата данни

За използването на `Firebase` отдалечена база данни трябва да се добави библиотеката `com.google.firebase.firestore`. При всяко използване на отдалечената база, трябва да се създаде нейна инстанция:

```
FirebaseFirestore firestore = FirebaseFirestore.getInstance();
```

Следва работата с основните `CRUD` операции:

-Извличането на информация от определена таблица (колекция) в базата данни става по следния начин:

```
firestore.collection("places")
    .whereEqualTo("userEmail", userRef)
    .get()
```

-Промяна на поле от таблица:

```
firestore.collection("places").document(place.getId()).update(
    "distance", distance)
```

-Добавяне на нов запис(документ) в таблица(колекция):

```
db.collection("places").add(newPlace)
```

-Изтриване на запис по ID:

```
firestore.collection("places").document(documentId).delete()
```

Модели

NTO100

Този модел представя таблицата, която отговаря за 100-те национални туристически обекта.

```
private String id;//уникален идентификатор на националния
                    обект
private String name;//името на националния обект
private String urlMap;//линкът към местоположението на картата
private String imgPath;//линк към снимката на обекта
private int distance;//разстояние от текущото местоположение
private String numberInNationalList;//номер на обекта в
                                    националния списък
private String description;//описание на обекта
private boolean isActive;//дали обекта е активен в текущия
                        списък с националните обекти
```

Place

Place представя дестинациите на потребителите и репрезентира таблица places от базата данни.

```
private String id;//уникален идентификатор на мястото
private String name;//име на мястото
private String urlMap;//линкът към местоположението на картата
private String imgPath;//линк към снимката на обекта
private int distance;//разстояние от текущото местоположение
private boolean isFavourite;//дали мястото е добавено в любими
```

```
private boolean isVisited;//дали мястото е посетено
private DocumentReference userEmail;//връзка с потребителски
                               профил
private DocumentReference nto100;//връзка с национален обект
private String description;//описание на обекта
```

User

Този клас представя потребителските профили в системата и представлява users таблицата от външната база данни.

```
private String name;//име на потребителя
private String email;//имейл адрес на потребителя. Служи и за
                               уникален идентификатор
private String phone;//телефонен номер на потребителя
private String password;//хешираната потребителска парола
private boolean rememberMe;//функция „запомни ме“ при вход
private boolean notifications;//активни известия за дадения
                               потребителски профил
private boolean loginFirst;//флаг за първо влизане в
                               потребителския профил
private int points;//точките на потребителя
```

5. Потребителски интерфейс и ръководство

Потребителският интерфейс използва светли цветове. Освен това, приложението променя темата си, заедно с темата на устройството – тъмна и светла.

Навигацията между страниците ще се извършва предимно чрез навигационно меню и бутони.

Приложението използва предимно списъчни изгледи и текстови полета.

Вход

След стартиране на приложението се визуализира формата за вход. При отваряне на клавиатурата, бутоните, заедно с текстовите полета се качват нагоре върху снимката, за да предоставят на потребителя лесно влизане в системата, без да се налага да затварят клавиатурата след като въведат данните си.

Ако потребителят е използвал вече приложението и е активирал вход с пръстов отпечатък, устройството ще го подкани да се верифицира чрез него.

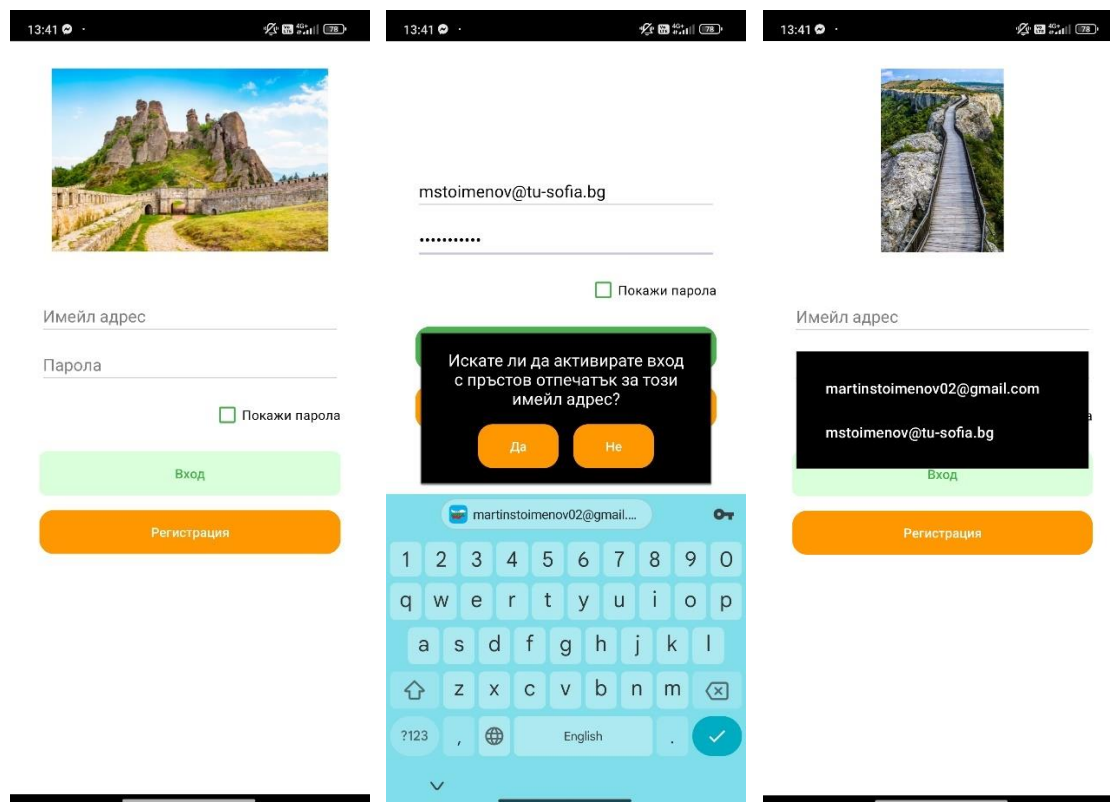
Ако пък за пръв път влиза в профила си, след като натисне бутона „Вход“, ще бъде попитан дали иска да включи верификация с пръстов отпечатък. Ако избере „да“, от следващия път ще може да се логва без да е необходимо да използва имейл и парола.

Ако потребителят разполага с повече от един профил и въведе данните и за втория, отново ще му бъде поискано разрешение да включи функцията „Вход с пръстов отпечатък“.

При следващото стартиране на приложението, след верифициране с пръстов отпечатък, на екрана ще се визуализира списък с всичките профили, за които е активна функцията „вход с пръстов отпечатък“ на това устройство.

Ако потребителят влиза в профила си за пръв път ще бъде автоматично пренасочен към менюто „Помощ“, където има кратко описание на функционалността на приложението.

Ако пък потребителят вече е влизал в профила си, ще бъде пренасочен към началната страница.



Регистрация

На формата за вход, освен възможност за влизане във вече съществуващ профил, има и бутон за регистрация на нови потребители.

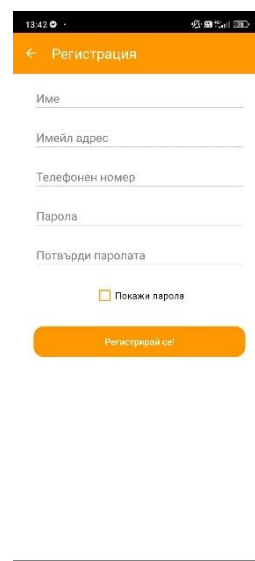
След натискане на бутона „Регистрация“ потребителят се препраща към форма, в която да въведе данните си. Данните трябва да са коректно въведени и да отговарят на следните изисквания:

- Всички полета трябва да са попълнени.

- Имейл адреса трябва да следва шаблона за имейл адрес: example@email.com.

- Телефонният номер може да съдържа само цифри или да започва с кода на държавата (+359, +49...). Освен това, позволява използването на стационарен телефон и неговия областен код (02/, 055/...).

- Паролата трябва да съдържа минимум 8 символа: малки, големи букви и цифри.



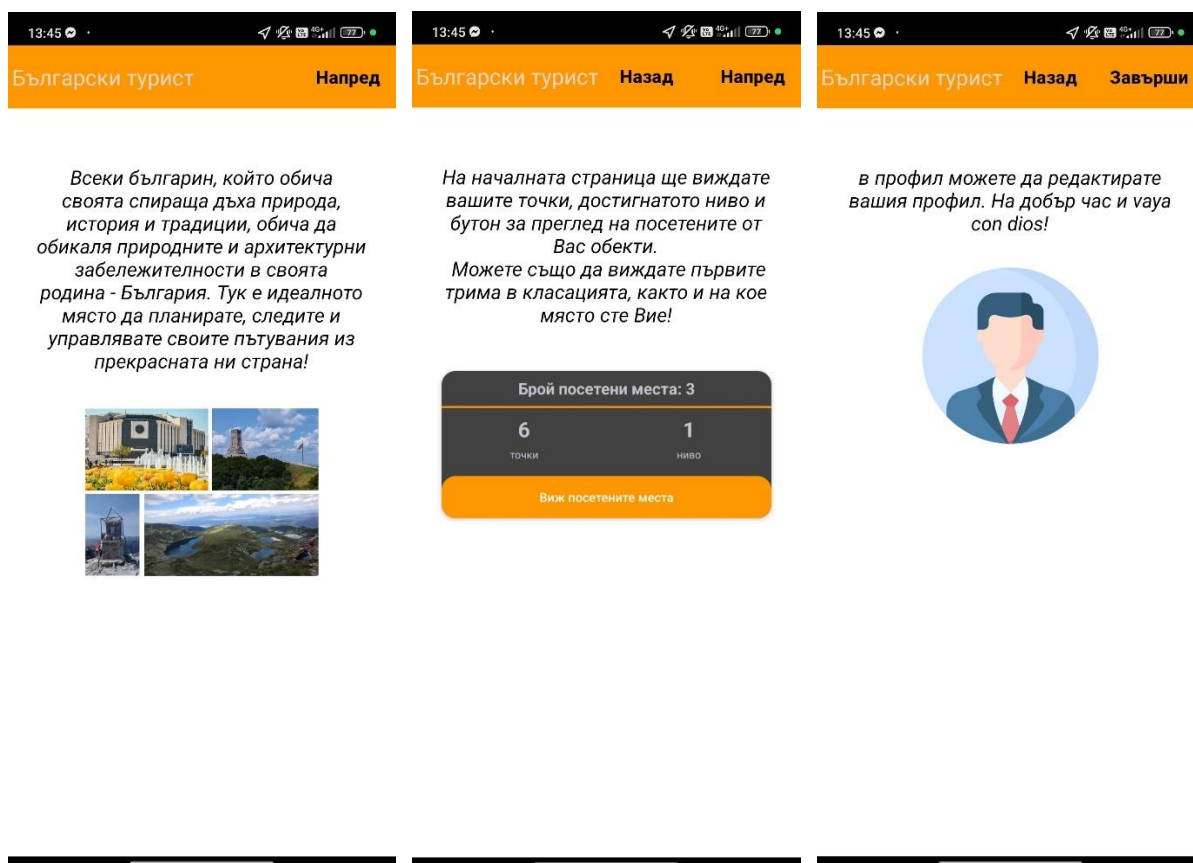
Ако данните не отговарят на този шаблон, бутонът за регистрация не е активен.

Създаване на профил с вече регистрирани имейл или телефон също е недопустимо.

След успешно въвеждане на данните във формата, потребителят се регистрира успешно в системата и се препраща към менюто „Помощ“, където има кратко описание на функционалността на приложението.

Меню „Помощ“

Менюто „Помощ“ се достъпва от въпросителната икона (?) в горния ляв ъгъл на приложението или се стартира автоматично при регистрация и първо влизане в потребителския профил. То съдържа основна информация за това как приложението функционира и улеснява работата на потребителя.



Начална страница

На началната страница се визуализират броя посетени дестинации, общия брой точки и нивото, което е достигнал потребителят, изчислено спрямо точките. Натискайки върху „Виж посетени места“ се отваря списък с всички посетени места, като всяко място може да бъде отворено в детайли.

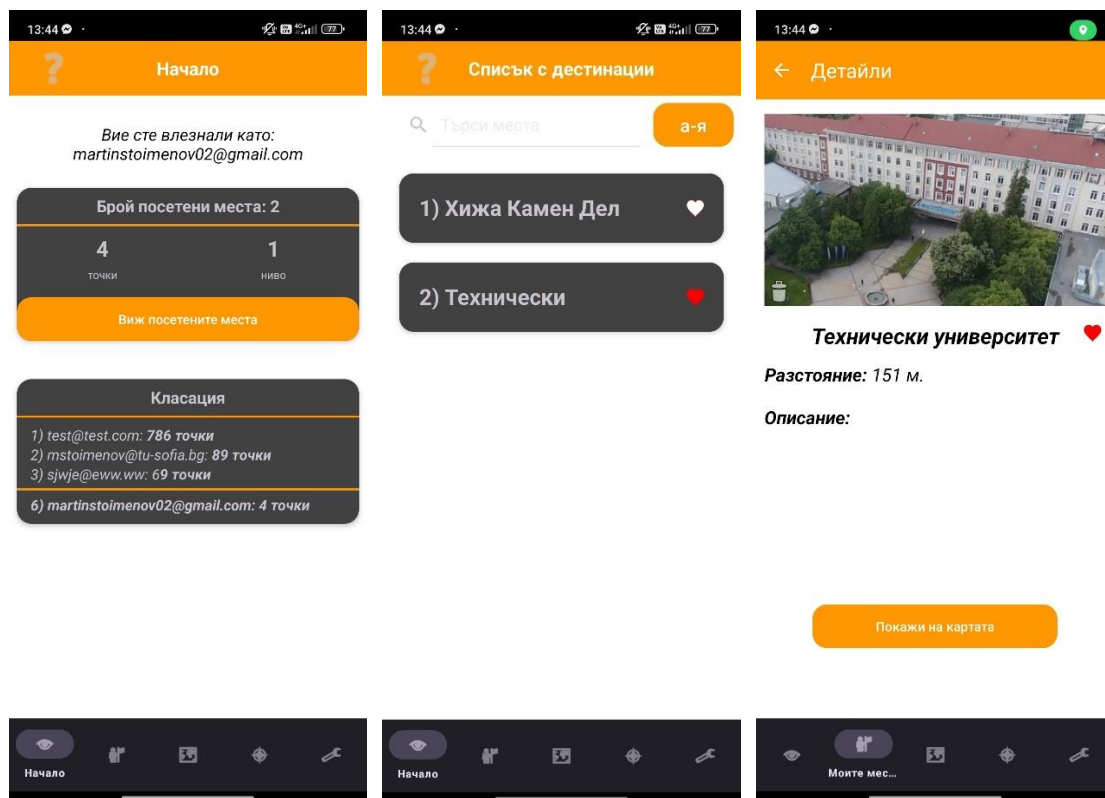
В страницата с детайлите, потребителят вижда снимката на дестинацията, името, описанието. Ако потребителят е пуснал локацията си, се изчислява разстоянието до мястото, в противен случай се подканва да я пусне. Ако изчислението на разстоянието продължи по-дълго от 20 секунди, се прекратява и не се изчислява повече. Причина за подобно забавяне би могла да бъде слабият интернет обхват на устройството на потребителя, също и разрешенията за използване на локацията на устройството от това приложение.

В страницата с детайли, потребителят може да добавя в любими дадената дестинация, да я изтрива или да я отваря на картата в Google Maps.

Всяка дестинация, която е от списъка със 100-те национални туристически обекта носи по 5 точки, а всяка друга – по 2. Ниво се вдига на всеки 100 точки.

Под информацията за посетените места и общия брой точки има класация, която показва потребителите на първите три места, заедно с техните точки, както и на кое място е текущият потребител.

От навигационното меню в долния край, потребителят може да навигира към останалите страници на приложението.



Страница „Мои места“

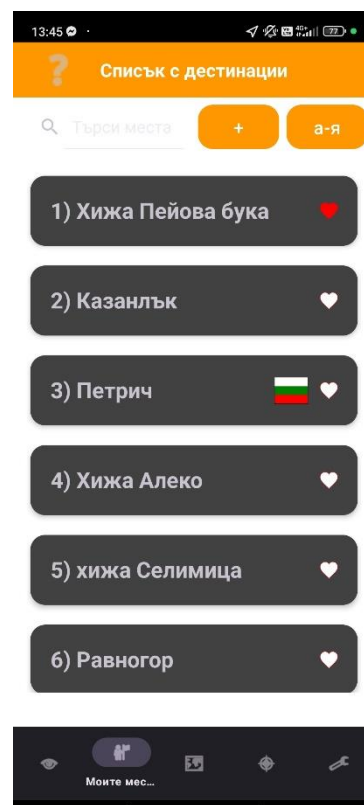
От навигационното меню, избирайки втората икона, се отваря списък с „Мои места“. Това е списъкът, в който се добавят различните дестинации, които желае потребителят да посети.

Всяко място в списъка може да бъде добавено в списъка с любими места, посредством бутона с икона във формата на сърце. Добавено място запълва сърцето с червено, в противен случай то е само контур.

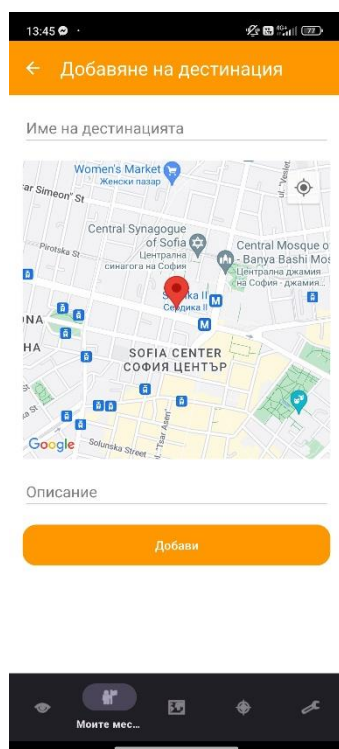
Всяка дестинация, част от 100-те национални туристически обекта има в своето поле българския флаг, за да се ориентира потребителят по-лесно.

Бъдещото развитие на приложението предполага добавяне на сортиране на списъка по име, по национален обект, по любими и по разстояние до обектите.

С бутона „Добави“ потребителят добавя още дестинации, които иска да посети.



Добаѝ място



В страницата за добавяне на място, потребителят трябва да въведе име(задължително) и описание(незадължително) на своята дестинация. Освен това, той трябва задължително да избере локацията на мястото което иска да посети от картата, която се визуализира. Ако локацията на устройството му е пусната, картата е центрирана спрямо това. Ако не е пусната, картата се центрира в центъра на София.

Останалите полета на дестинацията се попълват автоматично: снимката се търси в Google по въведенето име, разстоянието до дестинацията се изчислява автоматично, при отваряне на детайли за мястото и т.н.

Въведенето име се сравнява със списъка със 100-те национални туристически обекта и ако то съвпада с някой от тях, се закача към него.

Бъдещото развитие на проекта предвижда да може да се търси дестинация на картата, по въведенето име.

Детайли



Всяко място от списъка с „моите места“ може да се отвори в детайли, където се вижда цялата информация за него: снимка, разстояние от текущото местоположение, описание, дали е национален обект.

Всеки път при отваряне на страницата с детайли се обновява разстоянието до даденото място. То се обновява и периодично, докато потребителят е на същата страница. Крайното време за изчисляване на разстоянието е 20 секунди. След изтичането им потребителят се информира, че разстоянието не може да бъде изчислено. Това означава, че потребителят не е дал необходимото разрешение на приложението за използване на локацията, не е с

включена локация, или има слаб или никакъв интернет обхват.

С бутона „Покажи на картата“ локацията се отваря директно в google maps, което улеснява потребителя в навигацията.



С бутона „Посети“, мястото може да бъде отбелязано като „посетено“. За да се случи това, обаче, потребителят трябва да е с пусната локация и да бъде не по-далече от 500 метра от мястото.

От тази страница мястото също може да бъде добавяно в списъка с любими и може да бъде изтривано, посредством иконата „Кошче“.

100 НТО

В тази страница от навигационното меню се намира списъкът с всички национални туристически обекти. Бъдещото развитие на приложението предполага добавяне на сортиране по име, по номер в националния списък на туристическите обекти и по разстояние до обектите.

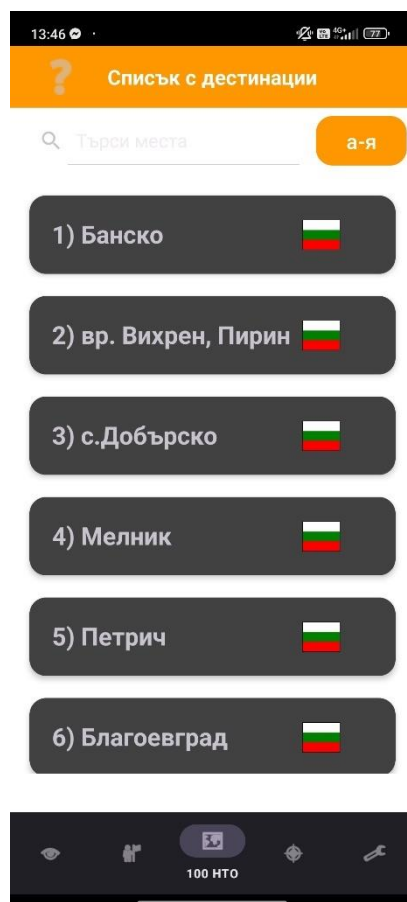
Детайли на национален туристически обект



В прозореца с детайли, потребителят отново вижда подробна информация за обекта: снимка, разстояние от текущото местоположение, описание. Тук отново разстоянието се опреснява периодично. Крайното време за изчисляване на разстоянието е 20 секунди. След изтичането им потребителят се информира, че разстоянието не може да бъде изчислено. Това означава, че потребителят не е дал необходимото разрешение на приложението за използване на локацията, не е с включена локация, или има слаб или никакъв интернет обхват.

С бутона „Покажи на картата“ локацията се отваря директно в google maps, което улеснява потребителя в навигацията.

С бутона „Добави за посещение“ текущият туристически обект се добавя в списъка с „моите места“, готов за посещение.



Профил

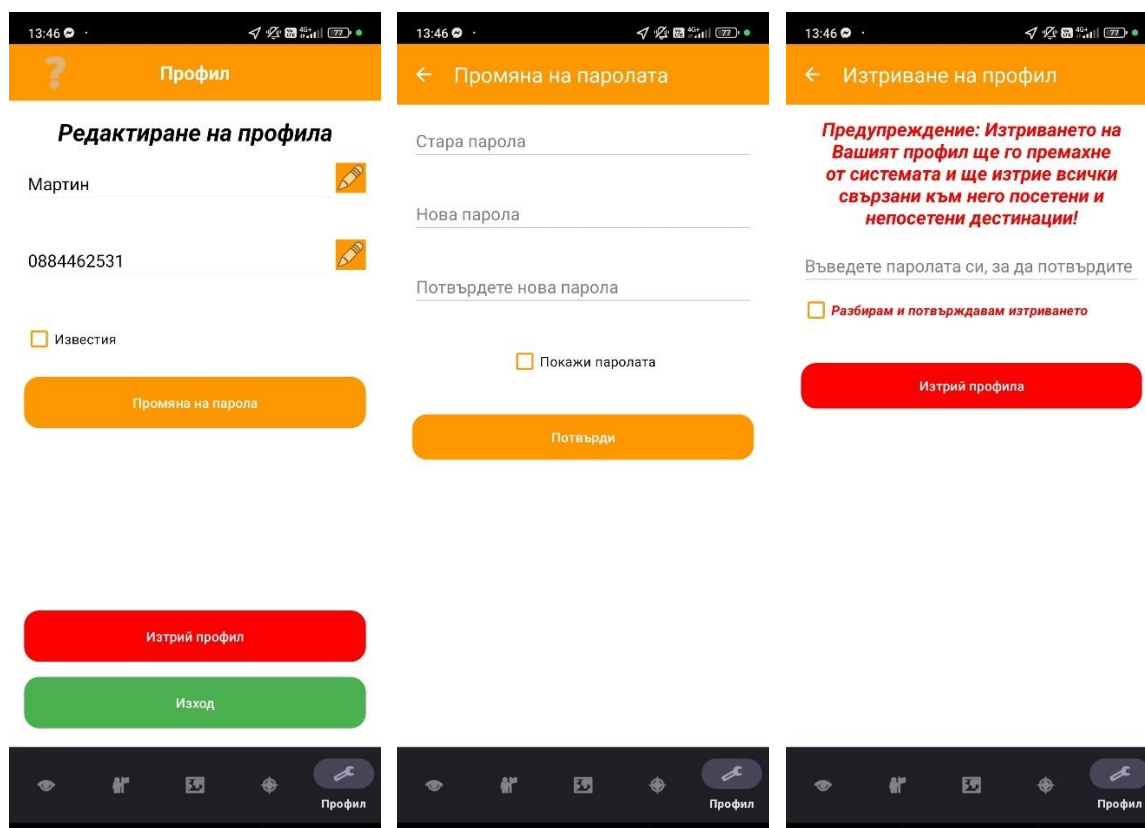
В страницата „Профил“ от навигационното меню, потребителят вижда информацията за своя профил и може да я редактира(име, телефон) посредством иконата „Молив“.
След като промени информацията си, той я запазва с бутона с икона „Дискета“.

От тук потребителите могат да променят паролата си. За целта е необходимо да въведат старата си парола, след това новата и да я потвърдят. Ако старата парола е невалидна, смяната е невъзможна. Новата парола отново трябва да отговаря на шаблона за пароли на приложението: поне 8 символа, да съдържа малки, големи букви и цифри. Ако всички полета са попълнени коректно, бутона за потвърждение става активен и потребителят променя паролата си. За в бъдеще е предвидена функционалност „Забравена парола“ на страницата за вход.

От тук потребителят позволява получаването на известия за бъдещата функция, когато е в близост до даден обект.

От тук потребителят може да изтрива профила си. Изтриването на потребителския профил, води до изтриване на всички свързани с него дестинации, което означава че той ще си загуби всички посетени и непосетени места, както и точките. За да изтрие профила си, потребителят трябва да се съгласи с предупреждението и да въведе паролата си.

В „профил“ се намира и бутона за изход от текущия профил. Потребителят се препраща към страницата за вход в приложението.



В близост

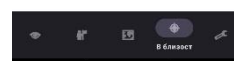
Бъдещото развитие на приложението включва възможността потребителите да виждат всички места в близост до 20 км от текущото им местоположение, което да улесни организацията на пътуването им.

Освен визуализацията на най-близките дестинации от списъците с лични места и от 100-те национални обекта, потребителят ще може да разреши получаването на известия в случай, че се приближи до някоя дестинация, дори приложението да не е активно.

Към този момент на тази страница, потребителят се оповестява, че тази функционалност все още не е налична и предстои да бъде разработена.



Очаквайте скоро



6. Бъдещо развитие на проекта

Бъдещото развитие на приложението предполага добавянето на нови полезни функционалности, които да привлекат повече потребители и да улеснят тяхното пътуване.

В близост

Това е функционалност, чрез която потребителите ще могат да проследят всички дестинации в близост до 20 км. от текущото им местоположение и да получават известия в случай, че се приближат до някоя дестинация, дори приложението да не е активно.

Функция “запомни ме”

Чрез тази функция потребителите ще бъдат навигирани директно към началната страница на приложението след като го отворят и няма да се налага да въвеждат имейл адрес и парола или да се идентифицират с пръстов отпечатък всеки път.

Автоматично вземане на информация от гугъл

При създаване на място, както и за всички национални обекти, да бъде възможно взимането на актуална информация за дестинацията от гугъл, като работно време и телефон, за да могат потребителите да са наясно кога ще имат възможността да го посетят.

Посещаване на мястото посредством снимка

С напредването на технологиите и навлизането на изкуствения интелект, става все по-възможна реализацията на „умни“ приложения. Бъдеща функционалност предвижда посещаване на мястото, посредством снимка. Тогава AI ще може да сравни направената снимка с други снимки на дадената дестинация и да прецени дали потребителят е на мястото.

Потребителска обратна връзка

Обратната връзка от потребителите на едно приложение е много важна за разработчиците и притежателите му. Тя помага то да се развива. За да покрие потребителските очаквания, бъдеща функционалност на приложението е добавянето на възможността потребителите да могат да репортват и да дават отчети, идеи и предложения за подобрене и добавяне на нови полезни функционалности.

Приложение за iOS

Бъдещото развитие на приложението налага възможността за употреба на повече устройства, което води до нуждата то да бъде достъпно и за потребителите на iPhone.

7. Заключение

„Български турист“ е функциониращо приложение, готово да бъде пуснато за употреба. Смятам, че би могло да бъде конкурентно на посочените съществуващи разработки.

Споделяйки идеята за приложение с мои колеги и приятели, повечето от тях я оцениха като необходимост и биха използвали приложението, за да организират по-добре посещенията на туристически и не само дестинации в България!

Как ви се струва идеята за приложение, в което да можете да добавяте и планирате своите пътувания из цяла България?

 Copy

19 responses



8. Използвана литература

1. Firebase

<https://firebase.google.com/docs/android/setup>

https://www.youtube.com/watch?v=aiX8bMPX_t8&ab_channel=EasyTuto

https://firebase.google.com/docs/firestore/query-data/queries#java_2

https://www.youtube.com/watch?v=_gpreGNtNCM&ab_channel=CodingTutorials

2. Location

<https://developer.android.com/develop/sensors-and-location/location/permissions>

<https://developer.android.com/develop/sensors-and-location/location/retrieve-current>

3. FingerPrint

https://www.youtube.com/watch?v=RInxqVYnvU8&ab_channel=CodingWithEvan

4. XML layouts

4.1. Cardview - <https://developer.android.com/jetpack/androidx/releases/cardview>

4.2. App icon - <https://developer.android.com/codelabs/basic-android-kotlin-compose-training-change-app-icon#2>

4.3. Center - <https://stackoverflow.com/questions/26533510/android-toolbar-center-title-and-custom-font>

4.4. Icons - https://developer.android.com/reference/android/R.drawable#ic_media_rew

4.5. Toas - <https://developer.android.com/guide/topics/ui/notifiers/toasts>

5. Други

5.1. Материалите от упражненията