Дипломна Работа

Тема:

Проектиране и разработване на андроид мобилно приложение за целите на рокерската общност

Дипломант: Марин Николаев Маринов

Факултетен № 121216061

Специалност: Компютърно и софтуерно инженерство

Научен ръководител: гл. ас. д-р инж. Иван Станков

Съдържание

[1. Увод 3](#_Toc64129119)

[1.1 Цел и задача на дипломната работа 4](#_Toc64129120)

[1.2 Структура на дипломната работа 4](#_Toc64129121)

[2. Идеен проект 5](#_Toc64129122)

[3. Използвани технологии 5](#_Toc64129123)

[3.1 Git, Github, npm и Node.js 5](#_Toc64129124)

[3.2 Cordova и Phonegap 6](#_Toc64129125)

[3.3 JavaScript, CSS и HTML 6](#_Toc64129126)

[3.4 jQuery 7](#_Toc64129127)

[3.5 Onsen UI 7](#_Toc64129128)

[3.6 Firebase 8](#_Toc64129129)

[3.6.1 Cloud Firestore 8](#_Toc64129130)

[3.6.2 Firebase Authentication 8](#_Toc64129131)

[3.6.3 Cloud Storage 9](#_Toc64129132)

[3.7 Atom IDE 9](#_Toc64129133)

[3.8 Google Maps API 10](#_Toc64129134)

[4. Архитектура 11](#_Toc64129135)

[4.1 Архитектура на приложението 11](#_Toc64129136)

[4.2 Архитектура на базата данни 14](#_Toc64129137)

[5. Разработка 17](#_Toc64129138)

[5.1 Инициализиране на проект 17](#_Toc64129139)

[5.2 Инсталиране на приставки за достъп до локация и камера 18](#_Toc64129140)

[5.3 Дизайн на приложението и основни библиотеки 18](#_Toc64129141)

[5.4 Структуриране на проекта 19](#_Toc64129142)

[6. Ръководство за потребителя 24](#_Toc64129143)

[6.1 Системни изисквания и инсталация 24](#_Toc64129144)

[Системни изисквания 24](#_Toc64129145)

[Инсталация: 24](#_Toc64129146)

[6.2 Вход и регистрация 26](#_Toc64129147)

[6.3 Начален екран, основно меню и настройки на профила 27](#_Toc64129148)

[6.4 Списък с потребители, „Любими потребители“, Допълнителна информация 29](#_Toc64129149)

[6.5 Карти 34](#_Toc64129150)

[6.6 Социална стена 35](#_Toc64129151)

[6.7 Чат стаи 38](#_Toc64129152)

[7. Идеи за развитие 41](#_Toc64129153)

[7.1 Версия за iPhone I iPad 41](#_Toc64129154)

[7.2 Добавяне на още езици и възможност потребителя да си избира 41](#_Toc64129155)

[7.3 Добавяне на потребителски роли „Модератори и администратори“ 42](#_Toc64129156)

[7.4 Възможност за промяна и изтриване на съобщения от обикновените потребители 42](#_Toc64129157)

[7.5 Възможност за коментиране на постовете в „Социалната стена“ 42](#_Toc64129158)

[8. Заключение 42](#_Toc64129159)

[9. Използвани източници и референции 43](#_Toc64129160)

[10. Приложение 43](#_Toc64129161)

# Увод

Живеейки в XXI век, ние ставаме свидетели на изключително бързото развитие на технологиите. Те съпътстват нашия живот в абсолютно всеки аспект, без изключение, което ги прави и необходимост, и забавление. Всеки от нас притежава смартфон в наши дни и комуникацията и свързаността се развиват в много бързи темпове, Това от своя страна води до постоянно увеличаване в търсенето им и съответно до по-голям пазар. Използването на мобилни приложения за всякакви цели като игри, слушане на музика, споделяне на снимки, четене на книги, запознанства и комуникация с други. Това улеснява търсенето на подходящи уеб-сайтове за изброените цели, като ни предоставят готов продукт отговарящ на нуждите ни.

Писането на мобилно приложение за различните операционни системи изисква богати познания и умения за работа с определени езици за програмиране swift/objective-c за Apple и iOS базирани устройства, Java за Андроид, C# за windows phone и др. Тук на помощ за разработчиците идват системите, които предлагат създаването на крос платформените приложения, които написани на една технология се превеждат до C#, swift, Java.

Статистиката показва, че потенциала на крос платформените приложения расте всяка изминала година. Двете големи предимства на крос-платформеното програмиране са: финансова ефективност и спестяващи време. Някои от най – добрите крос – платформени мобилни инструменти за разработка са: Xamarin, PhoneGap, appcelerator, alphaSoftware, React Native, Vue Native, Ionic.

## Цел и задача на дипломната работа

Целта на дипломната работа е да се създаде мобилно приложение, което да обедини мото общността на България. Да предлага следните функционалности:

* регистрация и вход
* създаване на профил
* споделяне на социални мрежи и данни за контакт
* организиране на чат стаи
* споделяне на снимки
* споделяне на постове
* споделяне на настоящо местоположение.
* запазване на потребители в „Любими“
* възможност за преглед на всички потребители

## Структура на дипломната работа

* Увод – изяснява целите и задачите на дипломната работа
* Идеен проект – представя основната идея на приложението
* Използвани технологии – дава информация какви технологии се използват за разработването на приложението
* Архитектура – дава информация за архитектурата и базата данни
* Разработка – разкрива начина по който е реализирано приложението
* Ръководство на потребителя – представя интерфейса и обяснява как се използва приложението
* Бъдещо развитие – дава информация за бъдещи планове и развитие на следващите версии на приложението
* Заключение – обобщава съдържанието и дава ясно дали целите за изпълнени успешно
* Използвана литература – съдържа всички материали, които са използвани за написването на дипломната работа

# Идеен проект

Основната идея на дипломната работа е да се разработи мобилно приложение  
 „BG MOTORISTI“ , която основна цел е да предостави среда на определена група от хора с общи интереси, в която могат да си помагат, да споделят полезна информация и да си организират срещи. Основния интерфейс функционален, опростен, лесен за употреба и подходящ за всички мобилни устройства и таблети.

При регистрация всеки потребител може да качи своя снимка и сподели своите контакти като Facebook, Instagram, Email, телефонен номер, Скайп, както и външен сайт източник.  
Тази информация може да бъде открита в списъка с всички потребители или през картата.

Чрез картата потребителят може да види къде и кога са се намирали останалите потребители, като е отбелязано и неговото текущо местоположение. Да избира между сателитен изглед или обикновен, да обновява текущата си локация както и да използва street view изглед.

На социалната стена потребителя може да споделя текстови постове с или без снимка. Постовете могат да бъдат харесвани. Идеята е вдъхновена от Туитър, Стената на Фейсбук и Инстаграм.

Чат групите са мястото където потребителите могат да си правят чат стаи на база интереси и да си пишат един с друг. Да качват снимка за корица на стаята и кратко описание. Идеята е вдъхновена от форумите на различните мото групи.

# Използвани технологии

## Git, Github, npm и Node.js

Git е безплатен софтуер с отворен код, който представлява система за разпространение, контролиране с версии на малки и големи проекти. GitHub е уеб услуга, използваща Git. През 2002 г., Linux kernel проектът започва да използва патентована DVCS система наречена BitKeeper за разработка. През 2005г. отношенията между компанията BitKeeper и разработчиците на Линукс се развалят, което довежда до търсенето на алтернатива. Така е създадена нова система Git от Линус Торвалдс за управление на разработката на Linux през 2005. Поради нуждата да се контролира огромната база от код на Linux ядрото, основна цел при разработката на Git е била бързината, опростения дизайн и напълно разпределената работна среда. Всяка локална Git директория е хранилище с пълна история и възможности за следене на версиите. Това прави Git независим от мрежови връзки към централен сървър. [[1]](Git_book" \l "_[1]_)npm е най-големият софтуерен регистър в света. Разработчиците по цял свят използват npm за да споделят и разменят пакети по подразбиране за средата за изпълнение на JavaScript Node.js. Те могат да публични и частни (платени). npm, Inc. е дъщерно дружество на GitHub. Състои се от клиент на команден ред, наричан още npm, и онлайн база данни с пакети, наречена npm регистър. [[2]](npm" \l "_[2]_)Node.js (или само Node) е сървърно приложение, създаващо среда за изпълняване на JavaScript код (runtime environment), а не език за програмиране. Node.js прави възможно изпълнението на JavaScript код директно на сървъра, както обикновено това се случва в уеб браузъра. Когато се каже, че дадено приложение е написано на Node.js, се разбира, че то е написано на JavaScript и се изпълнява от Node.js. [[3]](Какво_е" \l "_[3]_)

## Cordova и Phonegap

Apache Cordova (по-рано PhoneGap) е рамка за разработване на мобилни приложения, създадена от Nitobi. Adobe Systems закупува Nitobi през 2011 г., преименува го на PhoneGap и по-късно пусна версия с отворен код на софтуера, наречена Apache Cordova. Apache Cordova дава възможност на софтуерните програмисти да създават хибридни уеб приложения за мобилни устройства, използващи CSS3, HTML5 и JavaScript, вместо да разчитат на специфични за платформата API, като тези в Android, iOS или Windows Phone.Apache Cordova се грижи да опакова в инсталационен файл в зависимост от операционната система (app.apk за Android, app.ipa за ios) CSS3, HTML и JavaScript. [[4]](Cordova" \l "_[4]_) Получените приложения са хибридни, което означава, че те не са нито native мобилно приложение (защото цялото изобразяване на оформлението се извършва чрез уеб изгледи вместо вградената рамка на потребителския интерфейс на платформата), нито чисто уеб базирани (тъй като те не са само уеб приложения, а са пакетирани като приложения за разпространение в магазините (app.apk, app.ipa) и имат достъп до API на инсталираното устройство). Apache Cordova идва също с удобни инструменти като Phonegap Comand Line Interface и Phonegap Desktop App, които помагат в разработката и предоставят симулатор за тест. Команди за стартираране на нов проект, създване на нужната платформа или пакетиране на приложението в инсталационен файл.

## JavaScript, CSS и HTML

HTML е основният маркиращ език за описание и дизайн на уеб страници. Той е стандарт в Интернет, а правилата се определят от международния консорциум W3C. Текущата версия на стандарта е HTML 5.0. Той използва отварящи и затварящи тагове <div>(съдържание)</div> за да укаже къде точно започва и завършва съдържанието на даден елемент от страницата. На всеки елемент могат да му се дават различни атрибути (id, class, type и др.) за да може да се обработват по-лесно със CSS и JavaScript. Предназначението на уеб браузърите е да могат да прочетат HTML документите и да ги превърнат в уеб страници. Браузърите не показват HTML таговете, а ги използват, за да интерпретират съдържанието на страницата.

CSS (Cascading Style Sheets) е език за описание на стилове, използва се основно за описване на представянето на документ, написан на език за маркиране (HTML, XML…). CSS е създаден с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Стиловете могат да се пишат заедно със елементите на документа който описват (inline). Но това прави документа дълъг и трудно четим. Затова се въвежда възможността да се пишат на отделен файл с разширение (.css) да бъдат импортиран целия файл със стилове в HTML документа.

Javascript език за програмиране, разпространяван с повечето Уеб браузъри. Поддържа обектно-ориентиран и функционален стил на програмиране. Създаден е в Netscape през 1995 г. от Брендан Ейк. Последните проучвания за 2017 показват, че 94,9% от всички уебсайтове в мрежата използват този език за програмиране. [[5]](За_какво" \l "_[5]_) Може да се използва за разработка на мобилни приложения, десктоп и уеб апликации, зареждане на информация чрез AJAX, създаване на анимации, презентации, поп-ъп прозорци и др. JavaScript не трябва да се бърка с Java . Javascript е стандартизиран под името EcmaScript и е разработен първоначално от Брендан Айк под името Mocha, като по-късно е преименуван на LiveScript и накрая на JavaScript. LiveScript е официалното име на езика когато за първи път бива пуснат в бета версиите на Netscape Navigator 2.0 през септември 1995 г., но е преименуван на JavaScript на 4 декември 1995г.[[6]](JavaScript:_The#_[6]_)

## jQuery

jQuery е разпространена библиотека на JavaScript, публикувана в началото на 2006 от Джон Резиг. В основата си jQuery опростява достъпа до всеки елемент на дадена уеб-страница, като по този начин позволява лесно изграждане на динамична функционалност в страниците. По статистика към днешна дата jQuery се използва във 77.3% от всички сайтове в световен мащаб[[7]](Статистика_за#_[7]_). Библиотеката се поддържа от всички съвременни версии на уеб браузърите. jQuery ни дава възможност за улеснена манипулация на елементи от страницата, динамично добавяне и премахване на класове, атрибути от елементите, добавяне на анимации както и разширени функции за обработка на типове данни в JavaScript (обекти, масиви, низове, числа). Опростен синтаксис за изпращане на заявки и други. [[8]](jQuery" \l "_[8]_)

## Onsen UI

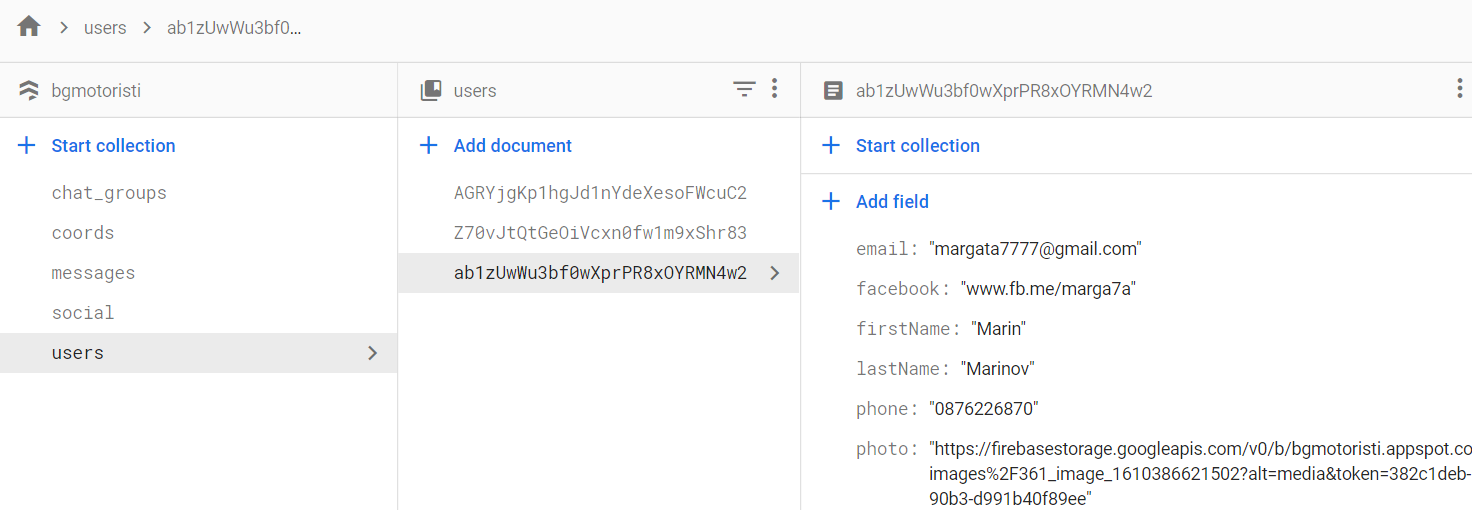
Onsen UI е библиотека с отворен код, която предоставя HTML компоненти подходящи за разработка на мобилни приложения със Cordova. Създадена през 2013г., първоначално е базирана на Angular JS, с поддръжка за jQuery. С излизането на версия 2, Onsen UI става самостоятелна JavaScript библиотека т.е. може да работи не зависимо без допълнителни библиотеки.

## Firebase

Firebase е платформа, разработена от Google за създаване на мобилни и уеб приложения. Първоначално е независима компания, основана през 2011 г., но през 2014 г. Google придобива платформата и сега е водещото им предложение за разработка на приложения. Firebase предлага 18 услуги разделени в 3 основни групи: Development, Quality and Grow.

### 3.6.1 Cloud Firestore

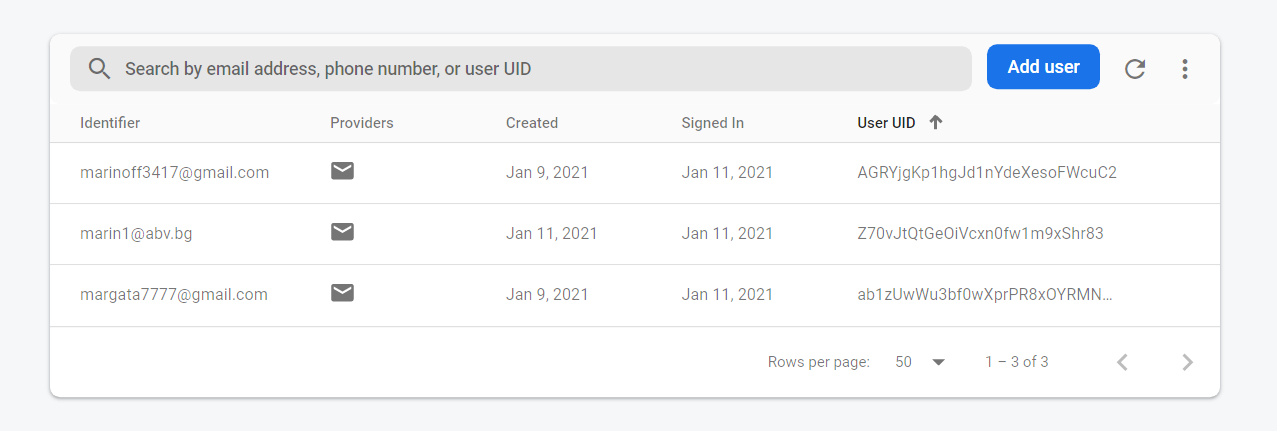
Cloud Firestore е нерелационна база данни подходяща за мобилни и уеб приложения. Предлага лесен запис на данни, синхронизация и извличане на данните във формат JSON. Лесно структуриране на данните с колекции и документи. Извличане на нужните данни чрез изразителни заявки. Всички заявки се мащабират с цел намаляване на размера от данни. Предоставя възможност да се закачим за избрана колекция и да ни информира при настъпване на промени, така приложението се синхронизира с базата в реално време и показва винаги актуална информация.



фигура 1. Представя структурата на базата данни вляво колекциите  
в средата документа за всеки запис и вдясно полетата на документа.

### 3.6.2 Firebase Authentication

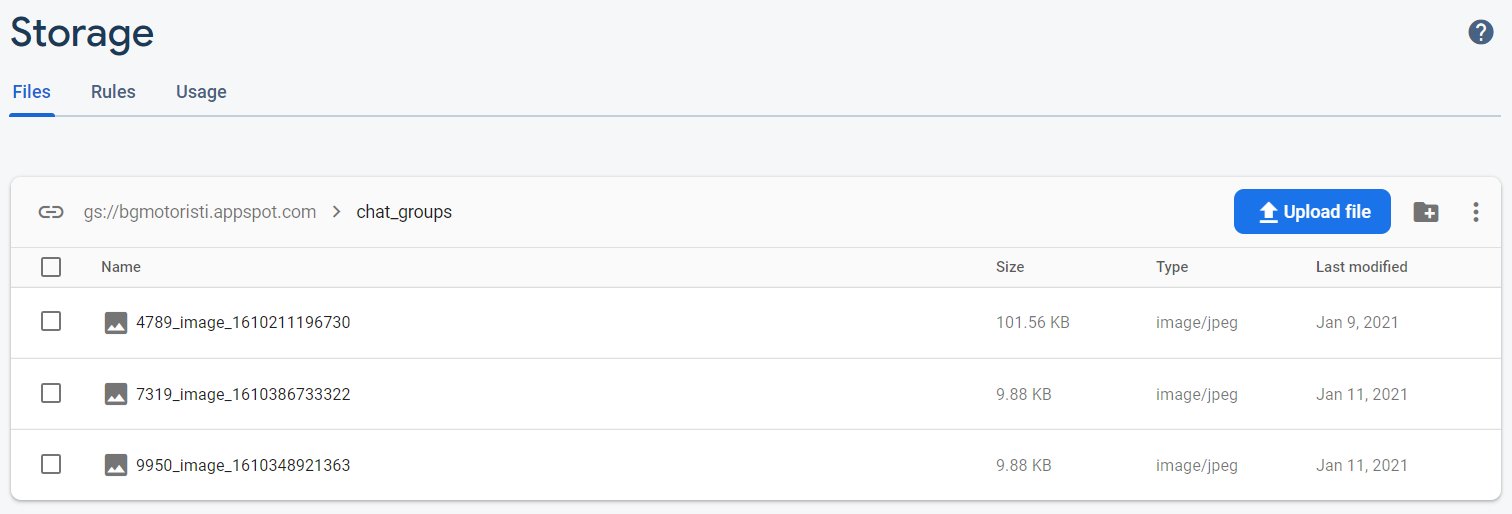
Firebase Authentication има за цел да улесни изграждането на защитени системи за удостоверяване, като същевременно подобри практиката за влизане и включване за крайни потребители. Той предоставя цялостно решение за самоличност, поддържащо акаунти с имейл и парола, удостоверяване на телефона и влизане чрез Google, Twitter, Facebook и GitHub и др. Създадена от същия екип, разработил Google Sign-in, Smart Lock и Chrome Password Manager, Firebase security прилага целия опит на Google за управление на една от най-големите бази данни за акаунти в света.[[9]](Firebase_Authentication#_[9]_)



фигура 2. Списък с регистрираните потребители.

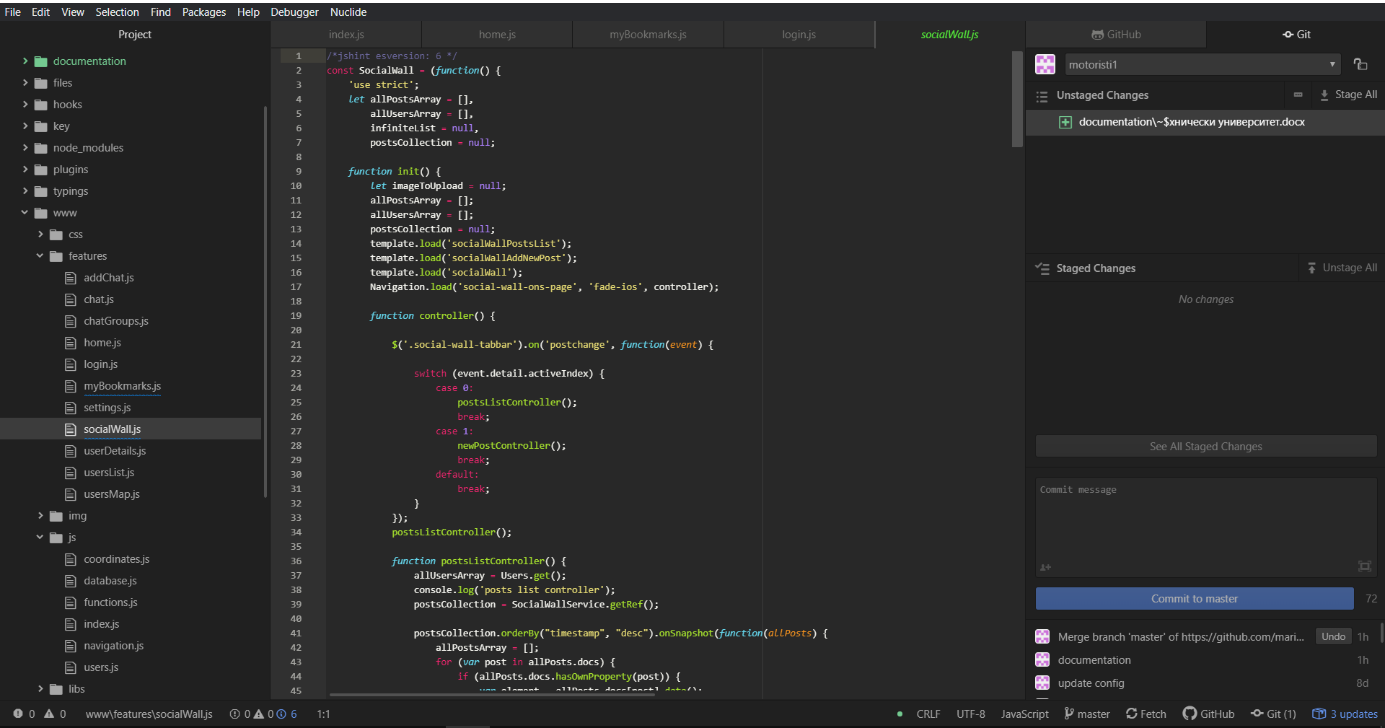
### 3.6.3 Cloud Storage

Cloud Storage е създадена за да може лесно и бързо да се съхранява снимки и видеа, като предоставя и добра синхронизация между себе си и Cloud Firestore.



фигура 3. Списък с качените файлове в Cloud Storage.

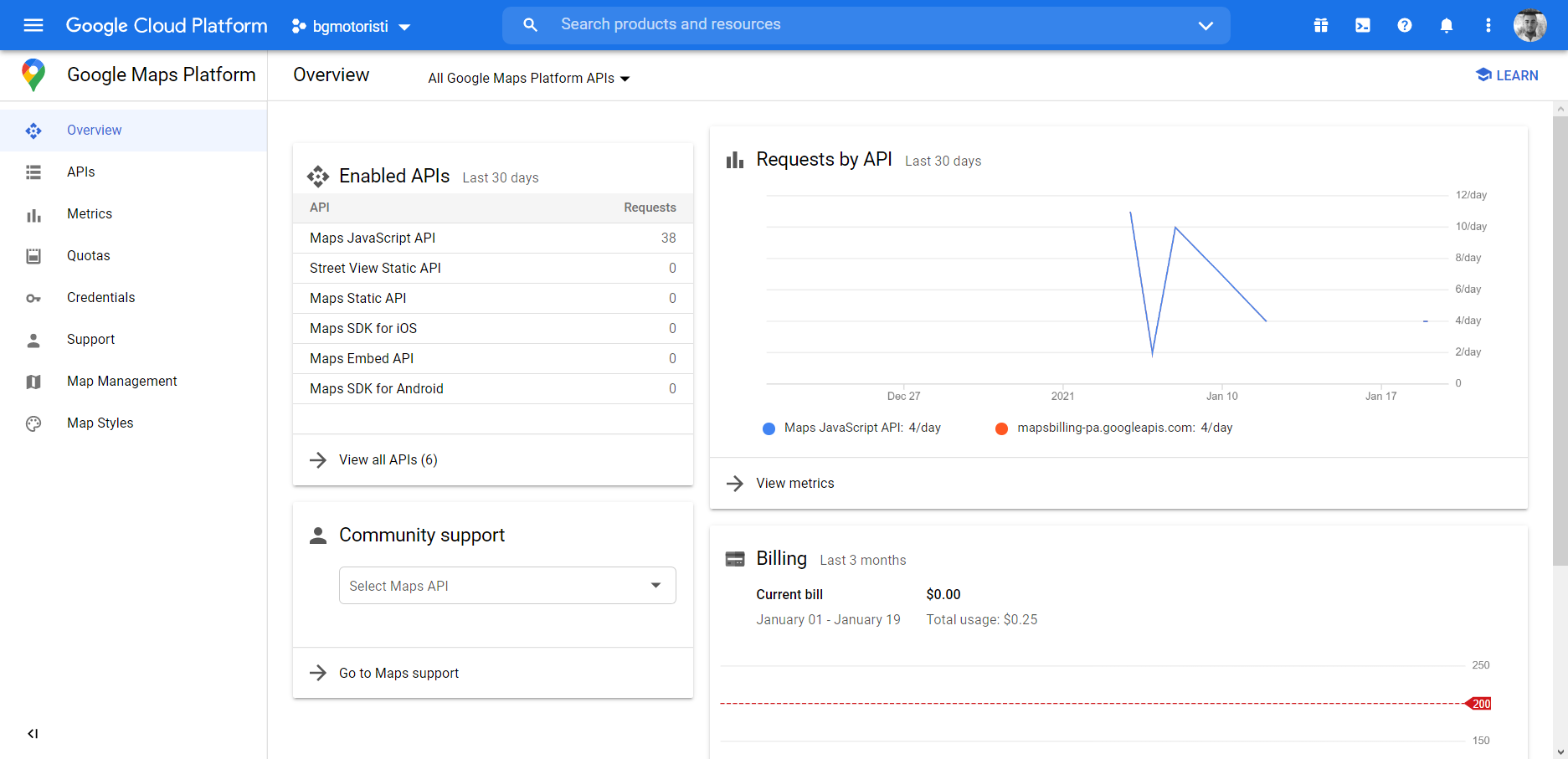
## Atom IDE

Atom е безплатен текстов редактор с отворен код и подходящ за macOS, Linux и Microsoft Windows. Разработен от GitHub, той предлага вграден Git Control. Предлага и множество приставки и за улеснение по време на разработката както и различни теми, които могат да бъдат настройвани по избор. Има и много разширения с безплатен лиценз, които се поддържат от общността. Atom е написан е на Coffee Script и LESS. Първата версия на Атом излиза на 25 юни 2015г.

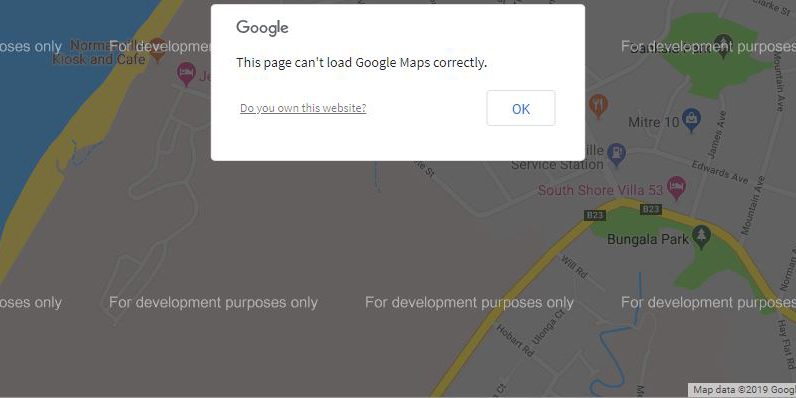
фигура 4. Изглед от Atom IDE

## Google Maps API

Гугъл карти е картографска услуга на Гугъл, с която могат да се разглеждат географски карти в произволен уеб браузър. Картите са с променлив мащаб и показват широк спектър от информация за разглеждания регион или местност. Google Maps API ни позволява да вградим Гугъл карти в наша уебстраница използвайки JavaScript. Настоящата версия на Maps API е 3. За да използваме Гугъл карти трябва да си направим акаунт в Google Cloud platform и да дадетем данни за разплащателна сметка, защото услугата не е безплатна. След като го направим получаваме ключ, който използваме като инициализираме картата в нашия проект. Може да използваме и картите без да предоставяме сметка за плащане но в такъв не могат да се използват пълноценно. Освен това има и съобщение, което казва, че картите са в режим на разработка



фигура 6. Изглед от моя акаунт в Google Cloud Platform

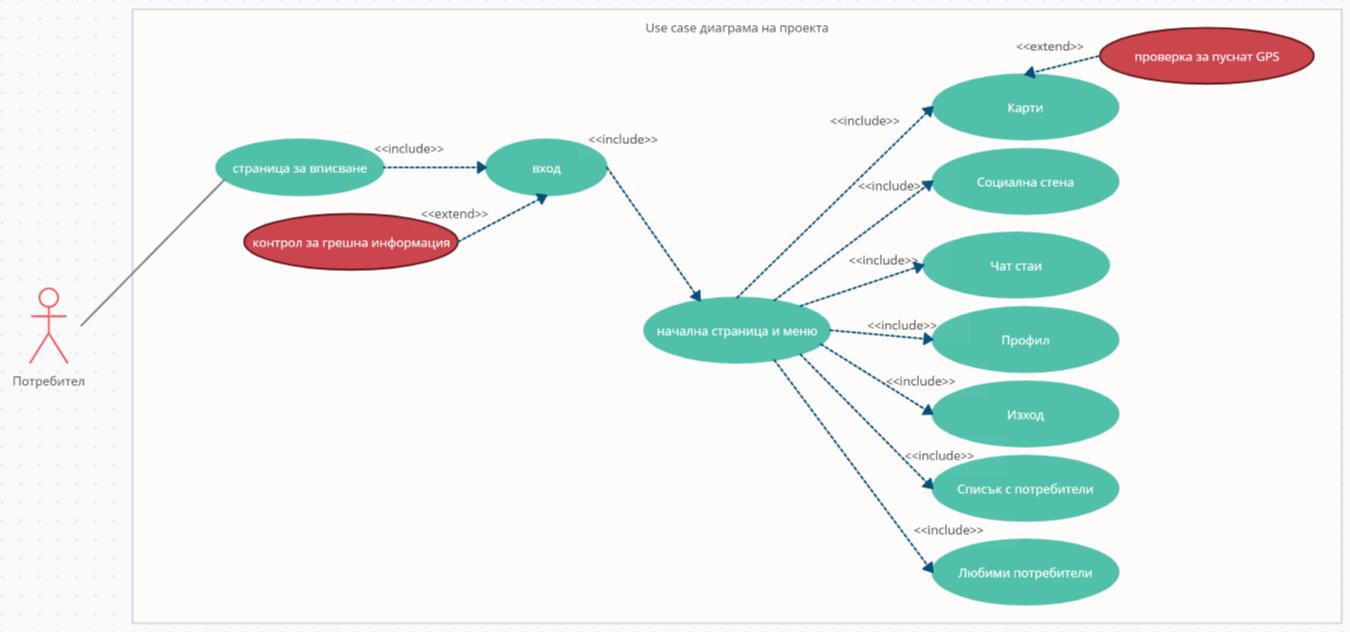
****

фигура 7. Съобщение за от Гугъл карти и неговото решение  [[10]](Solving_Google#_[10]_)

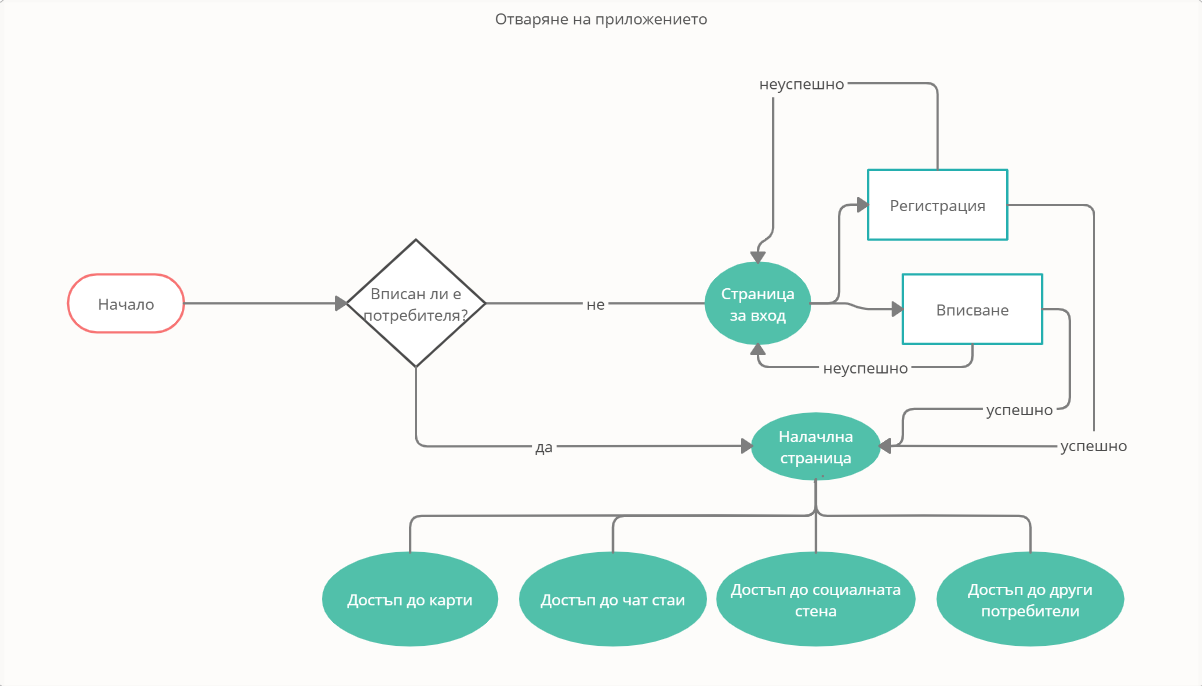
# Архитектура

## 4.1 Архитектура на приложението

Тук ще разгледаме архитектурата на мобилното приложение и услугите които ще предоставя на потребителите.



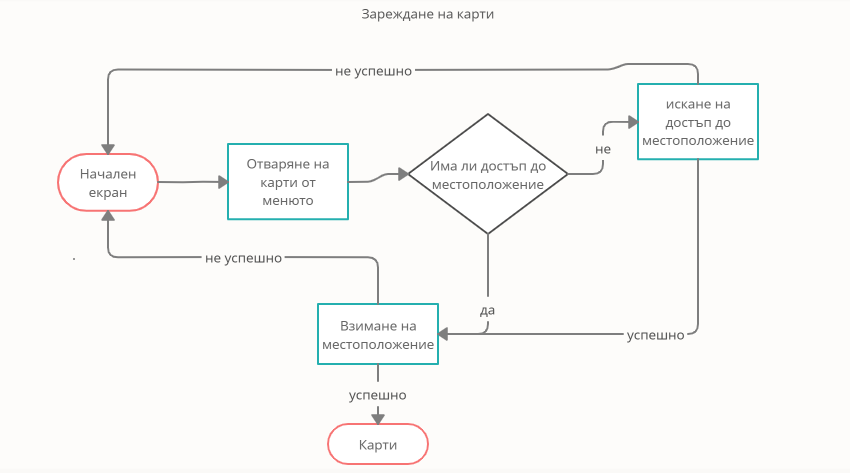
*фигура 8. Use case диаграма на проекта*



фигура 9. Блок-схема на зареждане на приложението

Всички потребители имат еднакви права и поради тази причина те могат да извършват едни и същи операции. Всички операции могат да се извършват само и единствено след вписване/оторизиране на потребителя. Без да се е вписал успешно, той няма достъп до нито един от екраните в приложението освен страницата на вписване.

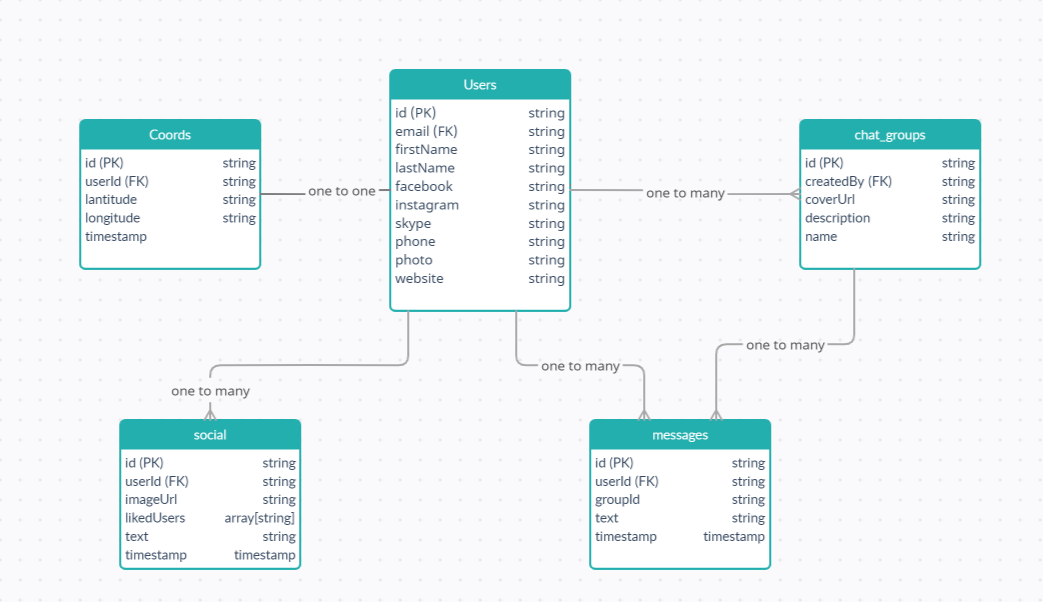
Когато потребител отвори приложението за първи път той ще види страницата за вход. На нея може да се впише с имейл и парола или да се регистрира с имейл, парола име и фамилия. При грешка той остава на този екран. При успешно вписване получава достъп до всички функции на апликацията и е изпратен на начална страница. Веднъж вписан в приложението, потребителя не е нужно да въвежда отново своите данни, а ще бъде пратен директно на начална страница. Тя е отправна точка към всички други страници.



фигура 10. Блок-схема на отваряне на карта с потребители

Блок схемата представя последователността при опит за отваряне на картата с потребители. Потребителя трябва да активира GPS на устройството си и да сподели своята локация. Ако условията не са изпълнени, той ще бъде помолен да стартира GPS и даде достъп на приложението до локацията. В противен случай той няма да има достъп до картите и ще остане на началната страница. Ако си е пуснал GPS и е дал достъп на приложението до локацията картите ще се отворят без да се изисква нищо от потребителя.

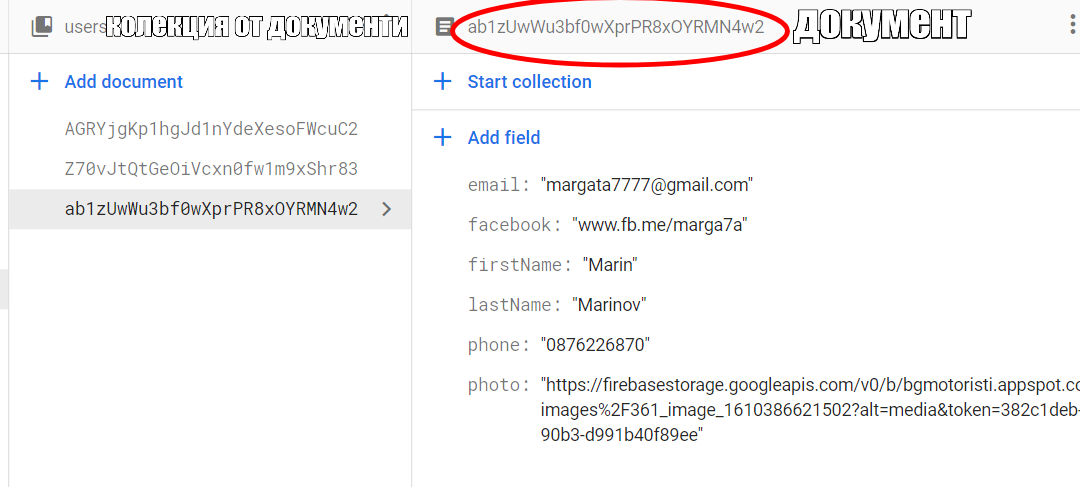
## 4.2 Архитектура на базата данни



фигура 11. Всички колекции и полетата за всеки документ

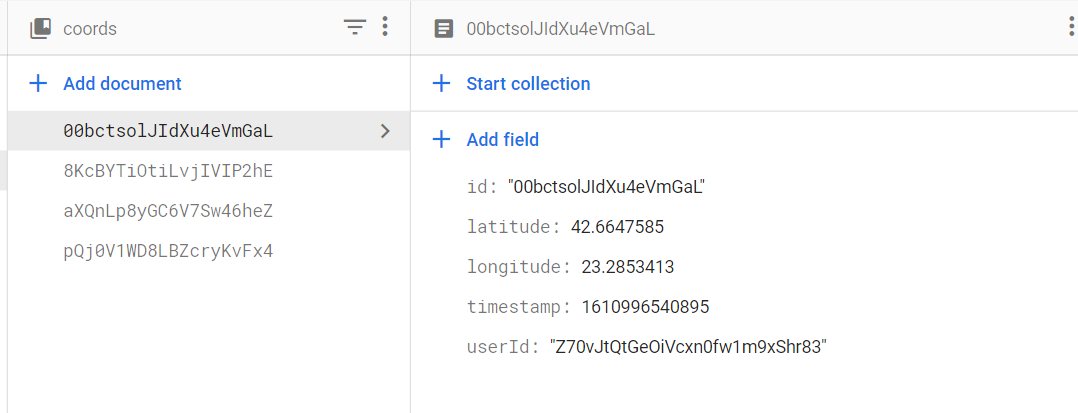
Базата данни е нерелационни и не съществуват физически връзки между отделните колекции. Идеята на този тип бази данни е информацията да се разбие на колкото може по малки парчета от данни (документи) като всеки документ държи в себе си информация за друг документ с който е свързан. Затова съм разбил всяка една страница от приложението в отделна колекция. Имаме една за потребителите (users) и 4 за картата, социалната стена, чат групите и съобщенията в тях, като общото между тях е, че всяка една има поле, което сочи към потребителя свързан с тях.

В “users” се пазят данни за всеки един потребител, неговия имейл(email), име(first\_name), фамилия(last\_name), както и ако е споделил социалните си мрежи и снимка (facebook, instagram,phone,website,skype, photoUrl).



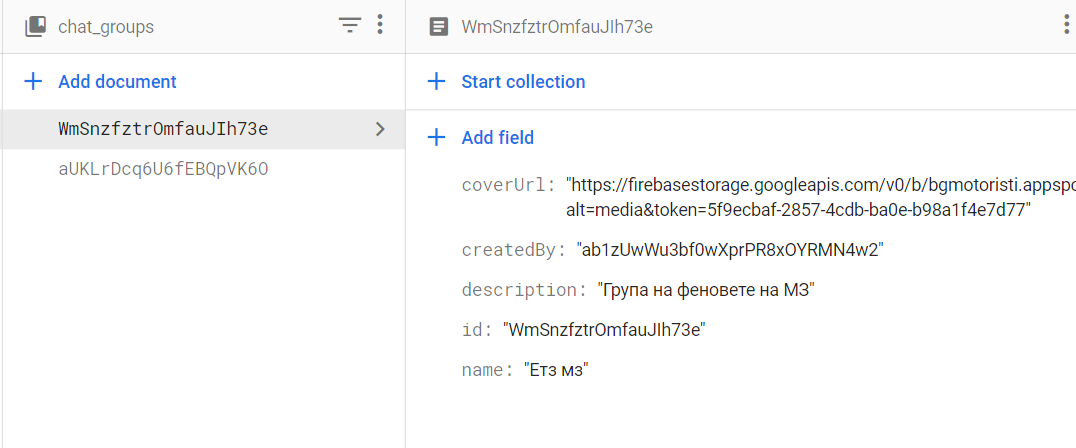
фигура 12. Колекция “users” и с три документа и полетата в един от тях

В coords -> id на документа , времето в което са взети координатите “timestamp”, самите координати ( lantitude ,longitude) както и “userId” - id-то на потребителя на чиито координати отговаря записа.



фигура 13. Колекция “coords” и с четири документа и полетата в един от тях

В chat\_groups имаме id на документа, корица на групата (coverUrl) , точно време кога е създадена (timestamp) и кратко описание за какво се отнася тя (description). Тук също имаме id на документа на потребителя (createdBy) създал чат стаята.



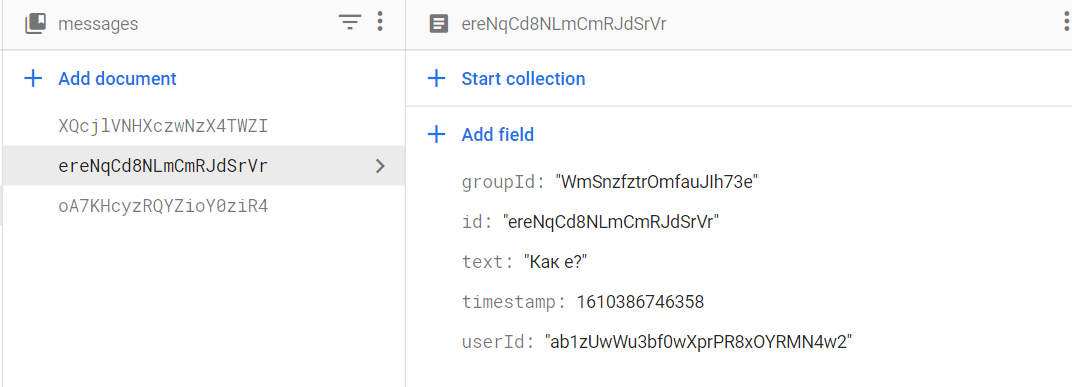
фигура 14. Колекция “chat\_groups” със два документа в себе си и полетата, които се съдържат в един документ.

В social ползваме “text” което е съдържанието на поста, “imageUrl” – снимката на поста, „timestamp“ – точно време кога е създаден поста, „likedUsers“ е масив от id-та на потребителите харесали поста и “userId” за потребителя създал поста.



фигура 15. Колекция “social” с три документа и полетата които се съдържат в един от документите.

В messages имаме следните полета: „id“ на съобщението, “userId” което показва ид-то на потребителя, който го е написал, “groupId” групата към която пренадлежи съобщението, “timestamp” – точно време кога е написано и “text” – съдържанието на съобщението



фигура 16. Колекция “messages” и документ на съобщение, който показва потребителя, групата на които принадлежи и неговото съдържание.

# Разработка

Основен език за писане на приложението е JavaScript, като е избран функционален стил пред обектно ориентиран. Всеки скрипт е вкаран в модул (клас), който представлява IIFE (Immediately invoked function expression), която скрива функциите и променливите вътре като създава вътрешен scope за всяка една функция на класа и не позволява на скриптовете да се презаписват или да си пречат. Така една функция или променлива от един клас, няма да презапише функция или променлива със същото име от друг клас. Направени така модулите споделят набор от методи към глобалния scope, но не може да се правят промени по тяхната функционалност от глобалния scope.

## Инициализиране на проект

За да стартиране проекта са ни нужни всички посочени технологии от точка 3.  
Започваме със създаването на начален проект чрез Phonegap CLI, отваряме терминал в желаната от нас директория („желателно е да е папка която се хоства от WinNMP“) и изпълняваме командата:



Това ще инициализира празен проект за нас със необходимите файлове за да може след това да пакетираме готовия продукт в инсталационен файл за съответната операционна система. След като успешно сме създали приложението стартираме WinNMP, отваряме браузъра и избираме директорията където се намира приложението.

## Инсталиране на приставки за достъп до локация и камера

За имаме достъп до камерата на устройството и неговото местоположение ни е нужно да инсталираме приставки. Тук използваме Cordova CLI, чрез следните команди изпълнени в директорията на проекта добавяме нужните ни приставки.

cordova plugin add cordova-plugin-camera

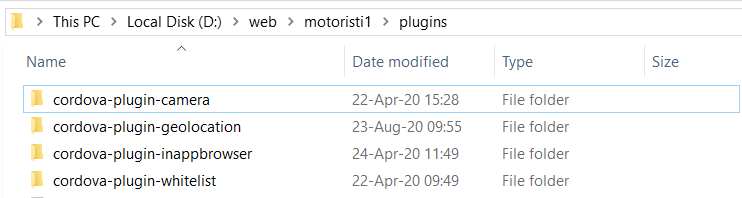
cordova plugin add cordova-plugin-geolocation

Нужни са ни още две приставки едната е задължнителна.

cordova plugin add cordova-plugin-whitelist

А другата за да може да отваряме страниците към социалните мрежи на потребителите без да затваряме приложението

cordova plugin add cordova-plugin-inappbrowser

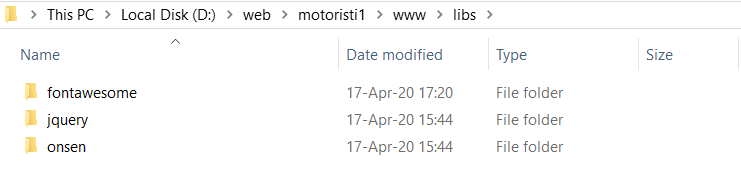


фигура 17. изглед от папка “plugins” след като сме добавили всички приставки успешно

## Дизайн на приложението и основни библиотеки

Нужни са ни само още няколко неща преди да започнем да наливаме основната логика в приложението. За дизайна използваме Onsen UI. Той е подходящ за екрани на телефони и таблети. В него има заложена функционалност за Single Page Application, т.е. да сменяме страниците в приложението без да се презареждат, която ще доразвием, за да работи в контекста на приложението. След Onsen UI, нашето приложение има нужда от иконки за тях използваме Fontawesome Icons. А за обработка на събития, анимации и работа с DOM дървото залагаме на jQuery.

Така добавяме нашите библиотеки в папка www (в нея се съдържат всички файлове, които не са част от Phonegap и после ще се пакетират. Нашите скриптове и стилове, библиотеки, иконки).



фигура 18. Изглед от папка “libs” след като сме добавили всички нужни библиотеки успешно

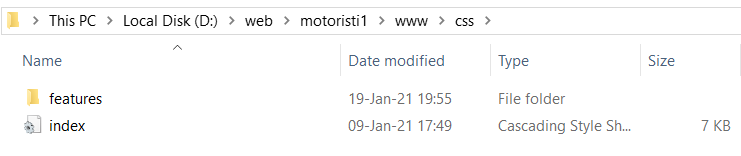
## Структуриране на проекта

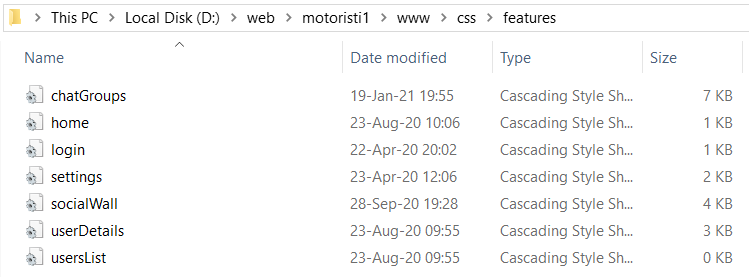
След като имаме основните библиотеки трябва да подготвим структурата на проекта. В нашия случай папките в папка “www” са разделени по следният начин:

www/css/ - в нея се съдържат стиловете на приложението, които сме написали

www/css/index.css – един файл, който държи глобалните стилове валидни за всички страници

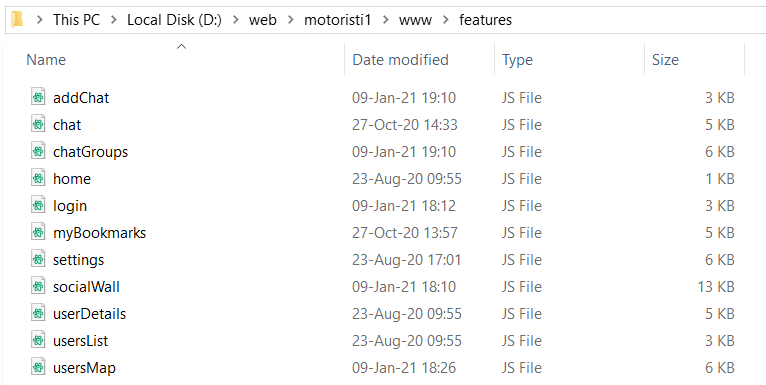
www/css/features/ - папка в която файловете са разделени по името на страницата, за която се отнасят например: www/css/features/home.css – тук стиловете са валидни само за началната страница.





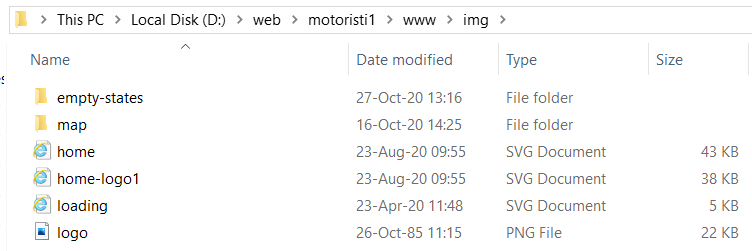
фигура 19 и 20. Структура на css файловете в приложението

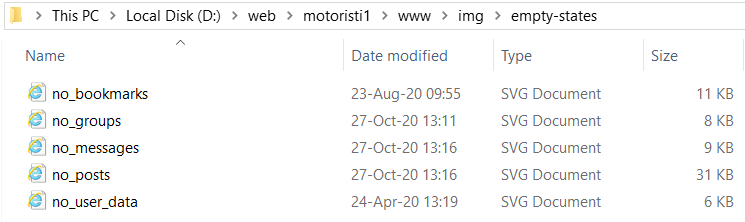
www/features/ - папка в която файловете са разделени по името на страницата. Тук всеки скрипт се отнася само за съответната страница например: www/features/home.js – този скриптов файл се отнася само за началната страница.

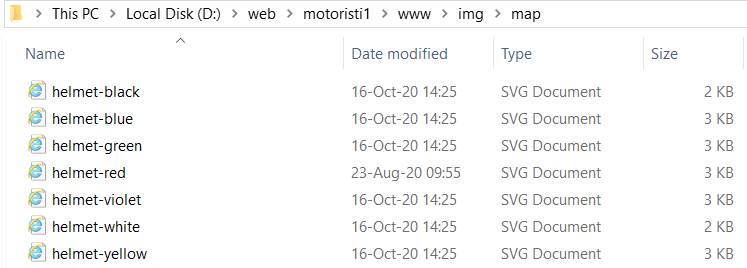


фигура 21. Всяка една страница в приложението си има скриптов файл

www/img – тук се намират всички графични изображения, които са част от апликацията.







фигура 22,23,24. показват как са разделени графичните елементи, които се използват в апликацията

www/js – тук са основните скриптове на приложението, те не са свързани само с един изглед

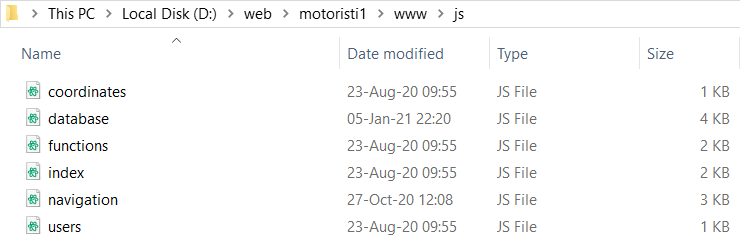
www/js/index.js отправна точка на апликацията от тук тя се стартира и инициализира базата данни

www/js/database.js – основен файл в нашето приложение, грижещ се за връзката между базата данни и приложението, дали е вписан потребителя или не (на база това апликацията ще поеме в различна посока към начална страница или страницата за вход) и синхронизацията на данните като потребителите и координатите на които се намират.

www/js/navigation.js – Файл грижещ се за навигирането в приложението, какво трябва да се случи преди да се смени екран, след като се смени, как да бъде заредена страницата (каква анимация да използва при смяната на екрана).

www/js/coords.js и www/js/user.js това са два файла с капсулована информация за координатите на всички потребители и информация за самите тях, тъй като тази информация се използва на различни екрани е изнесена тук при основните скриптове.

www/js/functions.js – споделени функции между всички файлове като валидиране на линк и конвертиране на изображения.



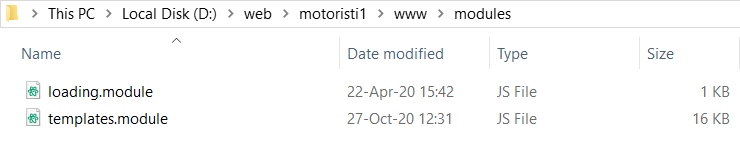
фигура 25. Основните скриптове нужни на приложението

www/libs/ - тук се намират библиотеките, които описахме в точка 5.3

www/modules/ - тук се намират скриптове, които манипулират екраните по някакъв начин:

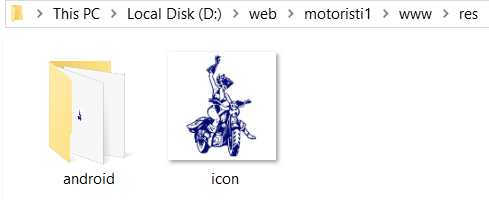
www/modules/loading.module.js – Той се грижи докато се зарежда приложението или тегли някакви данни, записва данни – да не може потребителя да използва другите функционалности.

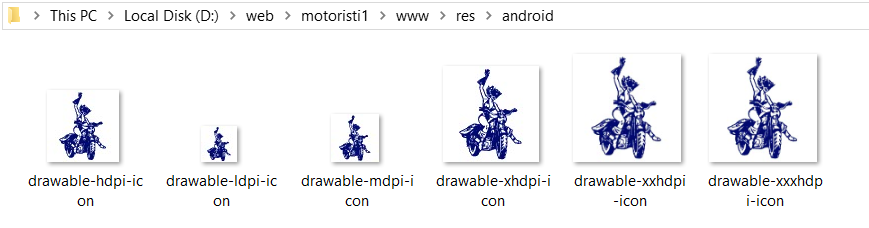
www/modules/templates.module.js – Той държи HTML темплейтите за всеки един екран и се грижи преди отварянето на всеки един от тях да го зареди в DOM дървото, но не и преди това, за да пести от ресурси.



фигура 26. Модулите които манипулират страниците на апликацията

www/res – Тук се намира основната иконка на приложението в размер 1024x1024, както и нарязаните иконки за андроид по спецификация с Adobe Illustrator.





фигура 27 и 28. Иконките нужни за приложението в съответния размер

* drawable-ldpi-icon.png размери 36х36
* drawable-mdpi-icon.png размери 48х48
* drawable-hdpi-icon.png размери 72х72
* drawable-xhdpi-icon размери 96x96
* drawable-xxhdpi-icon размери 144х144
* drawable-xxxhdpi-icon размери 192х192

www/services – Това са скриптове, които обработват по някакъв начин информацията, но не влияят пряко върху страниците и не могат да ги манипулират.

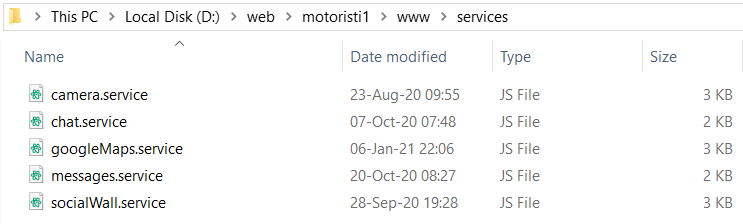
www/services/camera.service.js – Споделя методи за отваряне на камерата за снимка или отварянето на галерията, както и записването на снимката в буферната памет за обработка

www/services/chat.service.js – Споделя методи за осъществяване на постоянна връзка в реално време между приложението и чат стаите, както и запис на нова група в базата данни

www/services/googleMaps.service.js – Споделя метод за взимане на текущото местоположение на потребителя

www/services/messages.service.js - Споделя методи за осъществяване на постоянна връзка в реално време между приложението и съобщенията в избрана чат стая, както и запис на ново съобщение в базата данни

www/services.socialWall.service.js – Споделя методи за осъществяване на постоянна връзка в реално време между приложението и постовете в социалната стена , както и запис на нов пост в базата данни, харесване или не харесване на пост.



фигура 29. Всички services в приложението

# Ръководство за потребителя

## Системни изисквания и инсталация

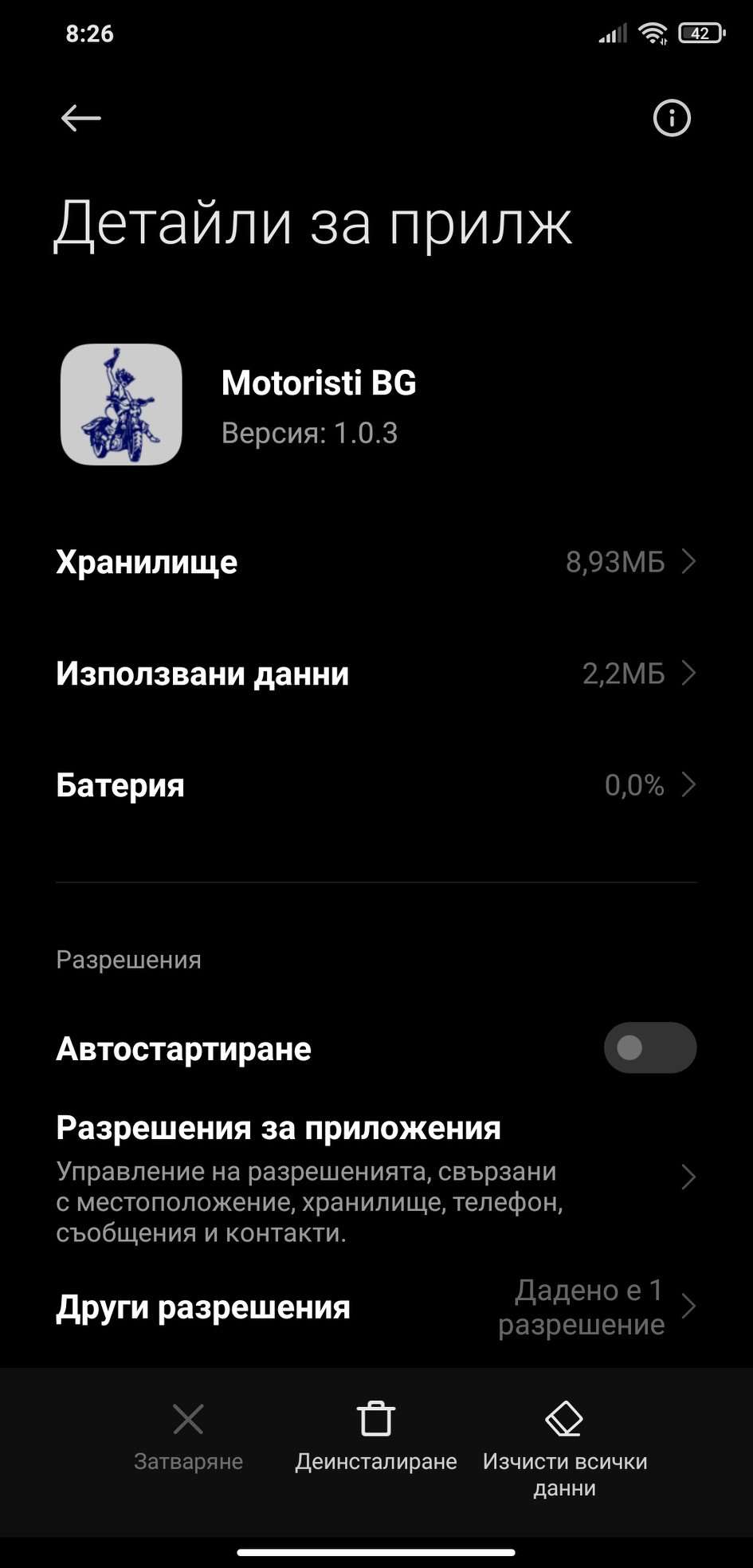
### Системни изисквания

* Постоянна връзка с интернет
* Андроид устройство с версия равна или по-висока от 6.0
* Разрешение за достъп и използване на „GPS“
* Разрешение за достъп и използване на „Камера“

### Инсталация:

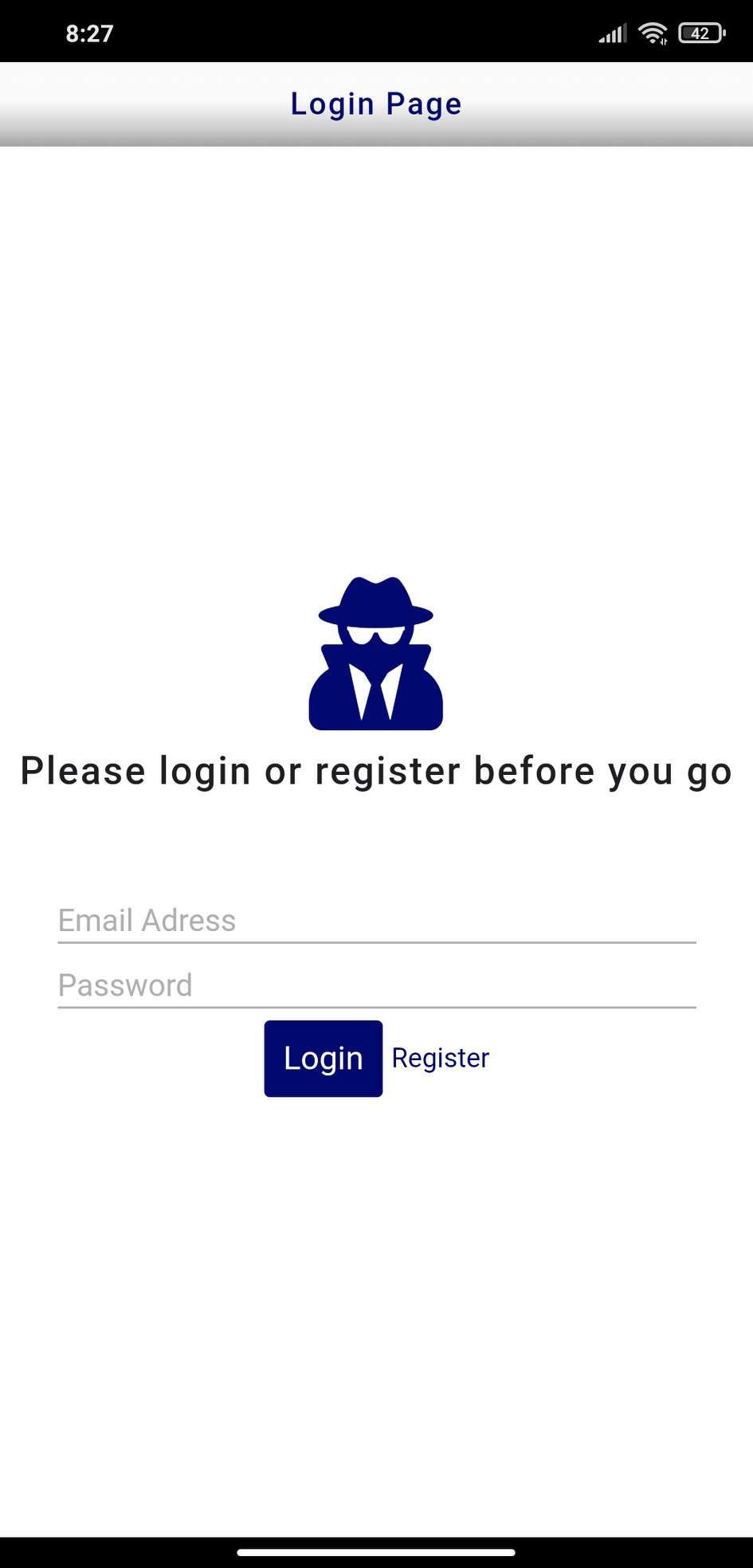
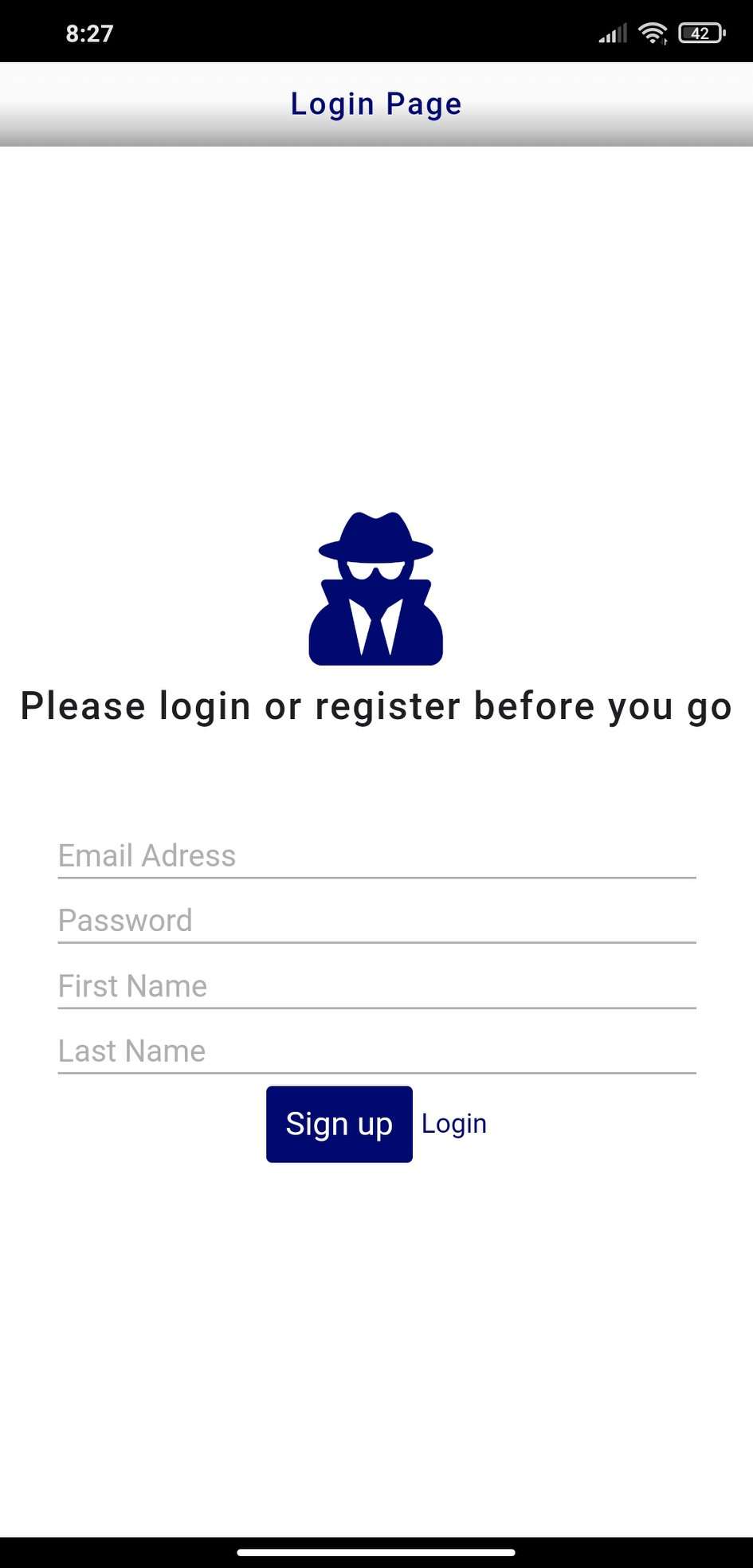
* Стартирайте инсталационния файл
* Следвайте инструкциите

След успешна инсталация в списъка с приложения трябва да виждате следната иконка:



фигура 30 и 31. показват как трябва да изглежда приложението след инсталация му

## Вход и регистрация

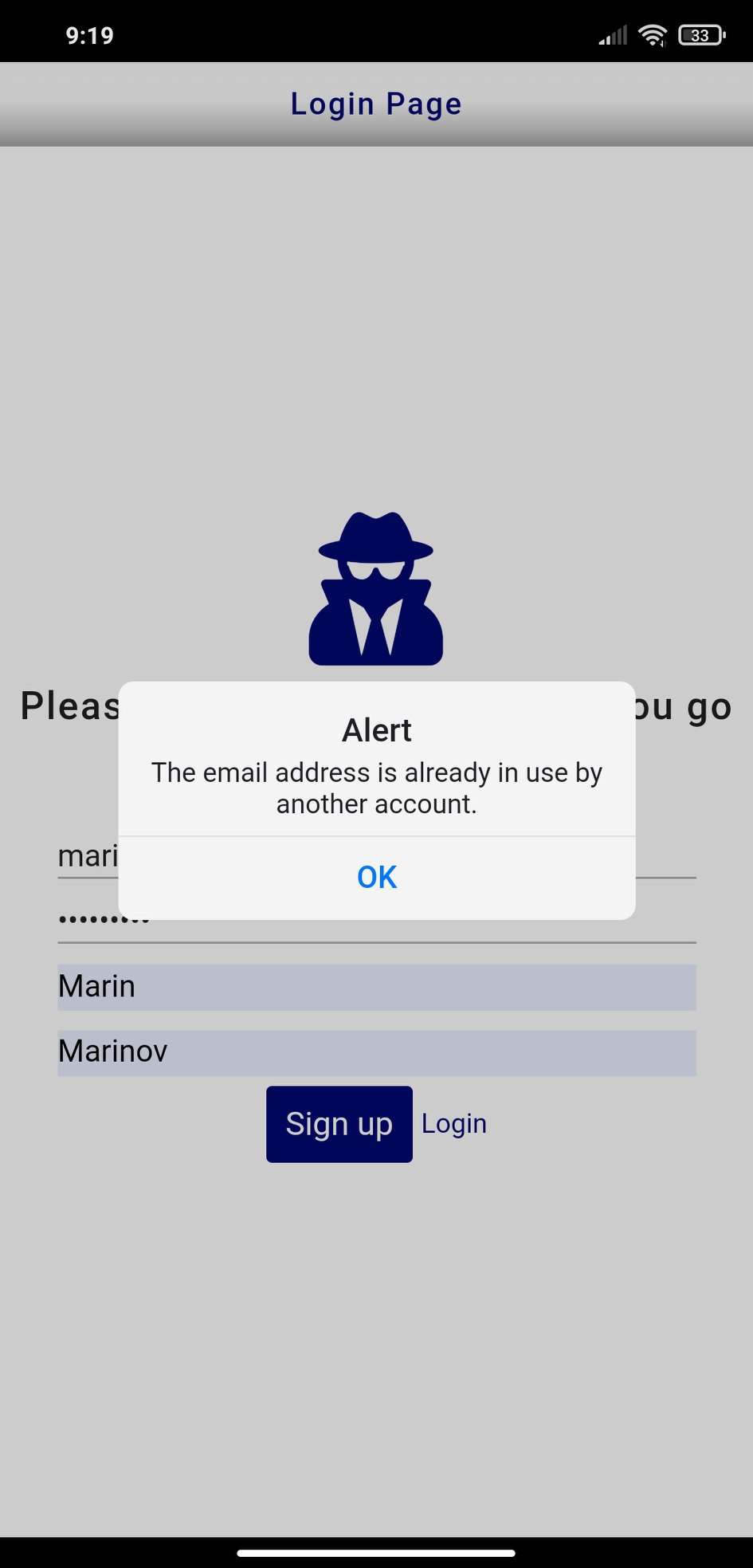
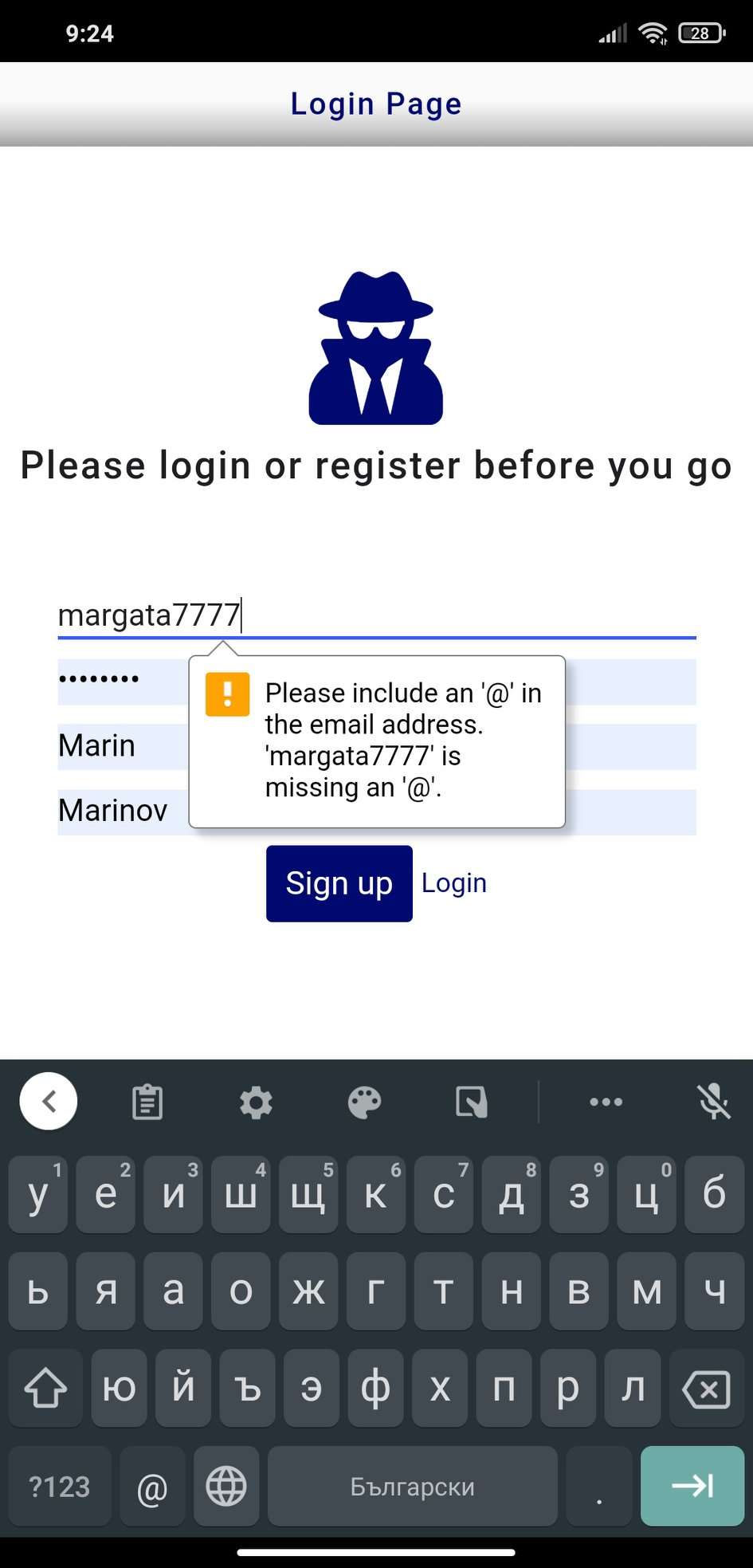
След успешна инсталация, проверете свързаността си с интернет и стартирайте приложението.

фигура 32 и 33. Страниците за вход (отляво) и за регистрация (отдясно)

При стартиране на приложението ще се зареди страницата за вход. На нея има 2 полета за имейл адрес и за парола. Както и два бутона за вход и за регистрация.

Тъй като нямаме все още акаунт избираме да се регистрираме чрез бутона “Register”, който ни зарежда страницата за регистрация (фигура 31). Попълваме полетата за имейл, парола, име и фамилия и натискаме бутона „Sign up“.

* При успешна регистрация приложението ще зареди началния екран
* При опит за регистрация с вече съществуващ имейл приложението ще изведе съобщение за грешка
* При опит за регистрация с невалиден имейл, приложението ще помоли за въвеждане на валиден такъв

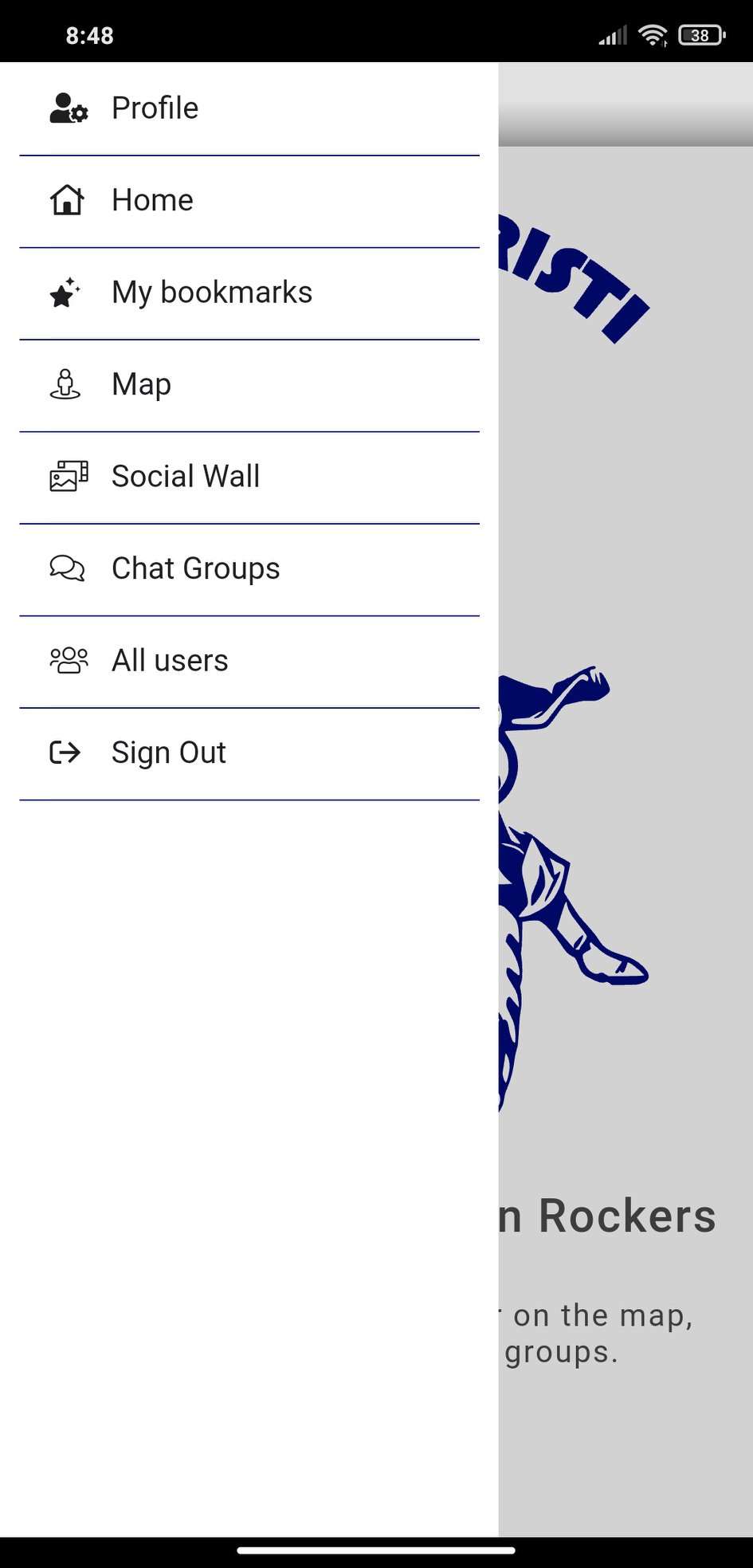


фигура 34 и 35. За непълен имейл адрес и имейл адрес използван вече от друг потребител приложението извежда грешка.

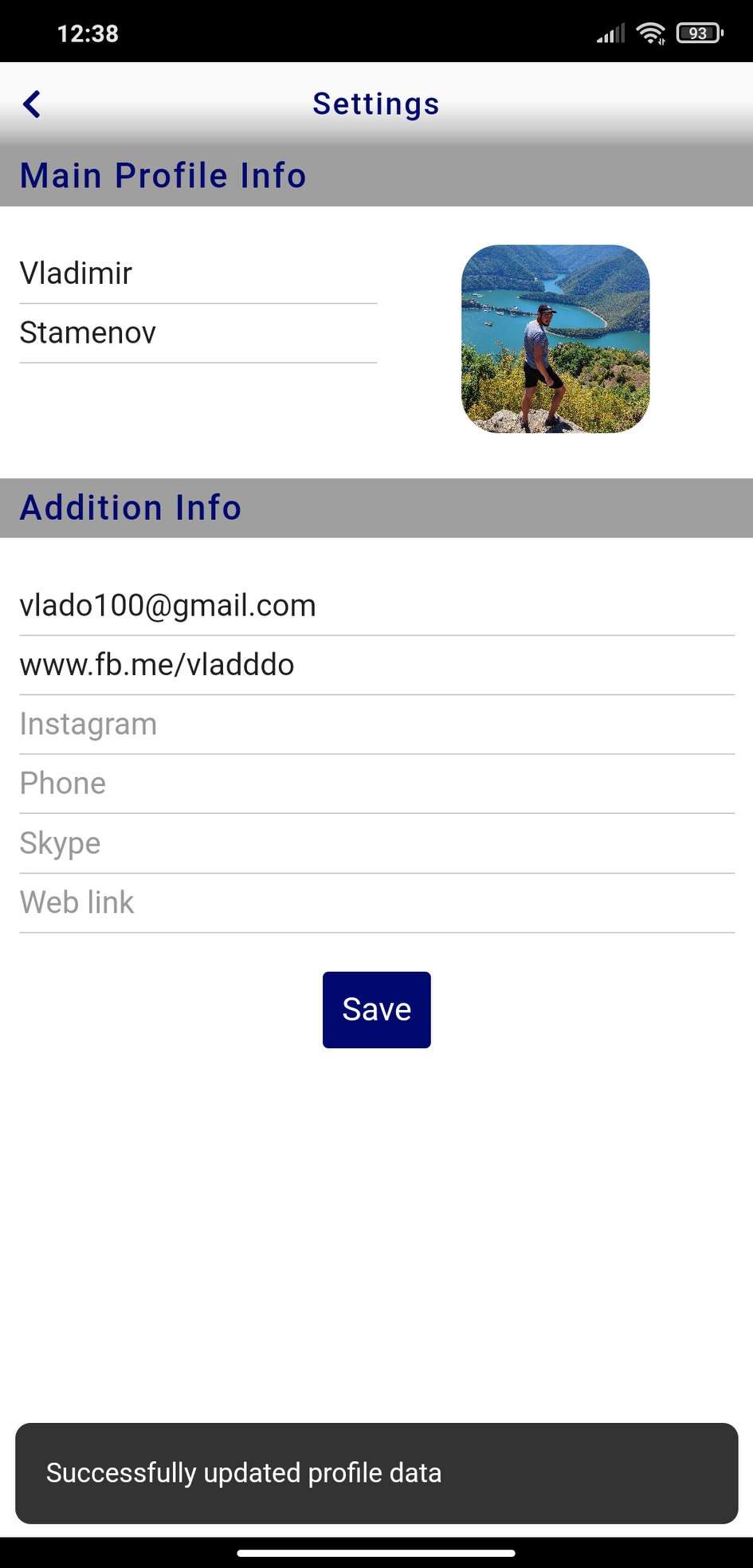
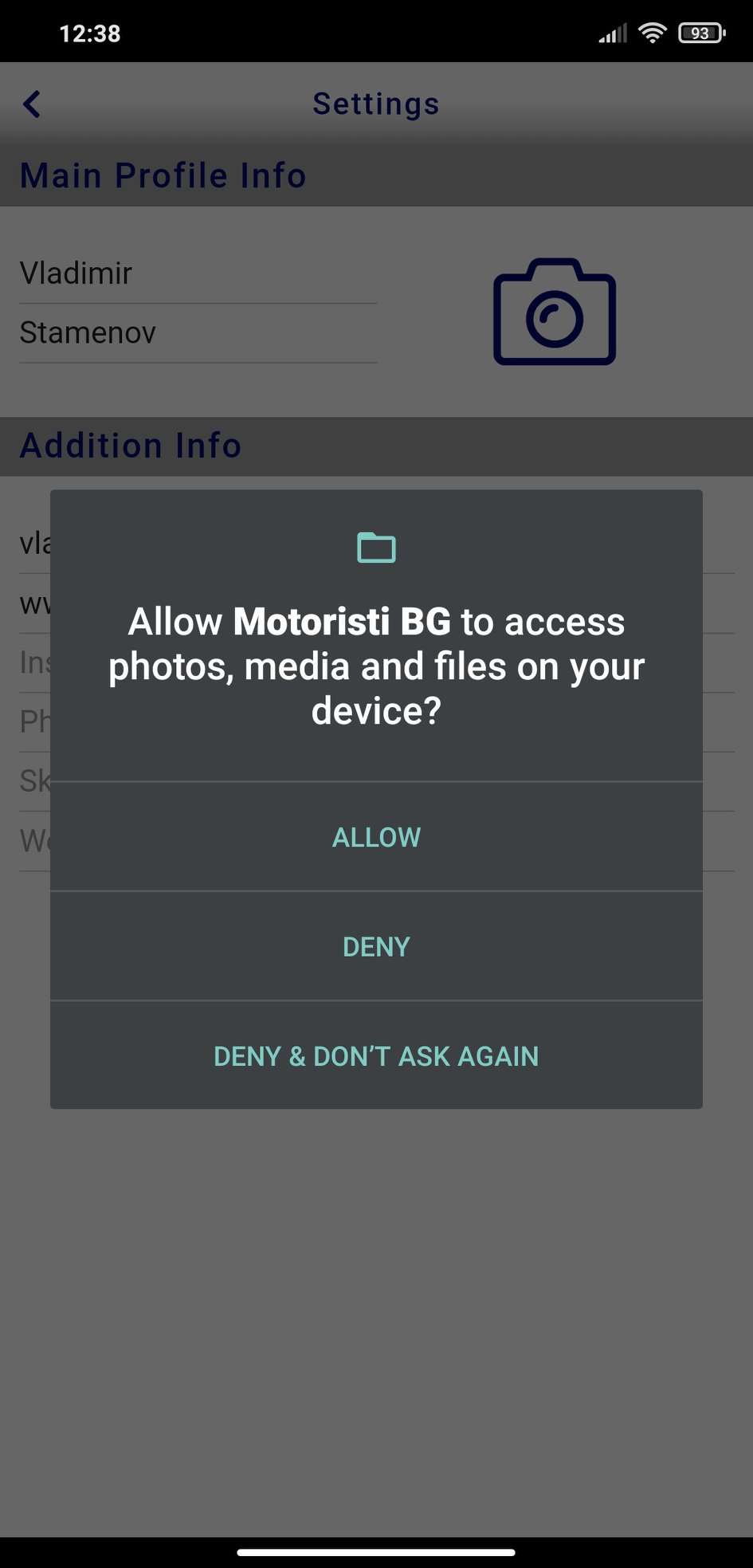
**Важно е да се отбележи, че имейл адреса с който се регистрираме, ще се вижда в профила ни, но може да бъде сменен или изтрит по-късно. Така той няма да бъде видим за останалите потребители, но си остава метод за вход в приложението.**

## Начален екран, основно меню и настройки на профила

След като сме се регистрирали успешно в приложението ще ни се покажа началния екран. На него в горната част вляво има бутон, който отваря основното меню. В средата е името на страницата на която се намираме. Менюто може да бъде отворено също чрез плъзване по екрана отляво надясно. И съответно затворено чрез плъзване по екрана отдясно наляво.

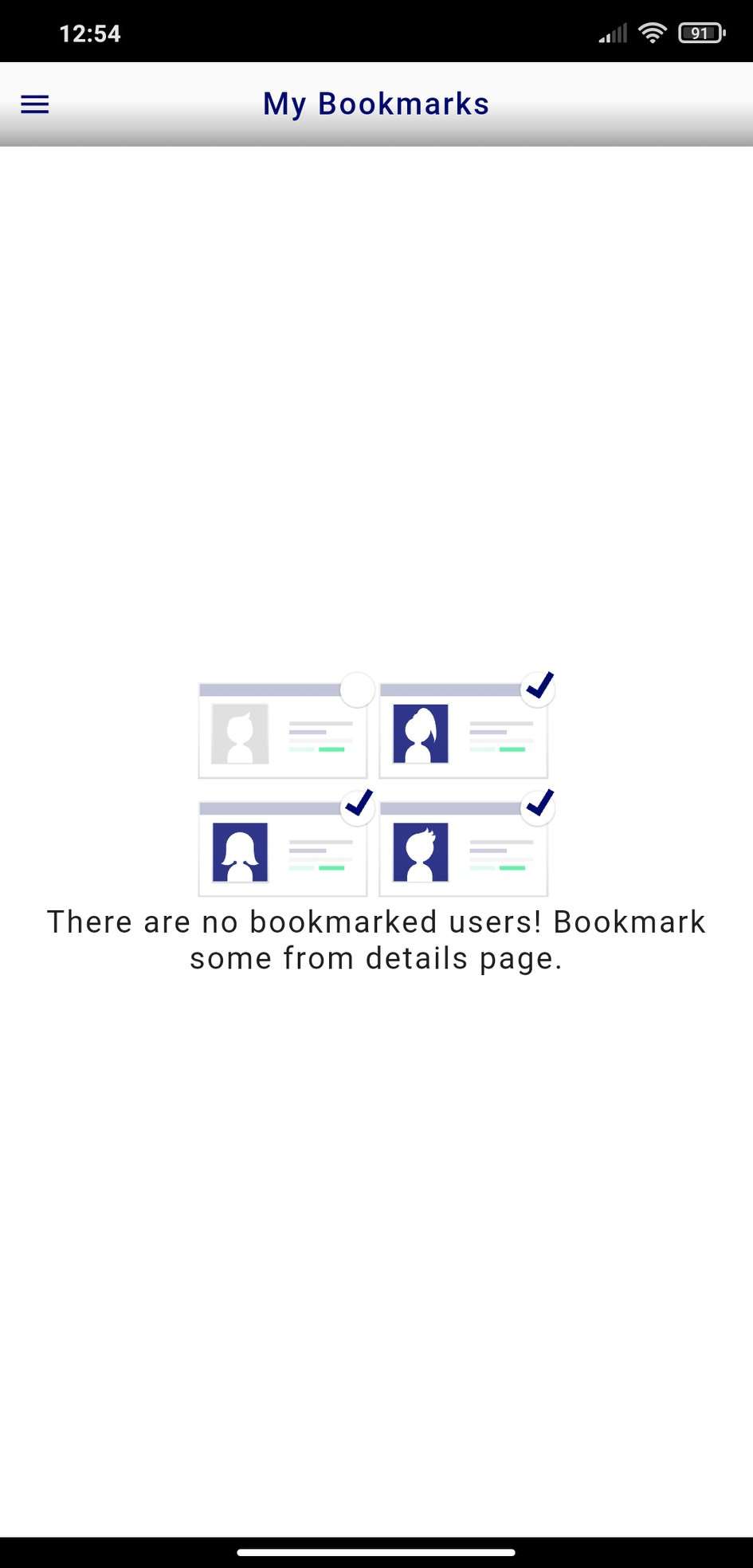
*фигура 36 и 37. При следващо отваряне приложението ще ни впише само и ще видим началния екран. С плъзване надясно по екрана се отваря менюто.*

След като сме отворили менюто избираме “Profile” за редакция на нашия профил. Тук може да редактираме името, фамилията, имейла, който е видим за другите потребители, както и останалата информация, която искаме да споделим.  
Чрез бутона с фотоапарат горе в дясно може да си качим профилна снимка. Тук приложението ще ни поиска достъп до камера и галерия. След като му разрешим имаме

две опции да качим снимка от галерията на устройството или да се отвори камерата и да се снимаме. След като сме избрали снимка – тя ще замени бутона с фотоапарата. Като завършим настройките на профила избираме бутона “Save” и чакаме операцията да приключи. Накрая ще изведе съобщение за успех. Важно е да се знае, че ако не се натисне “Save” всички редакции ще се загубят.

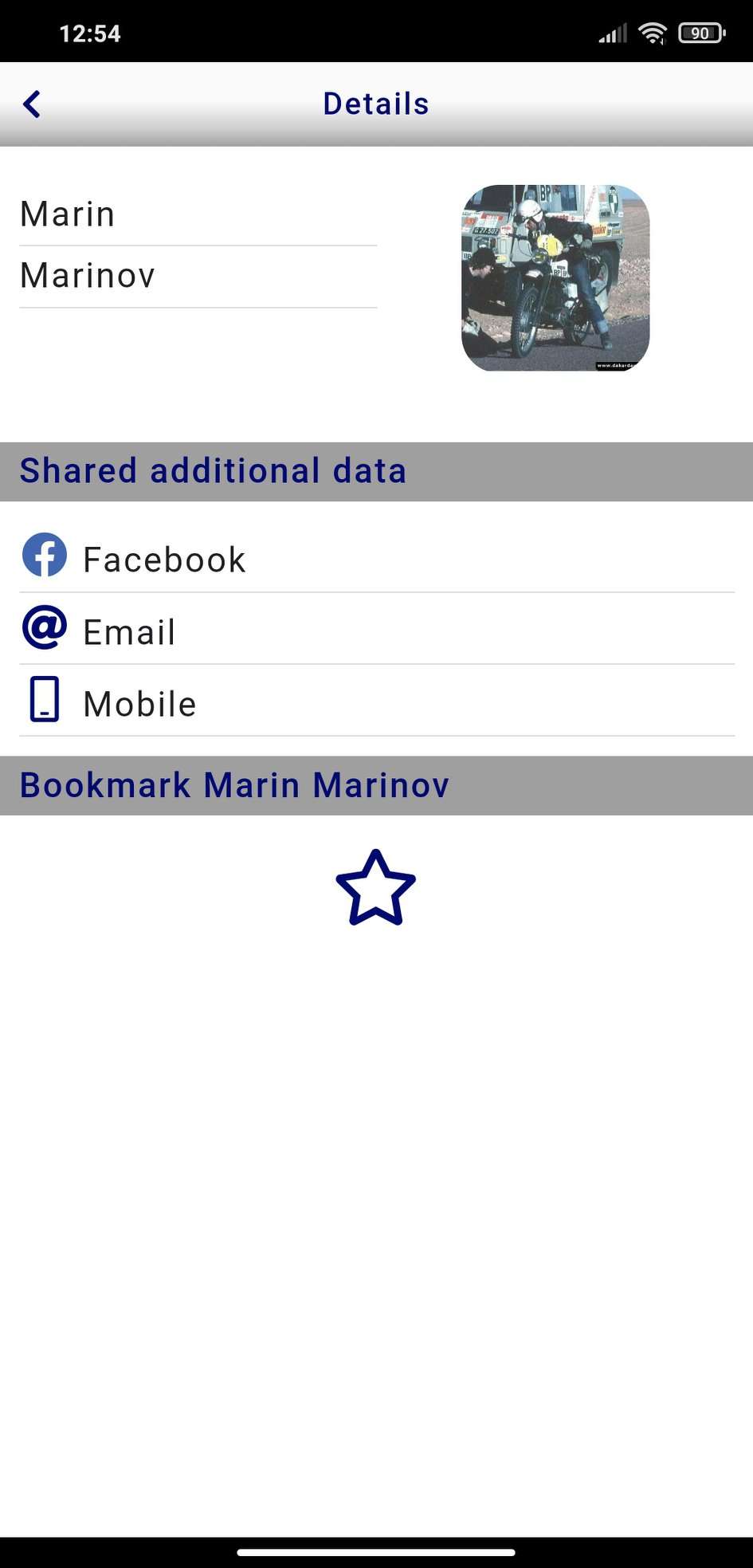
*фигура 38 и 39. Отляво потребителя е помолен да даде достъп до галерията на устройството. Отдясно успешно сме обновили профила си*

## Списък с потребители, „Любими потребители“, Допълнителна информация

Приложението ни дава възможност да видим всички регистрирани потребители и техните профили и как може да се свържем с тях (при условие, че са споделили такава информация). Също може да запазим потребители, за да може да ги откриваме по-бързо и лесно. Избираме от горе вляво бутона, който отваря менюто или плъзваме отляво надясно по екрана. След което избираме от менюто “My Bookmarks”. Ще ни се отвори страница, която казва че все още нямаме добавени потребители в „любими“.  
Отново отваряме менюто (по един от двата по-горе описани начини) и избираме “Users”. Тук в списък виждаме всички потребители регистрирани до момента.

*фигура 40 и 41. Отляво екрана без добавени потребители в любими. Отдясно списък със всички регистрирани потребители*

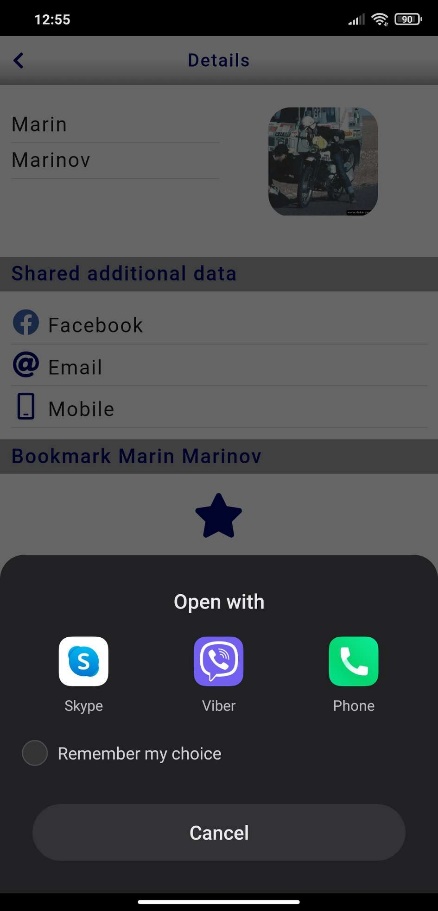
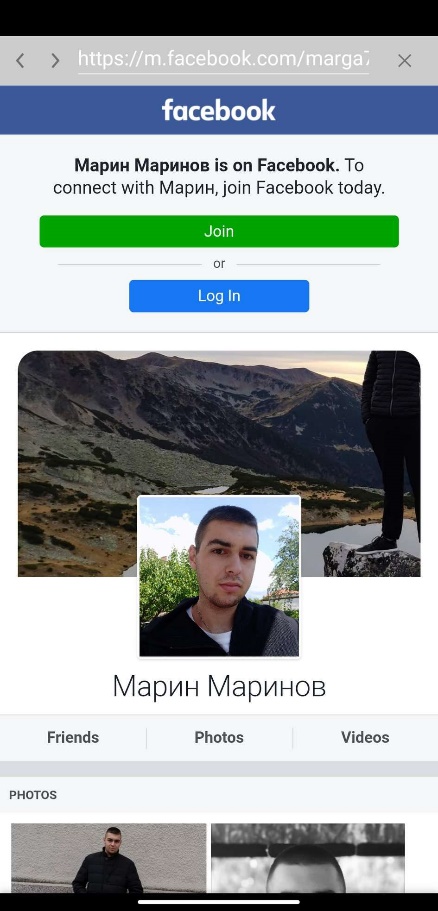
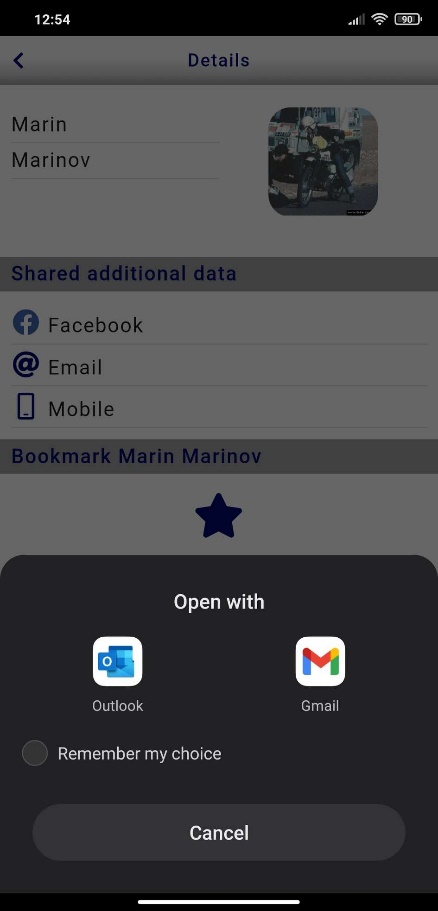
На новата страница виждаме списък с потребителите като те са разделени един от друг. В лявата част на всеки един има кратко представяне за самия него – снимка, име и фамилия. От дясно има бутон, който ни отвежда към страницата с допълнителна информация на избрания от нас потребител.



*фигура 42. Страница с детайлна информация за избран потребител*

Горе вляво може да видим, бутона за отваряне на менюто е заменен с бутон, който ни връща назад към списъка с потребители.

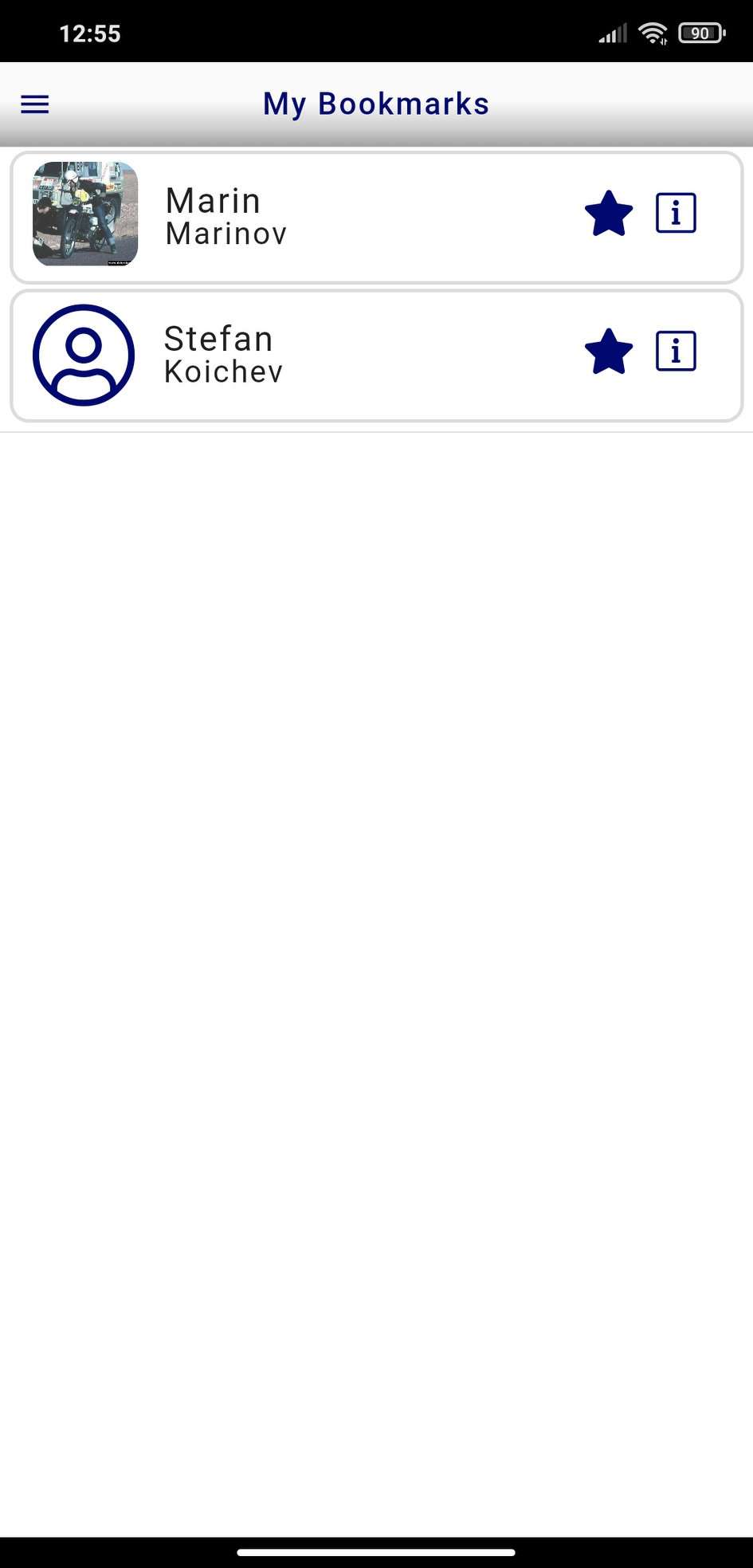
Под секцията с основна информация като име, фамилия и снимка за избрания от нас потребител, имаме секция с допълнителна информация за него. В случая “Facebook”, имейл адреса му и телефонния му номер. Всяко едно от тези полета представлява бутон с различни функции:

* Facebook – ще ви отвори сайта в браузър от приложението, който има бутон за затваряне. Той ще ви върне отново на страницата с допълнителна информация.
* Email – ще се отвори с приложението за подразбиране за отваряне на имейли, ако има няколко, ще ви попита кое да избере (Gmail, Outlook, Вграденото в устройството приложение за имейли и други). Отваряйки приложението за имейли, BG Motoristi остава да работи да работи на заден план.
* Телефонен номер – приложението ще отвори указателя ако е избран по подразбиране или ще ви попита, кое приложение да използва, ако имате други такива (Viber, Skype, WhatsApp…). Отваряйки приложението за извършване на обаждания BG Motoristi остава да работи на заден план.

*фигура 43, 44, 45. Избиране на бутони от секцията с допълнителна информация*

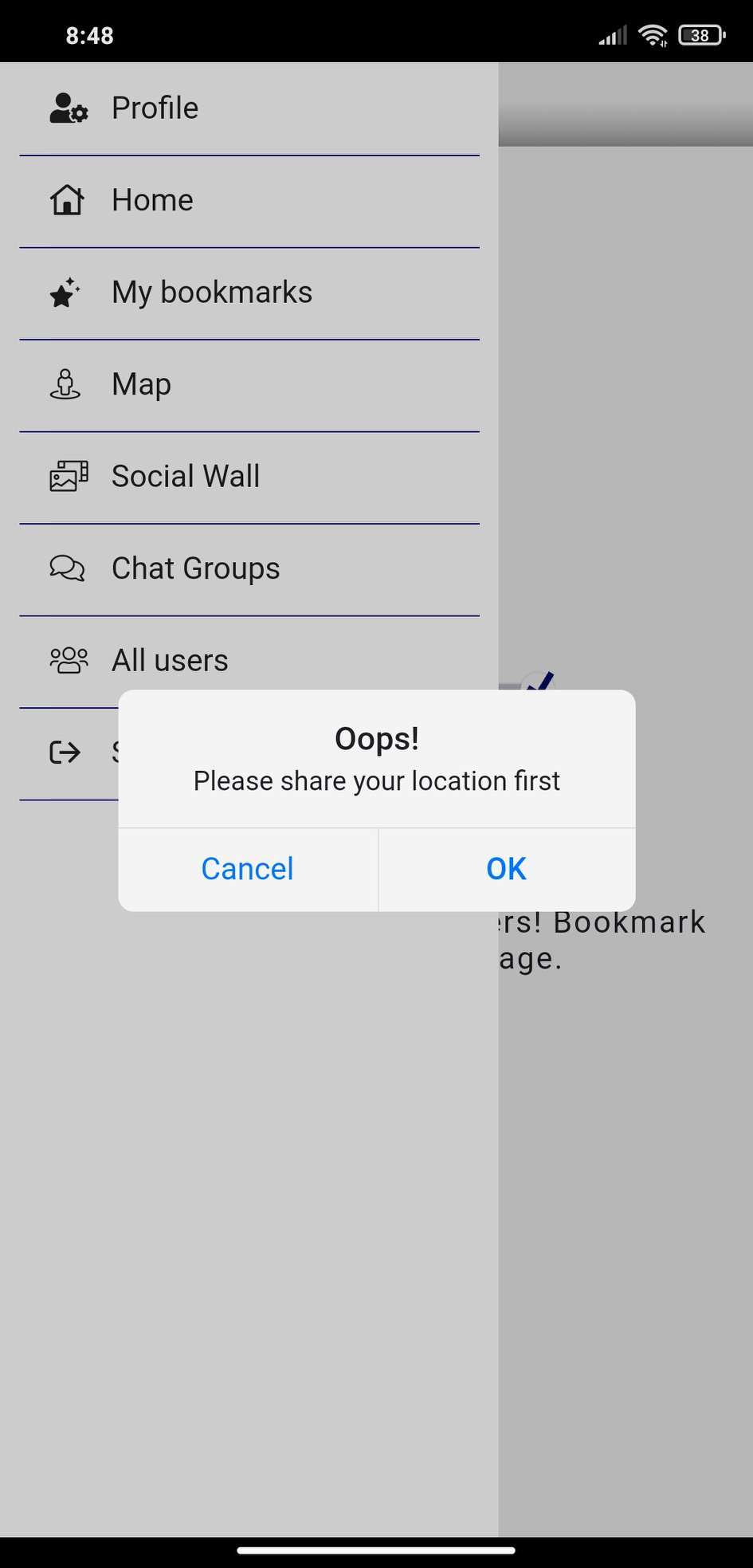
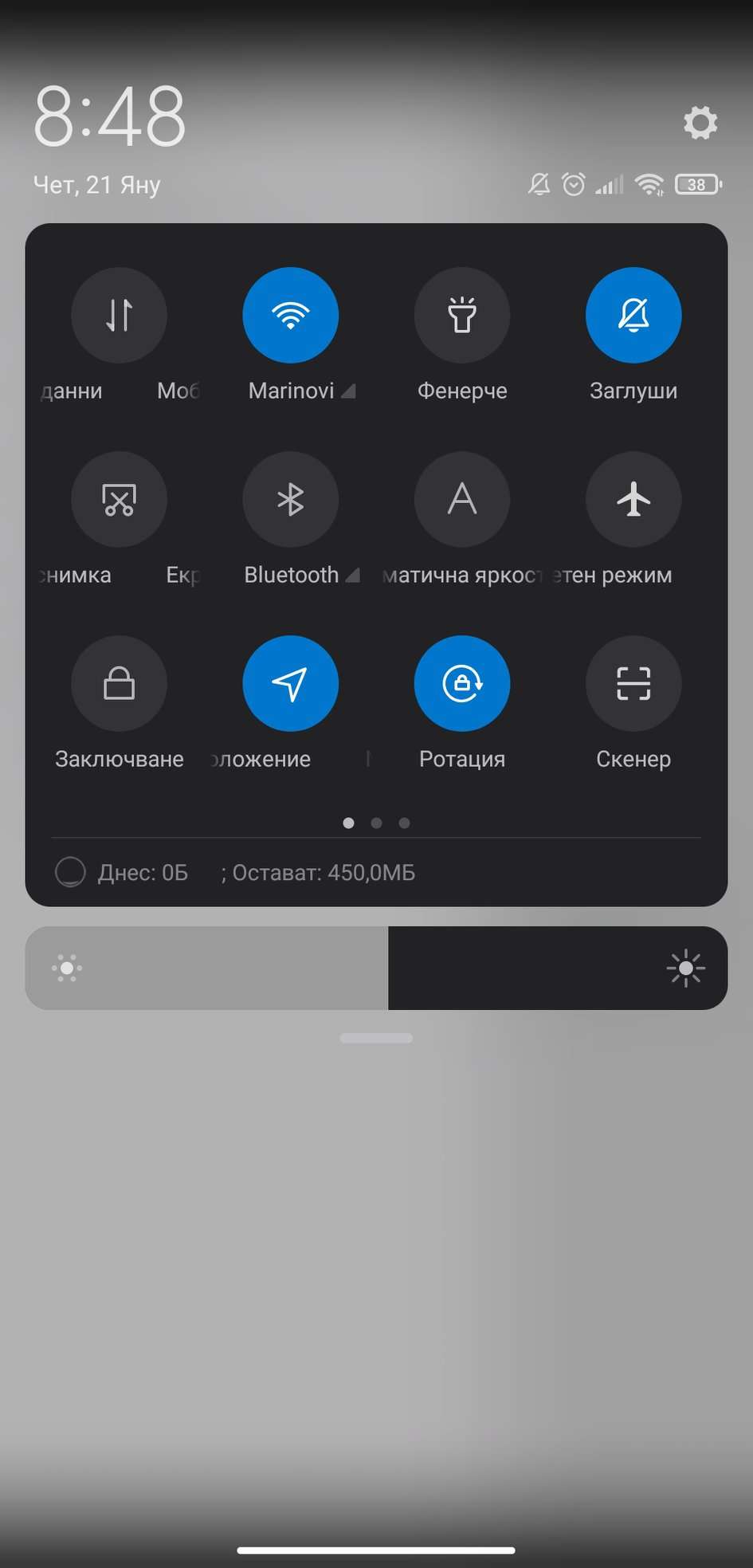
**забележка: Поведението за бутоните „Instagram“ и „уебсайт“ ще бъде същото като при “Facebook”.**

Най-отдолу има бутон под формата на звезда. С него добавяме или премахваме потребител от любими. Когато звездата е пълна – потребителя е добавен в списъка ни с любими, когато е празна – той не е част от списъка ни с любими потребители. След като добавим потребител в любими можем да се върнем назад към основния списък с бутона назад. От менюто да изберем “Bookmarks” и там ще видим списъка с любими потребители.



*фигура 46. Списък с любими потребители*

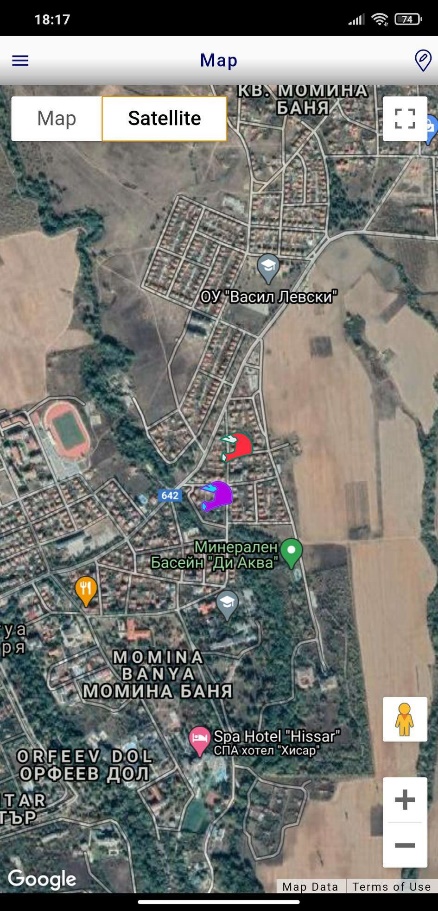
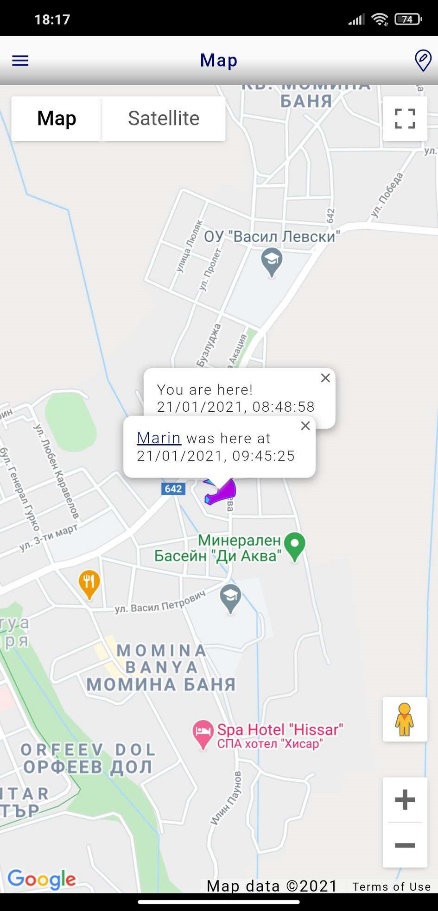
## Карти

За да се заредят картите първо трябва да сме да дали достъп до местоположението на устройството и да имаме пуснат GPS в противен случай, приложението ще изкара съобщение за грешка:

*фигура 47,48. Отляво съобщение за грешка. Отдясно включване на GPS на устройството.*

След като изпълнено всичко описано по-горе при избиране на “Maps” от менюто, се отваря карта, която е центрирана в нашето местоположение. Мястото на което се намираме е маркирано с червена каска. Местоположението на другите потребители също са маркирани с каски на картата. При клик върху каската се отваря малък прозорец със съобщение с информация. Информацията съдържа името и линк на потребителя, на когото принадлежи, както и точна дата и час от момента когато той е бил на това място. Ако кликнем върху името – то ще ни отведе към страницата с допълнителна информация за неговия профил, разгледана в предишната точка (6.4). Горе вдясно има бутон с иконка на маркер. С него може да обновим картата.

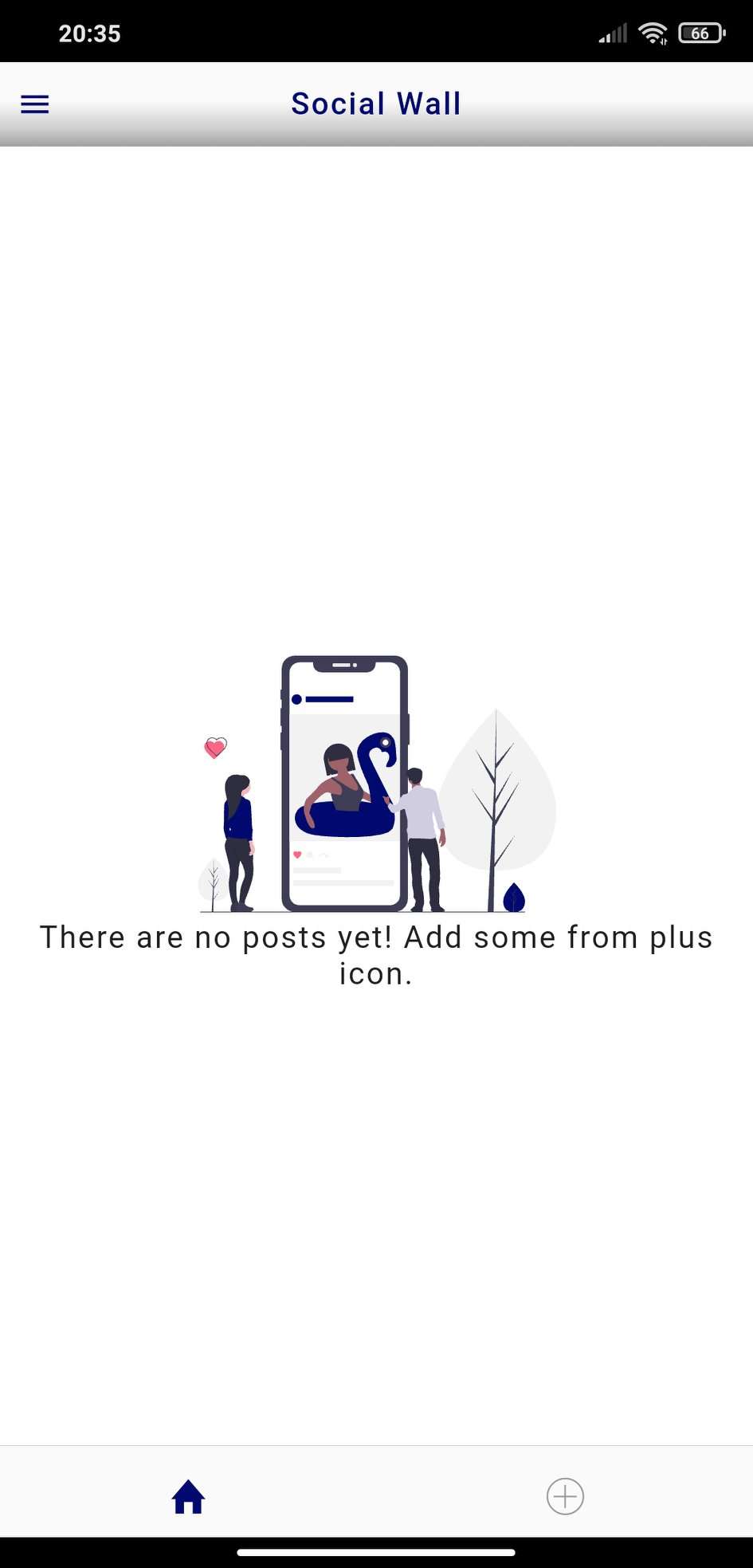
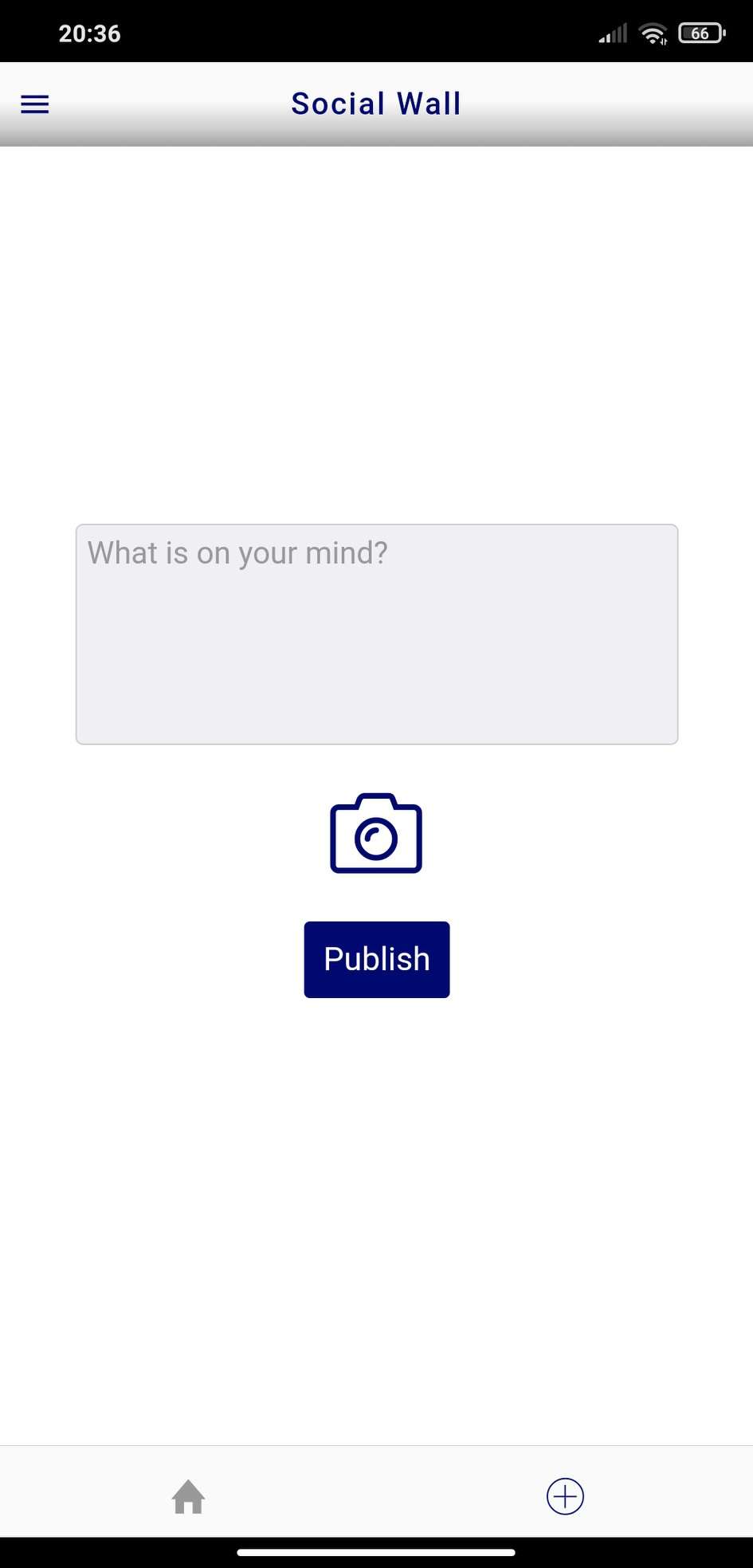
**забележка: бутона няма да работи, ако GPS е изключен.**

Други функции можем да използваме са вградените за „Гугъл карти“, а именно промяна на изгледа (избор между „Терен“ и „Сателит“), маркери със всички обекти намиращи се наоколо (магазини, бензиностанции, ресторанти и др.) както и „Google Street View“, който ни дава възможност за разузнаем на около.

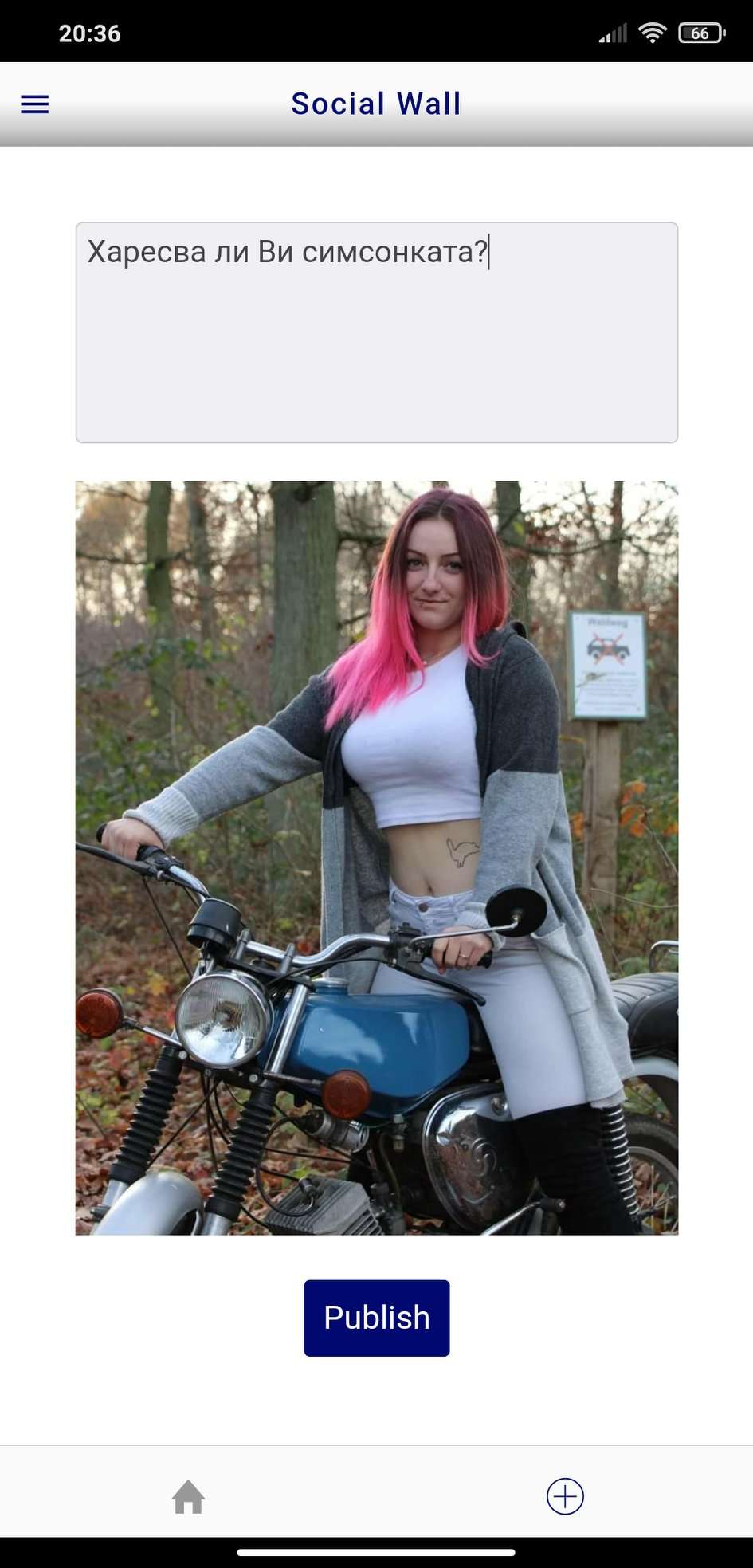
*фигура 49, 50, 51. Различни изгледи от картата.*

## Социална стена

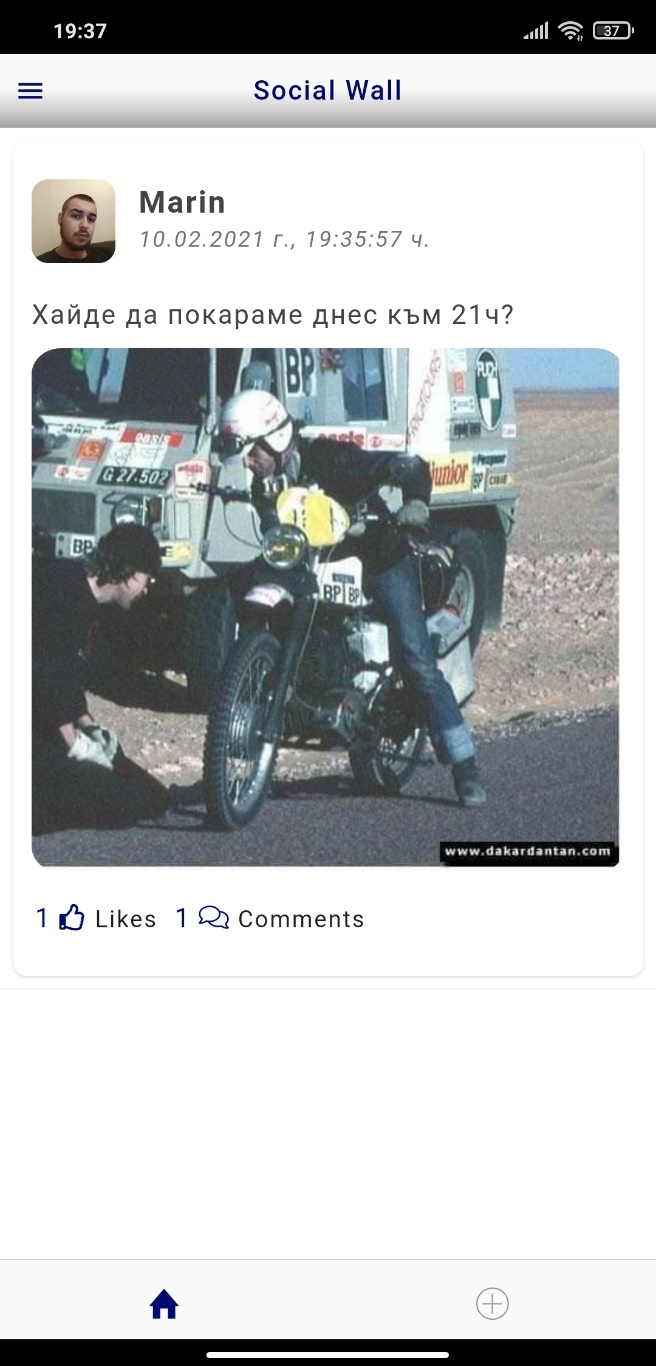
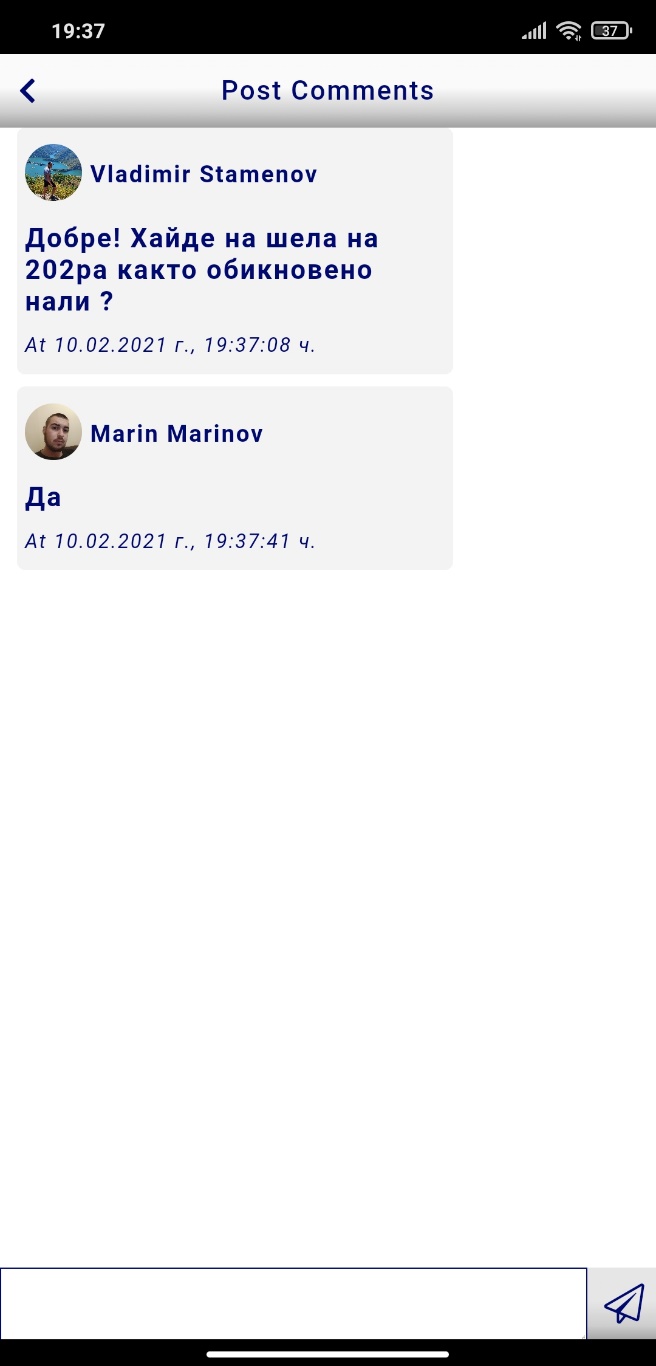
Социалната стена е мястото където потребителите могат да споделят постове със снимка или без. Избираме от менюто “Social wall”. На отворената страница виждаме съобщение, което ни казва, че все още няма качени постове. Най-отдолу има 2 бутона. Левият бутон показва списъка с постове, който в момента е празен, а десният за добавяне на нов пост.

*фигура 52, 53. Отляво екрана при отваряне на „Социалната стена“. Отдясно екрана за добавяне на нов пост.*

На втората снимка виждаме екрана за качване на постове. В текстовото поле може да напишем съобщението, което искаме да споделим. Ако искаме да качим снимка избираме бутона с фото апарат и избираме от галерията на устройството или от камерата. С бутона “Publish” споделяме поста. При успешно споделяне, приложението ни връща към списъка с постове, където виждаме и нашия пост.

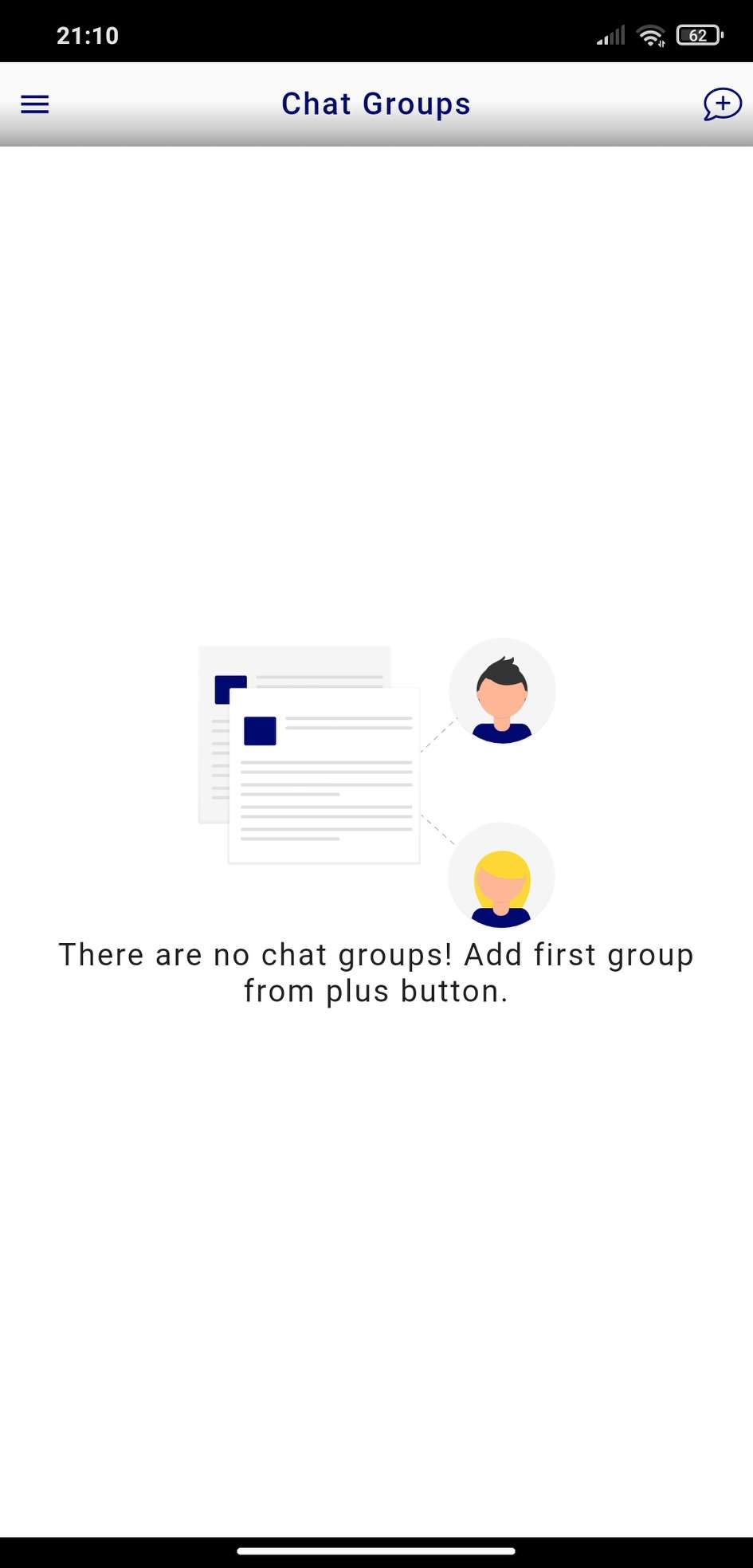
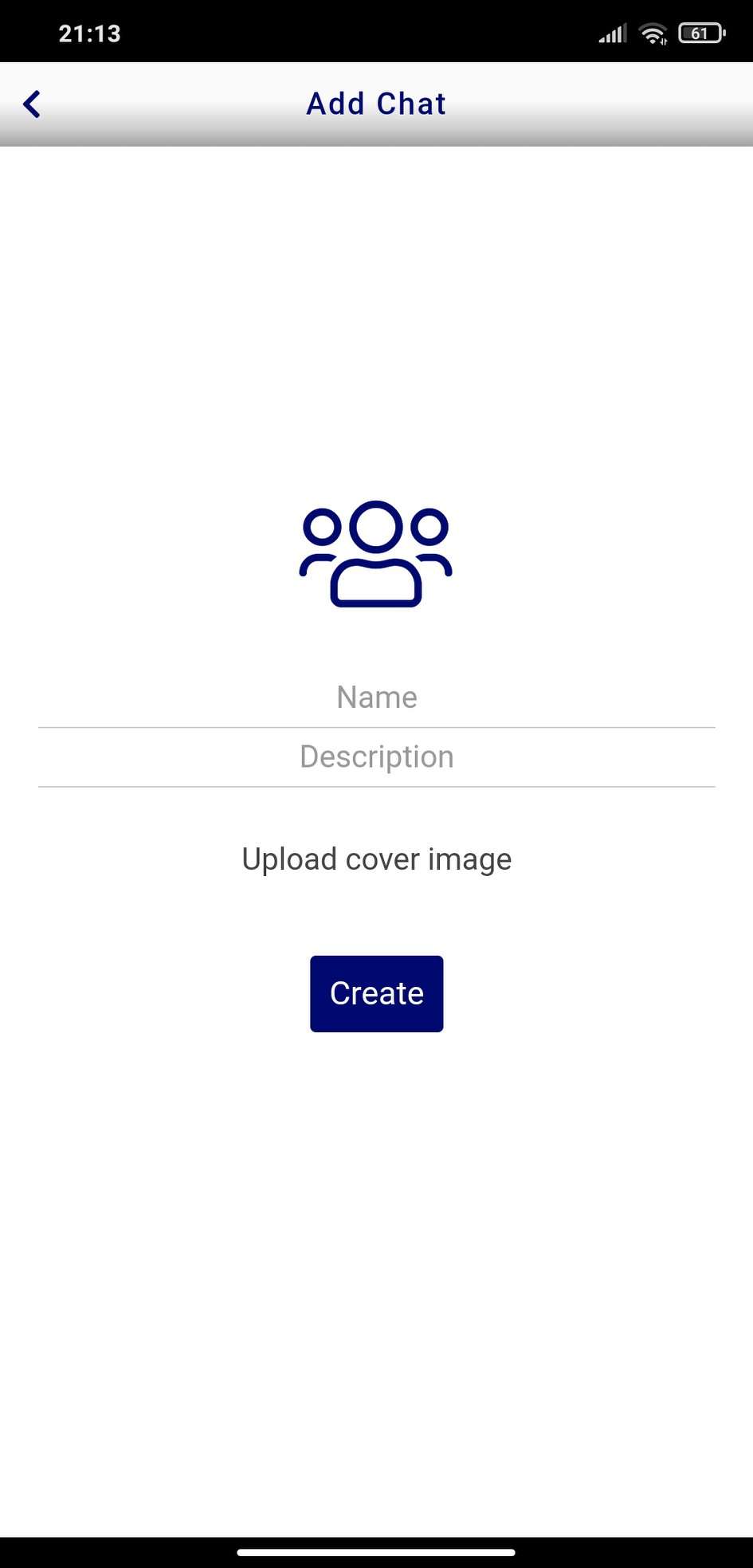
***фигура 54, 55. Отляво преди да качим поста. Отдясно вече качения от нас пост на „Социалната стена“*

Всеки един пост може да бъде харесан и коментиран от другите потребители. Те се подредени по дата на споделяне, т.е. най-новите ще бъдат най-отгоре. Ако нов пост бъде добавен докато разглеждаме стената с постове, бутон с камбанка ще се появи долу вдясно. При натискането му постовете ще се обновят и новите постове ще бъдат отгоре.

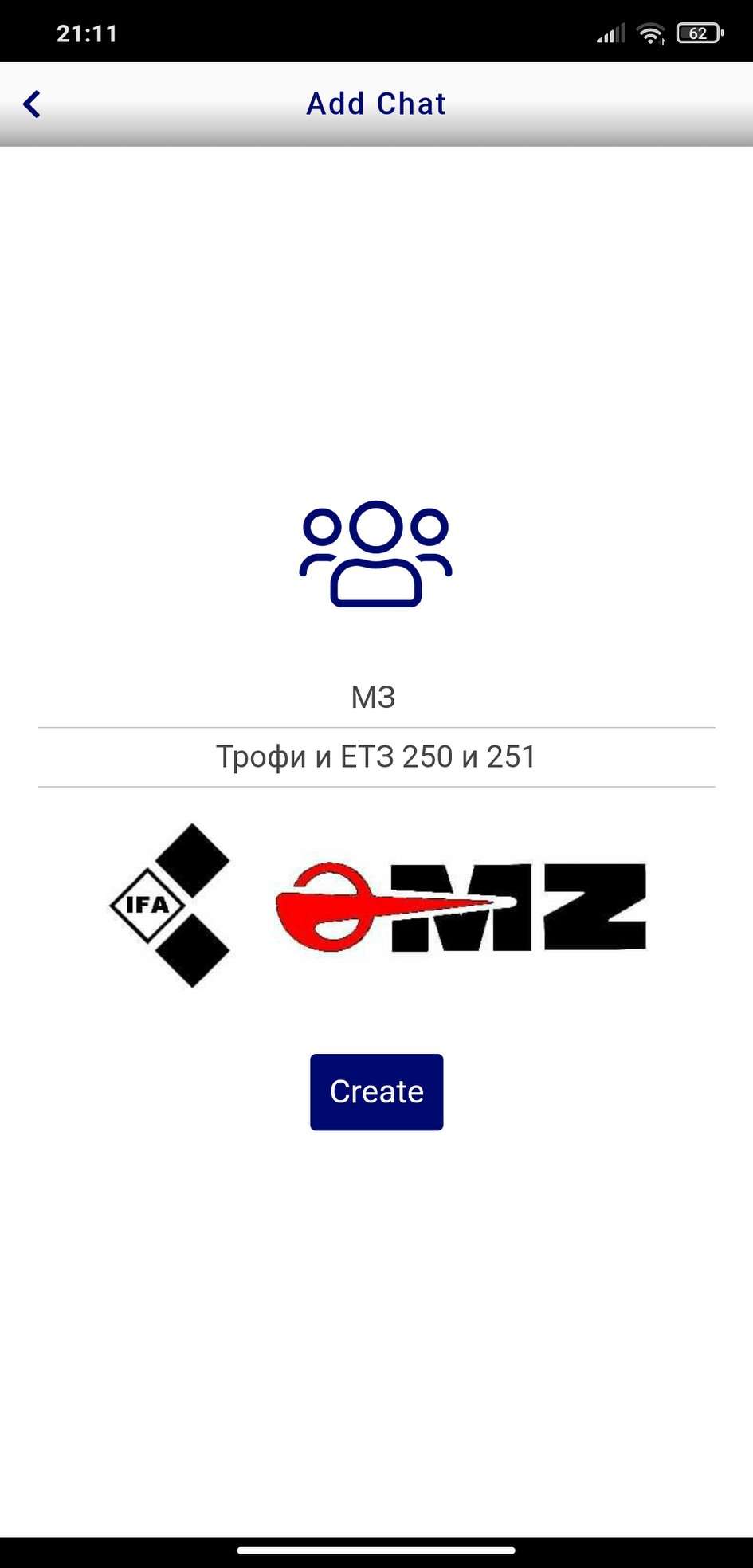
*фигура 56, 57. Отляво „Социалната стена“ и бутона „камбанка“ за обновяване. Отдясно обновената „стена“ след натискане на „камбанката“.*

## Чат стаи

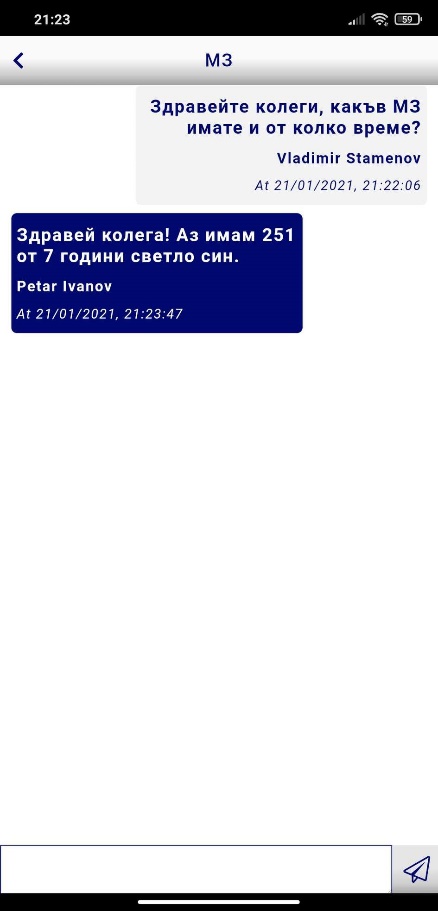
За да отворим чат отваряме менюто и избираме “Chat Groups”. На екрана виждаме, че все още няма чат стаи. Горе вдясно има бутон за добавяне на чат стаи. След натискането се отваря екран за създаване на чат група, подобен на този в „Социалната стена“. Имаме две текстови полета – за име на групата и кратко описание до 50 символа. Те са задължителни, без тях не може да се създаде група. Под тях има бутон “Upload cover image” за качване на лого или корица на групата. Логото не е задължително.

*фигура 58, 59. Отляво изглед от „Чат стаите“ когато няма създадени стаи. Отдясно екрана за добавяне на нова „чат стая“.*

След като сме готови избираме бутона “Create”. При успешно създаване на групата се връщаме към списъка с групите където виждаме новосъздадената група. Може да обновяваме списъка с чат групите през плъзване отгоре надолу по екрана.

*фигура 60, 61. Показват процеса по създаване на нова група*

След като сме създали първата си група, може да започнем да пишем в нея. Избираме групата в която искаме да пишем чрез клик. Отваря се екран, който ни казва, че все още няма съобщения. Отдолу виждаме текстово поле и бутон за изпращане на съобщение.  
След успешно добавяне на съобщение то се визуализира на екрана. Като съобщенията изпратени от нас са отдясно на сив фон със син текст. А другите потребители на син фон с бял текст. Съобщенията са подредени като най-старите са отгоре.

*фигура 62, 63, 64. Писане и четене на съобщения в чат групата.*

# Идеи за развитие

## Версия за iPhone I iPad

За момента приложението не поддържа версия за iPhone и iPad. Заради новите правила за разпространение на приложения от Apple е нужно закупуването на програмистки акаунт.

## Добавяне на още езици и възможност потребителя да си избира

Приложението сега е само на английски език. Добавянето на възможност в настройките на профила за избор на език би било полезно за потребителите. Първоначално езика ще се определя от езика на системата. Например ако устройството ни е на български език при първото отваряне ще се проверява и зарежда на български. В последствие чрез настройките на по-късен етап може да го променим.

*фигура 65. Примерен екран за смяна на езика*

## Добавяне на потребителски роли „Модератори и администратори“

Модераторите и администраторите ще се грижат за спазването на добрия тон в приложението. Модераторите ще могат да трият неприлични постове и съобщения. Както и да изпращат имейли за възстановяване на парола. Администраторите от своя страна ще могат блокират и изтриват потребители с лошо поведение, да променят профилите им, ако има обидно съдържание.

## Възможност за промяна и изтриване на съобщения от обикновените потребители

Понякога при писането на текст се допускат грешки. Затова би било хубаво да мога да променям собствените си съобщения или постове. Както и да ги изтривам.

## Възможност за коментиране и харесване на коментар в „Социалната стена“

За сега постовете могат да се само харесват и коментарат. Затова добавянето на възможност за харесване на коментарите, както и писането на отговор на коментар и харесване на отговора и така навътре докато е нужно.

# Заключение

С всеки изминал ден технологиите напредват. Стават по-надеждни и сигурни. До преди 15 години телефона беше средство за осъществяване на обаждания. Сега е неразделена част от нашето ежедневие. Използваме го за всичко – като източник на информация, игри, работа, снимки, музика, мост между нас и останалия свят. дори да си поръчваме храна. Така телефона измести настолните ни компютри, лаптопи , телевизори, камерите и фотоапарати. С няколко думи телефона е много функционално устройство, без което не можем, зависими сме от него и от това, което ни дава. Това е лошо от една страна, тази зависимост трябва да се контролира, от друга това е пазар, който не спира да се разраства. Мобилните приложения са все по-търсени и предпочитани в сравнение с десктоп приложенията. Хубавото е, че все още смарт-часовниците и смарт-гривните не са се развили до такава степен, че да изместят телефоните от употреба. Няма и изгледи скоро това да се случи така че разработките за мобилни устройства са една добра алтернатива с бъдеще. В тази ниша намират място и хибридните приложения. В момента се работи усилено в тази насока и се предлагат добри и модерни технологии за разработка.

Хибридните мобилни приложения могат да предоставят всичко необходимо, за крайния потребител. Те пестят време за създаване и поддръжка. С една технология могат да се пишат приложения за различни операционни системи което е и основното им предимство. Не се налага изучването на много технологии за всеки тип устройство. Понякога се достигат лимити, спрямо “native” приложенията, но въпреки предоставят почти всичко необходимо и понякога е добре да се замислим какво ще се прави с нашето приложение и каква технология да изберем. С крос платформените приложения имаме достъп до камера, GPS, жироскоп, инфрачервен порт, възможност за нотификации и други. С което потребностите на повечето потребители се задоволяват.

Зависимостта от мобилните устройства в наши дни е малко плашеща, както и завишената им употреба на услугите им в последната година поради настъпилата здравна криза. Но от друга страна ни дават възможност с няколко клика да намерим нужната ни информация и да си улесним живота като цяло. Това е и нашата цел като добри разработчици да направим живота на хората по-лесен и удобен без това да влияе върху здравето им. Да им представим всичко необходимо по-бързо, по-точно и по-разбираемо без да предизвикваме зависимости у тях, да пристъпваме личното им пространство и благополучие.

# Използвани източници и референции

#### [1] “Git book”, автори: „Scott Chacon, Ben Straub” <https://git-scm.com/book/en/v2> второ издание 2014г

#### [2] “npm”, официална документация <https://docs.npmjs.com/about-npm> версия 6.2.0 публикувана през юни 2018

#### [3] “Какво е Node.js?” автор „Мадлена Методиева“ <https://blog.superhosting.bg/node-js-intro.html> публикувана на 15.11.2018г. последна редакция 23.09.2020г.

#### [4] „Cordova“ официална документация, <https://cordova.apache.org/docs/en/latest/guide/overview/> публикувана 2012г последна редакция 2015г. последна версия 9.0.0

#### [5] “За какво се използва Javascript”, автор: „Speedflow”, <https://speedflow.bg/blog/what-is-javascript-what-it-is-used-for/> публикувана на 27.11.2017г.

#### [6] “JavaScript: The Good Parts 1st Edition” автор „Douglas Crockford“ ISBN-13: 978-0596517748 издадена през май 2008г.

#### [7] „Статистика за употреба на jQuery“ автор: „Web Technology Service“ <https://w3techs.com/technologies/details/js-jquery> публикувана на 30.01.2021г.

#### [8] “jQuery” – официална документация, <https://api.jquery.com> публикувана през май 2020г. последна версия 3.5.1

#### [9] “Firebase Authentication” – официална документация, автор „Google“ <https://firebase.google.com/docs/auth> последна редакция на 17.11.2020г.

#### [10] “Solving Google Maps ‘For Development Purposes Only’ Error” – автор „Accede maps“ [https://accedemaps.com/2019/04/09/solving-google-maps-for-development-purposes-only-error/](https://accedemaps.com/2019/04/09/solving-google-maps-for-development-purposes-only-error/%20) публикувана на 09.04.2020г.

# 10. Приложение

Зареждане на първия скрипт в приложението index.js

*let* auth,  
 db,  
 storage;  
*const* app = {  
 *// Application Constructor* initialize: *function*() {  
 *// document.addEventListener('deviceready', this.onDeviceReady.bind(this), false);  
 this*.bindEvents();  
 },  
 bindEvents: *function*() {  
 *if* (*typeof* cordova == "object") {  
 document.addEventListener('deviceready', *this*.onDeviceReady, *false*);  
 document.addEventListener("resume", *this*.onDeviceResume, *false*);  
 } *else* {  
 *this*.onDeviceReady("browser");  
 }  
 },  
 onDeviceReady: *function*() {  
 *setTimeout*(*function*() {  
 Database.init();  
 document.querySelector('#loading-modal-global').show();  
 ons.*ready*(*function*() {  
 ons.*disableDeviceBackButtonHandler*();  
 document.addEventListener('backbutton', *function*() {  
 *if* ($('#myNavigator').children().length > 1) {  
 Navigation.back();  
 } *else* {  
 navigator.app.exitApp();  
 }  
 }, *true*);  
 });  
 }, 0);  
 }  
};

Инициализиране на Firebase проекта

firebase.initializeApp({  
 apiKey: "AIzaSyBsVhYm-kvHpFaAA4LeMuAE33lUyP-ETaA",  
 authDomain: "bgmotoristi.firebaseapp.com",  
 databaseURL: "https://bgmotoristi.firebaseio.com",  
 projectId: "bgmotoristi",  
 storageBucket: "bgmotoristi.appspot.com",  
 messagingSenderId: "114109097623",  
 appId: "1:114109097623:web:84e08f245bbf48a53f6bd2",  
 measurementId: "G-QPS7XLB51K"  
});

Инициализиране на базата данни и вземането да нужната информация

*const* Database = (*function*() {  
  
 *function* init() {  
 auth = firebase.auth();  
 db = firebase.firestore();  
 storage = firebase.storage();  
 auth.onAuthStateChanged(*function*(user) {  
 *if* (user) {  
 start(); *// User is signed in.* Home.init();  
 } *else* {  
 Login.init(); *// User is signed out.* }  
  
 *function* start() {  
 db.collection('users').onSnapshot(*function*(querySnapshot) {  
 *let* usersArray = [];  
 *for* (*let* i = 0; i < querySnapshot.docs.length; i += 1) {  
 *let* userData = querySnapshot.docs[i].data();  
 userData.id = querySnapshot.docs[i].id;  
 usersArray.push(userData);  
 }  
 Users.set(usersArray);  
 });  
  
 db.collection('coords').onSnapshot(*function*(querySnapshot) {  
 *let* coordsArray = [];  
 *for* (*let* i = 0; i < querySnapshot.docs.length; i += 1) {  
 *let* coordsData = querySnapshot.docs[i].data();  
 *if* (coordsData.userId === localStorage.getItem('logged\_user\_id')) {  
 localStorage.setItem('logged\_user\_position', JSON.stringify(coordsData));  
 }  
 coordsData.id = querySnapshot.docs[i].id;  
 coordsArray.push(coordsData);  
 }  
 Coordinates.set(coordsArray);  
 });  
 }  
  
  
 });  
 }  
  
 *function* addImage(file, path) {  
 *if* (!path) { path = `profile-images/${Math.floor(Math.random() \* 10000)}\_${file.name}`; }  
 *return new* Promise((resolve, reject) => {  
 *// Create the file metadata  
 var* metadata = {  
 contentType: 'image/jpeg'  
 };  
 *var* uploadTask = storage.ref().child(path).put(file, metadata);  
 *// Listen for state changes, errors, and completion of the upload.* uploadTask.*on*('state\_changed', *// or firebase.storage.TaskEvent.STATE\_CHANGED  
 function*(snapshot) {  
 *switch* (snapshot.state) {  
 *case* 'paused': *// or firebase.storage.TaskState.PAUSED* Debug.log({  
 level: 1,  
 message: 'firebase.service => Upload is paused'  
 });  
 *break*;  
 *case* 'running': *// or firebase.storage.TaskState.RUNNING* Debug.log({  
 level: 1,  
 message: 'firebase.service => Upload is running'  
 });  
 *break*;  
 }  
 },  
 *function*(error) {  
 reject(error);  
 },  
 *function*() {  
 uploadTask.snapshot.ref.getDownloadURL().then(*function*(downloadURL) {  
 resolve(downloadURL);  
 }); *// Upload completed successfully, now we can get the download URL* });  
 });  
 }  
  
 *return* {  
 init: init,  
 addImage: addImage  
 };  
}());

Модул, който се грижи на навигацията в приложението

*const* Navigation = (*function* () {  
 'use strict';  
 *let* currPageId = '';  
   
 *const* loadPage = *function* (pageId, animation, controller) {  
 *if* (currPageId === pageId) {  
 *return*;  
 }  
 *//remove firebase hook  
 if* (currPageId === 'social-wall-ons-page') {  
 SocialWall.destroy();  
 }  
 *if* (currPageId !== 'chat-group-page') {  
 Chat.destroy();  
 }  
 *if* (currPageId !== 'add-comment-page') {  
 PostComments.destroy();  
 }  
 Loading.show();  
 $('#menu').attr('swipeable', *true*);  
 *let* appMenu = document.getElementById('menu'),  
 appNavigator = document.getElementById('myNavigator'),  
 anim = 'fade-ios';  
 *if* (animation !== '') {  
 anim = animation;  
 }  
 currPageId = pageId;  
 appMenu.close();  
 appNavigator.resetToPage(pageId, {  
 animation: anim,  
 animationOptions: {  
 duration: 0.2,  
 delay: 0,  
 timing: 'ease-in'  
 },  
 callback: controller,  
 data: {  
 moduleHash: pageId  
 }  
 });  
  
 };  
 *const* pushPage = *function* (page, anim, callbackFunction) {  
 Loading.show();  
 *let* appNavigator = document.getElementById('myNavigator'),  
 pageAnimation = 'slide-ios';  
 *if* (anim === '') {  
 pageAnimation = 'slide-ios';  
 } *else* {  
 pageAnimation = anim;  
 }  
 currPageId = page;  
 document.getElementById('menu').close();  
 appNavigator.pushPage(page, {  
 animation: pageAnimation,  
 callback: callbackFunction,  
 data: {moduleHash: page}  
 });  
 };  
 *const* openMenu = *function* () {  
 *let* appMenu = document.getElementById('menu');  
 appMenu.open();  
 };  
 *const* popPage = *function* () {  
 *let* appNavigator = document.getElementById('myNavigator');  
 *let* lastPage = appNavigator.pages[appNavigator.pages.length - 1];  
  
 *// clear chat hook  
 if* (lastPage.classList.value.indexOf('chat-group-page') > -1) {  
 Chat.destroy();  
 }  
 *if* (lastPage.classList.value.indexOf('add-comment-page') > -1) {  
 PostComments.destroy();  
 }  
 appNavigator.popPage({  
 callback: *function* () {  
 *console*.log('Pop page...');  
 }  
 })  
 };  
  
 *return* {  
 load: loadPage,  
 push: pushPage,  
 openMenu: openMenu,  
 pop: popPage  
 }  
}());

Глобален „loading“ модул, който е видим на всички страници

*const* Loading = (*function*() {  
  
 *let* loadingModal,  
 loadingModalHTML = '';  
  
 loadingModalHTML += '<ons-modal id="loading-modal-template" direction="up">';  
 loadingModalHTML += '<div class="container-outer"><div class="container-inner">';  
 loadingModalHTML += '<div class="icon-box"><i class="fal fa-tire fa-spin loading-icon"></i></div>';  
 loadingModalHTML += '</div></div>';  
 loadingModalHTML += '</ons-modal>';  
 $('ons-splitter').after(loadingModalHTML);  
 loadingModal = document.querySelector('#loading-modal-template');  
  
 *// actionn : show or hide  
 function* show() {  
 loadingModal.show();  
 }  
  
 *function* hide() {  
 loadingModal.hide();  
 }  
  
 *return* {  
 show: show,  
 hide: hide  
 }  
}());

Сървис, който използваме за достъп до камера/галерия на телефона

*const* CameraService = (*function*() {  
 'use strict';  
 *function* showCameraActionSheet() {  
   
 *return new* Promise((resolve, reject) => {  
 *let* actionSheetOptions = {  
 cancelable: *true*,  
 *class*: 'camera-api-options-sheet',  
 buttons: [`Take Picture`, `Choose From Gallery`, `Cancel`]  
 };  
  
 *function* takePicture() {  
 *let* cameraOptions = {  
 quality: 75,  
 destinationType: Camera.DestinationType.DATA\_URL,  
 sourceType: Camera.PictureSourceType.CAMERA,  
 encodingType: Camera.EncodingType.JPEG,  
 correctOrientation: *true*,  
 targetWidth: 500,  
 targetHeight: 500,  
 };  
 navigator.camera.getPicture(onSuccess, onFail, cameraOptions);  
  
 *function* onSuccess(imageData) { resolve(imageData); }  
  
 *function* onFail(errMessage) { reject(errMessage); }  
 }  
  
 *function* getFromGallery() {  
 *let* cameraOptions = {  
 quality: 75,  
 destinationType: Camera.DestinationType.DATA\_URL,  
 sourceType: Camera.PictureSourceType.PHOTOLIBRARY,  
 encodingType: Camera.EncodingType.JPEG,  
 correctOrientation: *true*,  
 cameraDirection: 1,  
  
 };  
 navigator.camera.getPicture(onSuccess, onFail, cameraOptions);  
  
 *function* onSuccess(imageData) { resolve(imageData); }  
  
 *function* onFail(errMessage) { reject(errMessage); }  
 }  
  
 ons.*openActionSheet*(actionSheetOptions).then(*function*(index) {  
 *if* (index === 0) {  
 takePicture();  
 } *else if* (index === 1) {  
 getFromGallery();  
 } *else* {  
  
 }  
 });  
 });  
 }  
 *return* {  
 showCameraActionSheet: showCameraActionSheet  
 };  
}());

Сървис, който се грижи за създаването на чат групи

*const* ChatService = (*function*() {  
 'use strict';  
 *let* chatGroupsRef;  
  
 *function* getRef() {  
 *if* (chatGroupsRef) {  
 *return* chatGroupsRef;  
 } *else* {  
 chatGroupsRef = db.collection('chat\_groups');  
 *return* chatGroupsRef;  
 }  
 }  
  
 *function* addGroupWithImage(file, name, description) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* postObject = {};  
 Database.addImage(file, `chat\_groups/${Math.floor(Math.random() \* 10000)}\_${file.name}`).then(*function*(fileUrl) {  
 *const* postRef = getRef().doc();  
 postObject.id = postRef.id;  
 postObject.name = name;  
 postObject.description = description;  
 postObject.coverUrl = fileUrl;  
 postObject.createdBy = localStorage.getItem('logged\_user\_id');  
  
  
 getRef().doc(postRef.id).set(postObject).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 });  
 });  
 }  
  
  
  
 *function* addGroupWithoutImage(name, description) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* postObject = {};  
 *const* postRef = getRef().doc();  
 postObject.id = postRef.id;  
 postObject.name = name;  
 postObject.description = description;  
 postObject.createdBy = localStorage.getItem('logged\_user\_id');  
 postObject.coverUrl = *null*;  
  
 getRef().doc(postRef.id).set(postObject).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 });  
 }  
  
 *return* {  
 getRef: getRef,  
 addGroupWithImage: addGroupWithImage,  
 addGroupWithoutImage: addGroupWithoutImage,  
 };  
}());

Сървис който взима текущата позиция на потребителя и я записва във Firestore

*const* MapsService = (*function*() {  
 'use strict';  
 *function* getCurrentLocation() {  
 *return new* Promise(*function*(resolve, reject) {  
 navigator.geolocation.getCurrentPosition(*function*(position) {  
 localStorage.setItem('logged\_user\_position', JSON.stringify({  
 latitude: position.coords.latitude,  
 longitude: position.coords.longitude,  
 altitude: position.coords.altitude,  
 accuracy: position.coords.accuracy,  
 altitudeAccuracy: position.coords.altitudeAccuracy,  
 heading: position.coords.heading,  
 speed: position.coords.heading,  
 timestamp: position.timestamp  
 }));  
 *if* (localStorage.getItem('logged\_user\_deny\_access\_to\_location')) {  
 localStorage.removeItem('logged\_user\_deny\_access\_to\_location');  
 }  
 db.collection('coords').where("userId", "==", localStorage.getItem('logged\_user\_id')).limit(1).get().then(*function*(firebaseData) {  
  
 *if* (firebaseData.docs.length) {  
 *let* data = firebaseData.docs[0].data();  
 db.collection('coords').doc(data.id).update({  
 id: data.id,  
 userId: localStorage.getItem('logged\_user\_id'),  
 latitude: position.coords.latitude,  
 longitude: position.coords.longitude,  
 timestamp: position.timestamp || *new* Date().getTime  
 });  
 resolve('Success');  
 } *else* {  
 *const* ref = db.collection("coords").doc();  
 *const* userId = localStorage.getItem('logged\_user\_id') || *null*;  
 *if* (userId) {  
 db.collection('coords').doc(ref.id).set({  
 id: ref.id,  
 userId: localStorage.getItem('logged\_user\_id'),  
 latitude: position.coords.latitude,  
 longitude: position.coords.longitude,  
 timestamp: position.timestamp || *new* Date().getTime  
 });  
 resolve('Success');  
 } *else* {  
 reject('Invalid user!');  
 }  
  
 }  
  
 });  
 }, *function*(error) {  
 localStorage.setItem('logged\_user\_deny\_access\_to\_location', 1);  
 reject(error);  
 });  
 });  
 }  
 *return* {  
 getCurrentLocation: getCurrentLocation  
 };  
}());

Сървис, с който се записват съобщенията в чата

*const* MessagesService = (*function*() {  
 'use strict';  
 *let* MessagesRef;  
  
 *function* getRef() {  
 *if* (MessagesRef) {  
 *return* MessagesRef;  
 } *else* {  
 MessagesRef = db.collection('messages');  
 *return* MessagesRef;  
 }  
 }  
  
 *function* addMessage(text, groupId) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* messageObject = {};  
 *const* postRef = getRef().doc();  
 messageObject.id = postRef.id;  
 messageObject.groupId = groupId  
 messageObject.text = text;  
 messageObject.userId = localStorage.getItem('logged\_user\_id');  
 messageObject.timestamp = *new* Date().getTime();  
  
 getRef().doc(postRef.id).set(messageObject).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 });  
 }  
  
 *return* {  
 getRef: getRef,  
 addMessage: addMessage,  
   
 };  
}());

Сървис за добавяне на постове със или без снимка, харесване и коментиране на пост

*const* SocialWallService = (*function*() {  
 'use strict';  
 *let* socialWallRef = *null*,  
 postCommentsRef = *null*;  
  
 *function* getRef() {  
 *if* (socialWallRef) {  
 *return* socialWallRef;  
 } *else* {  
 socialWallRef = db.collection('social');  
 *return* socialWallRef;  
 }  
 }  
  
 *function* getPostCommentsRef() {  
 *if* (postCommentsRef) {  
 *return* postCommentsRef;  
 } *else* {  
 postCommentsRef = db.collection('comments');  
 *return* postCommentsRef;  
 }  
 }  
  
 *function* addPostWithImage(file, postText) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* postObject = {};  
 Database.addImage(file, `social\_wall/${Math.floor(Math.random() \* 10000)}\_${file.name}`).then(*function*(fileUrl) {  
 *const* postRef = getRef().doc();  
 postObject.id = postRef.id;  
 postObject.text = postText;  
 postObject.userId = localStorage.getItem('logged\_user\_id');  
 postObject.likedUsers = [];  
 postObject.comments = 0;  
 postObject.timestamp = *new* Date().getTime();  
 postObject.imageUrl = fileUrl;  
  
 getRef().doc(postRef.id).set(postObject).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 *console*.log(error);  
 });  
 });  
 });  
 }  
  
 *function* likePost(postData) {  
 getRef().doc(postData.id).update(postData).then(*function*(success) {}).catch(*function*(error) {  
 *console*.log(error);  
 });  
 }  
  
 *function* addPostWithOutImage(postText) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* postObject = {};  
 *const* postRef = getRef().doc();  
 postObject.id = postRef.id;  
 postObject.text = postText;  
 postObject.userId = localStorage.getItem('logged\_user\_id');  
 postObject.likedUsers = [];  
 postObject.comments = 0;  
 postObject.timestamp = *new* Date().getTime();  
  
 getRef().doc(postRef.id).set(postObject).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 });  
 }  
  
 *function* addPostComment(commentMessage, postId) {  
 *return new* Promise(*function*(resolve, reject) {  
 *let* commentObject = {};  
 *const* postCommentRef = getPostCommentsRef().doc();  
 *const* postRef = getRef().doc(postId);  
 commentObject.id = postCommentRef.id;  
 commentObject.postId = postId;  
 commentObject.userId = localStorage.getItem('logged\_user\_id');  
 commentObject.text = commentMessage;  
 commentObject.timestamp = *new* Date().getTime();  
 postRef.get().then(*function*(res) {  
 getPostCommentsRef().doc(postCommentRef.id).set(commentObject).then(*function*(success) {  
 postRef.update({ comments: res.data().comments += 1 }).then(*function*(success) {  
 resolve(success);  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 }).catch(*function*(error) {  
 reject(error);  
 });  
 });  
  
 });  
 }  
  
 *return* {  
 getRef: getRef,  
 getPostCommentsRef: getPostCommentsRef,  
 addPostWithImage: addPostWithImage,  
 addPostWithOutImage: addPostWithOutImage,  
 likePost: likePost,  
 addPostComment: addPostComment  
 };  
})();

Инициализиране на страницата с картата и маркерите на нея

*const* UsersMap = (*function*() {  
 'use strict';  
 *function* init() {  
 *if* (localStorage.getItem('logged\_user\_deny\_access\_to\_location') == 1 || !localStorage.getItem('logged\_user\_position')) {  
 ons.notification.*alert*({  
 message: 'Please share your location first',  
 title: 'Oops!',  
 buttonLabels: ['Cancel', 'OK'],  
 primaryButtonIndex: 1  
 }).then(*function*(btnIndex) {  
 *if* (btnIndex === 1) {  
 MapsService.getCurrentLocation().then(*function*() {  
 loadModule();  
 }, *function*(error) {  
 ons.notification.*alert*('It seems you have to give permission from settings. Go to settings and allow to use location!');  
 });  
 } *else* {  
 ons.notification.*alert*('It seems you have to give permission from settings. Go to settings and allow to use location!');  
 }  
 });  
 } *else* {  
 loadModule();  
 }  
  
 *function* loadModule() {  
 template.load('usersMap');  
 Navigation.load('users-map-ons-page', 'fade-ios', controller);  
 }  
  
 *function* controller() {  
 Loading.hide();  
 *let* map,  
 coords = JSON.parse(localStorage.getItem('logged\_user\_position')),  
 usersArray = Users.get(),  
 markersArray = Coordinates.get().filter((coords) => {  
 *if* (coords.userId !== localStorage.getItem('logged\_user\_id')) { *return* coords; };  
 });  
 loadMap();  
 $('#menu').removeAttr('swipeable');  
  
 document.querySelector('.toolbar .right').addEventListener('click', *function*() {  
 MapsService.getCurrentLocation().then(*function*() {  
 coords = JSON.parse(localStorage.getItem('logged\_user\_position'));  
 usersArray = Users.get();  
 markersArray = Coordinates.get().filter((coords) => {  
 *if* (coords.userId !== localStorage.getItem('logged\_user\_id')) { *return* coords; };  
 });  
 loadMap();  
 }, *function*() {  
 ons.notification.*alert*('It seems you have to give permission from settings. Go to settings and allow to use location!');  
 });  
 });  
  
 *function* loadMap() {  
 initMap();  
 initMarker(coords.latitude, coords.longitude, "You are here!", coords.timestamp, 1);  
 *//another users position  
 if* (markersArray.length) {  
 *for* (*let* i = 0; i < markersArray.length; i++) {  
 *let* userData = usersArray.filter((user) => {  
 *if* (user.id === markersArray[i].userId) { *return* user; };  
 })[0];  
 initMarker(markersArray[i].latitude, markersArray[i].longitude, `<span onclick="UserDetails.init('${userData.id}')">${userData.firstName}</span> was here at `, markersArray[i].timestamp)  
 }  
 }  
 }  
  
 *function* initMap() {  
 map = *new* google.maps.Map(document.getElementById("map"), {  
 center: { lat: coords.latitude, lng: coords.longitude },  
 zoom: 12  
 });  
 }  
  
 *function* initMarker(markerLatitude, markerLongitude, markerTitle, timePicked, currentUser) {  
  
 *let* currentLocationMarker,  
 markerIcon,  
 markersArray = [  
 'img/map/helmet-red.svg',  
 'img/map/helmet-blue.svg',  
 'img/map/helmet-green.svg',  
 'img/map/helmet-black.svg',  
 'img/map/helmet-violet.svg',  
 'img/map/helmet-white.svg',  
 'img/map/helmet-yellow.svg',  
 ];  
 *if* (currentUser === 1) {  
 markerIcon = markersArray[0];  
 } *else* {  
 markerIcon = markersArray[Math.floor(Math.random() \* 6)];  
 *if* (!markerIcon) { markerIcon = markersArray[6]; }  
 }  
  
 currentLocationMarker = *new* google.maps.Marker({  
 position: {  
 lat: markerLatitude,  
 lng: markerLongitude  
 },  
 animation: google.maps.Animation.DROP,  
 map: map,  
 title: markerTitle,  
 icon: *new* google.maps.MarkerImage(  
 markerIcon,  
 *new* google.maps.Size(30, 30),  
 *new* google.maps.Point(0, 0),  
 *new* google.maps.Point(15, 15),  
 *new* google.maps.Size(30, 30))  
 });  
  
  
 currentLocationMarker.addListener('click', *function*() {  
 map.setCenter(currentLocationMarker.getPosition());  
 *let* infowindow = *new* google.maps.InfoWindow({  
 content: '<div class="map-popup">' + markerTitle + '<div>' + *new* Date(timePicked).toLocaleString() + '</div></div>',  
 position: currentLocationMarker.getPosition()  
 });  
 infowindow.open(map);  
 });  
 }  
 }  
  
 }  
  
 *return* {  
 init: init  
 };  
}());

Инициализиране на страницата за вход и методите за регистрация и вписване

*const* Login = (*function*() {  
 'use strict';  
 *function* init() {  
 template.load('login');  
 Navigation.load('login-ons-page', 'fade-ios', controller);  
  
 *function* controller() {  
 Loading.hide();  
 $('#menu').removeAttr('swipeable');  
 *let* signUpForm = document.getElementById('sign-up-form'),  
 loginForm = document.getElementById('login-form');  
 *//signUp* signUpForm.addEventListener('submit', (e) => {  
 e.preventDefault();  
 *let* email = signUpForm['sign-up-email'].value;  
 *let* password = signUpForm['sign-up-password'].value;  
 auth.createUserWithEmailAndPassword(email, password).then((cred) => {  
 *return* db.collection('users').doc(cred.user.uid).set({  
 firstName: signUpForm['sign-up-first-name'].value,  
 lastName: signUpForm['sign-up-last-name'].value,  
 email: email  
 }).then(() => {  
 localStorage.setItem('logged\_user\_id', cred.user.uid);  
 Home.init();  
 });  
 }).catch((error) => {  
 ons.notification.*alert*({  
 message: error.message  
 });  
 });  
 });  
 $('#go-to-register').on('click', () => {  
 $('.login-form-container').addClass('hide');  
 $('.sign-up-form-container').removeClass('hide');  
 });  
  
 $('#back-to-login').on('click', () => {  
 $('.sign-up-form-container').addClass('hide');  
 $('.login-form-container').removeClass('hide');  
 });  
  
 loginForm.addEventListener('submit', (e) => {  
 e.preventDefault();  
 *let* email = loginForm['login-email'].value;  
 *let* password = loginForm['login-password'].value;  
 auth.signInWithEmailAndPassword(email, password).then((cred) => {  
 localStorage.setItem('logged\_user\_id', cred.user.uid);  
 }, (error) => {  
 ons.notification.*toast*({  
 message: error.message,  
 timeout: 4000  
 });  
 });  
 });  
 }  
  
 }  
  
 *return* {  
 init: init  
 };  
}());