**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Доцент департамента  программной инженерии  факультета компьютерных наук,  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Л. Макаров  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **АНДРОИД-ПРИЛОЖЕНИЕ "ЗЕЛЕНАЯ ВОЛНА ДЛЯ ПЕШЕХОДОВ"**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.16-01 81** **01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ162  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.О. Скудняков /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.04.16-01 81 01-1-ЛУ |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **АНДРОИД-ПРИЛОЖЕНИЕ "ЗЕЛЕНАЯ ВОЛНА ДЛЯ ПЕШЕХОДОВ"**  **Пояснительная записка**  **RU.17701729.04.16-01 81 01-1**  **Листов 35** | | |
|  |  | |
|  | |

**АННОТАЦИЯ**

В данном документе приведена пояснительная записка к программе «Андроид-приложение "Зеленая волна для пешеходов"». С помощью приложения пользователь может получить информацию о рекомендуемой скорости движения для пересечения перекрестка на зеленый свет.

В данном программном документе, в разделе «Введение» указано наименование программы, а также документы, на основании которых ведется разработка.

В разделе «Назначение и область применения» указано функциональное и эксплуатационное назначение программы вместе с краткой характеристикой области применения программы.

В разделе «Технические характеристики» содержатся следующие подразделы:

* Постановка задачи для последующей разработки программы;
* Описание применяемых методов;
* Описание внутренних алгоритмов программы, позволяющих выполнять поставленную задачу;
* Описание и обоснование выбора метода организации входных и выходных данных;
* Описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [6];
2. ГОСТ 19.102-77 Стадии разработки [7];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [8];
4. ГОСТ 19.104-78 Основные надписи [9];
5. ГОСТ 19.105-78 Общие требования к программным документам [10];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [11];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [12].

Изменения к данному Техническому заданию оформляются согласно ГОСТ 19.603-78 [13], ГОСТ 19.604-78 [14].

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 5](#_Toc513656436)

[1.1 Наименование программы 5](#_Toc513656437)

[1.2 Документы, на основании которых ведется разработка 5](#_Toc513656438)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 6](#_Toc513656439)

[2.1 Назначение программы 6](#_Toc513656440)

[2.1.1 Функциональное назначение 6](#_Toc513656441)

[2.1.2 Эксплуатационное назначение программы 6](#_Toc513656442)

[2.2 Краткая характеристика области применения программы 6](#_Toc513656443)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc513656444)

[3.1 Постановка задачи на разработку программы 7](#_Toc513656445)

[3.2 Описание алгоритма и функционирования программы 7](#_Toc513656446)

[3.2.1 Архитектура реализации 7](#_Toc513656447)

[3.2.2 Структура базы данных 8](#_Toc513656448)

[3.2.3 Общая схема работы поиска светофоров 9](#_Toc513656449)

[3.2.4 Создание и первоначальная настройка Activity 9](#_Toc513656450)

[3.2.5 Поиск ближайших светофоров 10](#_Toc513656451)

[3.2.6 Выбор одного ближайшего светофора 11](#_Toc513656452)

[3.2.7 Работа с настройками светофоров 11](#_Toc513656453)

[3.2.8 Расчет рекомендуемой скорости 12](#_Toc513656454)

[3.2.9 Автоматическое обновление информации для пользователя 12](#_Toc513656455)

[3.3 Описание и обоснование метода организации входных и выходных данных 13](#_Toc513656456)

[3.3.1 Описание метода организации входных и выходных данных 13](#_Toc513656457)

[3.3.2 Обоснование метода организации входных и выходных данных 13](#_Toc513656458)

[3.4 Описание и обоснование метода выбора технических и программных средств 13](#_Toc513656459)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 14](#_Toc513656460)

[4.1 Предполагаемая потребность 14](#_Toc513656461)

[4.2 Экономические преимущества продукта по сравнению с отечественными или зарубежными аналогами и образцами. 14](#_Toc513656462)

[5. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 15](#_Toc513656463)

[ПРИЛОЖЕНИЕ 1 16](#_Toc513656464)

[ПРИЛОЖЕНИЕ 2 17](#_Toc513656465)

# ВВЕДЕНИЕ

## Наименование программы

Полное наименование программы – «Андроид-приложение "Зеленая волна для пешеходов"».

Наименование программы на английском языке – «Pedestrian Green Wave Android Application».

Краткое наименование программы – «Зеленая волна».

## Документы, на основании которых ведется разработка

Приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» № 2.3-02/1212-01 от 12.12.2017 "Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук".

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## Назначение программы

### Функциональное назначение

Функциональным назначением программы является помощь пешеходам в их безостановочном движении к точке назначения. Основная функция приложения – расчет времени до прибытия к светофору и подсказка пользователю, с какой скоростью следует двигаться, чтобы горел зеленый свет ко времени прибытия.

### Эксплуатационное назначение программы

Программой может пользоваться любой пешеход со смартфоном на операционной системе Android версии выше 4.2. Пользователь вводит расписание светофоров и периодичность смены сигналов. Далее при приближении к светофору с уже установленным расписанием пользователь сможет плавно скорректировать свою скорость, чтобы она совпадала с рекомендуемой скоростью движения. Таким образом, будет осуществляться безостановочное движение пользователя по регулируемым перекресткам.

## Краткая характеристика области применения программы

Любой пешеход может быть заинтересован в данном приложении. Движение с периодическими остановками на перекрестке может стать причиной разных неприятностей, например пешехода могут окатить водой из лужи проезжающие автомобили. Бывают ситуации, когда пешеход, видя, что он не успевает на зеленый свет, пытается перебежать оживленную улицу. Данное приложение поможет заранее узнать, с какой скоростью нужно идти, чтобы успеть вовремя.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

## Постановка задачи на разработку программы

Цель работы - реализовать приложение для ОС Android, которое будет помогать пешеходам при переходе регулируемых перекрестков.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1. Информирование пользователя о рекомендуемой скорости движения
2. Добавление светофора
3. Использование сервиса OpenStreetMap для получения ближайших светофоров
4. Возможность установления расписания светофоров
5. Сохранение расписания в удаленную базу данных
6. Отображение карты с добавленными светофорами
7. Расчет средней скорости пешехода за определенный промежуток времени
8. Проверка входных пользовательских данных
9. Выбор ближайшего светофора автоматически
10. Выбор желаемого светофора пользователем

## Описание алгоритма и функционирования программы

### Архитектура реализации

Реализация приложения построена на архитектуре Model-View-Presenter[1] (Рис. 1). В основном окне view – это GreenwaveActivity, выполняющий интерфейс GreenwaveView с основными функциями оповещения Presenter о действиях пользователя.

Presenter получает пользовательские запросы с помощью своего View и переадресует их Model, или решает проблему самостоятельно.

Model работает с данными, такими как настройки светофоров и их расположения. Model работает только с Presenter, ничего не зная о существовании View.

Все классы работают по контракту их интерфейсов. Для GreenwaveActivity – GreenwaveView, GreenwavePresenter – GreenwavePresenterApi, GreenwaveModel – GreenwaveModelApi.

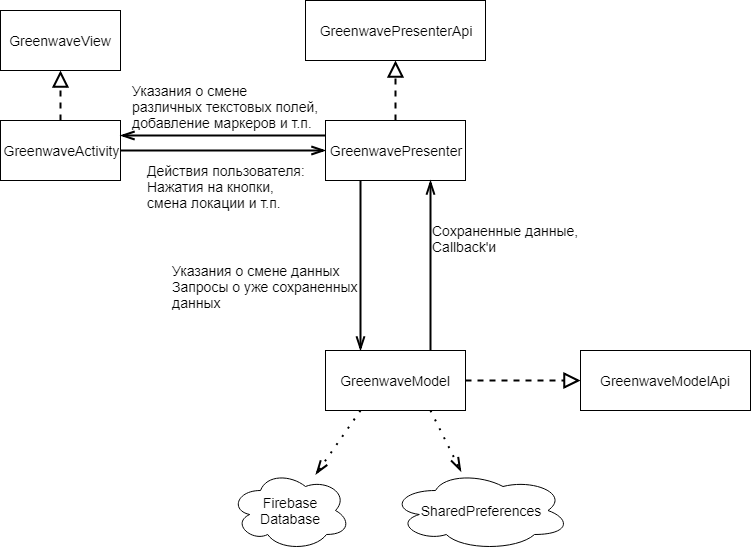


Рис. 1 - Архитектура приложения

### Структура базы данных

Данные хранятся в Firebase Realtime Database[2] . Есть ветка lights (Рис. 2) со всеми настройками от пользователей. Новые настройки добавляются по ключу – по долготе и широте светофора, но с замененными точками на точки с запятой (требование БД). Таким образом ключи будут уникальны. По ключу хранятся настройки светофора. Для удобства работы, в БД сериализуется Java объект, реализующий Parcelable. Так его будет удобно десериализовать. Хранятся поля: greenCycle, redCycle, Identifier, startOfMeasurement, set.



Рис. 2 - Структура хранения информации в Firebase Database

### Общая схема работы поиска светофоров

Запрос к OSM API должен содержать определенные координаты прямоугольника, в котором будет производиться поиск светофоров. Эти координаты получаются при подписке клиента (Java объект в GreenwaveActivity) на событие LocationUpdate. В callback’e приходят координаты. View переадресовывает их presenter’у, тот в свою очередь делает запрос к модели. Именно там и происходит вся работа с сетью. Итак, делается запрос на поиск всех светофоров, в итоге получается набор объектов. Далее их надо отфильтровать: сначала отсекаем те, что не находятся на пути движения (Рис. 3), далее фильтруем по расстоянию. В итоге выбираем ближайший светофор как активный и начинаем расчет времени прибытия до него. Каждые 150 метров эта процедура повторяется. Также каждое передвижение проверяем, прошел ли человек светофор. Если угол между вектором движения человека и вектором движения к светофору больше 90, значит выбранный светофор должен быть перевыбран.

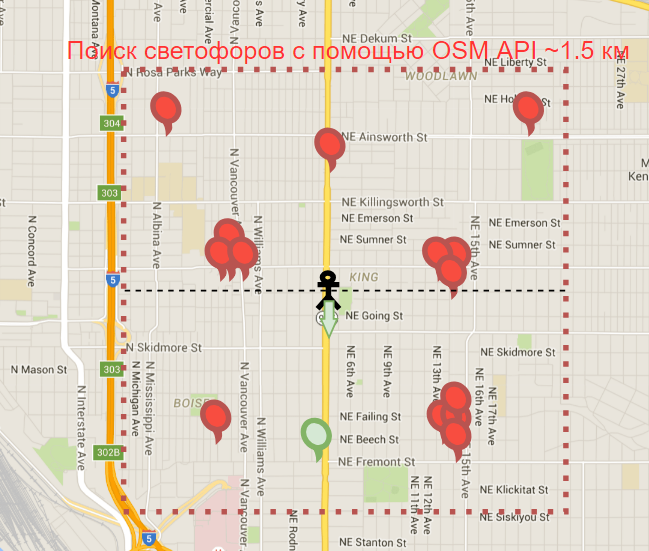


Рис. 3 - Схема работы поиска светофоров

### Создание и первоначальная настройка Activity

При первом входе пользователь должен подтвердить свое согласие на использование геолокации. Если согласие получено, то начинаем создание карты, на которой будет отображать вся информация. Этот вызов асинхронный, результат будет получен в методе onMapReady:

**override fun** onMapReady(map: GoogleMap?) {  
 **if** (*DEBUG*) Log.d(**TAG**, **"(53, GreenwaveActivity.kt) onMapReady $**map**"**)  
 **this**.**map** = map  
  
 map?.setOnMapLongClickListener **{ presenter**.addMapMark(**it**) **}** map?.setOnCameraMoveStartedListener **{  
 presenter**.onCameraMoved()  
 **}** map?.setOnMarkerClickListener **{** showLightInfoView(**it**) **}** map?.setOnInfoWindowClickListener **{** Log.d(**TAG**, **"choose light"**)  
 **presenter**.chooseNewLight(**it**.*position*)  
 **}** map?.setOnInfoWindowLongClickListener **{ presenter**.openLightSettings(**it**) **}** map?.setOnMapClickListener **{** hideLightInfoView()  
 **followUser** = **false** *// ux feature* **}  
  
 presenter**.onMapReady(map)  
 getDeviceLocation()?.addOnSuccessListener **{ presenter**.requestNearestLights(**it**) **}**}

В нем настраивается основное взаимодействия пользователя с картой: движение камеры, создание новых маркеров, взаимодействие с маркерами.

После того, как карта загрузилась, оповещаем об этом presenter и делаем запрос на поиск ближайших светофоров.

### Поиск ближайших светофоров

Presenter обрабатывает объект Location, полученный из его View и делает запрос к Model на получаение ближайших светофоров:

**override fun** requestNearestLights(location: Location) {  
 **if** (*DEBUG*) Log.d(**TAG**, **"(99, GreenwavePresenterr.kt) requestNearestLights for $**location**"**)  
 **model**.requestNearestLights(location.*latitude*.toFloat(), location.*longitude*.toFloat())  
 forceChooseNewClosestLight()  
}

Модель делает загрузку светофоров из OpenStreetMap Api с помощью Retrofit и RxJava:

**private fun** requestNearestLightsFromApi(lat: Float, lng: Float) {  
 **osmService**.fetchChunkData(createQuery(lat, lng))  
 .subscribeOn(Schedulers.io())  
 .observeOn(AndroidSchedulers.mainThread())  
 .subscribe(  
 **{** result **->** Log.d(**TAG**, **"result = ${**result.**elements}"**)  
 onReceiveNearestLights(result)  
 **}**,  
 **{** error **->** Log.d(**TAG**, **"error: $**error**"**)  
 **}** )  
}

Использование связки Retrofit[3] и RxJava[4] позволяет вынести всю работу с интернетом в background треды, таким образом приложение стабильно будет отвечать на запросы пользователя.

Результат приходит в метод onReceiveNearestLights(result). Там происходит разбор полученного результата и создание Java объектов TrafficLight. С этими объектами гораздо проще работать, чем с сырым ответов OSM Api[5].

В итоге вызывается метод presenter.onReceiveNearestLights, в который передается List с ближайшими светофорами.

Presenter создает новые маркеры на карте, используя информацию из полученного списка светофоров

Запрос на обновление ближайших светофоров будет произведен автоматически, когда пользователь пройдет достаточное расстояние с позиции прошлого обновления. Настройка расстояния производится программистом в файле GreenwaveConstants.kt

### Выбор одного ближайшего светофора

За обновление ближайшего светофора отвечает presenter. Всякий раз, когда происходит обновление позиции, проверяется, нужно ли обновить ближайший светофор (по такому же принципу, как обновляются все светофоры). Если время пришло, делается запрос к модели:

**override fun** getNearestLight(currentLocation: Location): TrafficLight? {  
 **nearestLights** ?: **return null  
 lastQueryLightLocation** = currentLocation  
  
 **val** latLng = LatLng(currentLocation.*latitude*, currentLocation.*longitude*)  
 **val** movementVector = getMovementVector(currentLocation)  
  
 **val** closestLights = **nearestLights**!!  
 .*filter* **{** isLightCloserThan(**it**, *NEAREST\_LIGHT\_DISTANCE*, currentLocation) **}** .*filter* **{** isLightInFront(**it**, latLng, movementVector) **}** .*sortedBy* **{** getDistance(latLng, LatLng(**it**.**lat**, **it**.**lng**)) **}  
  
 if** (*DEBUG*) Log.d(**TAG**, **"(79, GreenwaveModel.kt) nearest lights: $**closestLights**"**)  
  
 **if** (closestLights.*isNotEmpty*()) {  
 **closestLight** = closestLights.*first*()  
 **return closestLight** }  
 **return null**}

Для начала рассчитывается вектор движения пользователя, чтобы определить, находится ли светофор перед пользователем, или сзади него. Все светофоры фильтруются по радиусу обзора, признаку нахождения перед пользователем. Потом выбирается ближайший и возвращается presenter’у. Теперь presenter готов отображать информацию с подсказками о движении, но ему нужны настройки.

### Работа с настройками светофоров

Все настройки хранятся в Firebase Realtime Database. Для удобства пользователей включено кэширование ответов удаленной БД. Для считывания настроек создана система, построенная на callback механизме, так как вся работа с сетью должна быть асинхронной. Когда клиент хочет получить настройки, он вызывает соответствующий метод в Model:

**override fun** requestSettingsForLight(identifier: String, request: Int) {  
 **val** lightsRef = FirebaseDatabaseSingletone.getFirebaseInstance().getReference(*LIGHTS\_REFERENCE\_FIREBASE*)  
 **val** noSettingsTimer = *timer*(**"no\_settings"**, **false**, *SECOND\_IN\_MILLIS*, *SECOND\_IN\_MILLIS*,  
 **{** replyToRequest(request, LightSettings(identifier))  
 **this**.cancel()  
 **}**)  
 lightsRef.addListenerForSingleValueEvent(**object** : ValueEventListener {  
 **override fun** onDataChange(snapshot: DataSnapshot) {  
 **var** lightSettings = LightSettings(identifier)  
 **if** (snapshot.hasChild(identifier)) {  
 **val** data = snapshot.child(identifier).getValue(LightSettings::**class**.*java*)  
 lightSettings = data!!  
 }  
 replyToRequest(request, lightSettings)  
 noSettingsTimer.cancel()  
 }  
  
 **override fun** onCancelled(p0: DatabaseError?) {  
 **if** (*DEBUG*) Log.d(**TAG**, **"(44, GreenwaveModel.kt) onCancelled firebase: $**p0**"**)  
 }  
 })  
}

В методе получается доступ к БД, заводится таймер, для сигнализирования о том, что настроек нет, навешивается обработчик на получение данных по определенной ссылке. Если настройки есть, то обработчик сработает, распарсит настройки и вернет их по определенному callback’у. Но этот обработчик может висеть бесконечно долго, если данных на том конце нет. Тогда через секунду сработает таймер, остановит ожидание и вернет пустой объект настроек.

Для записи настроек все немного проще, потому что не приходится ждать ничего в отдельных обработчиках.

**override fun** updateLightSettingsInRemoteDb(light: LightSettings) {  
 **val** lightsRef = FirebaseDatabaseSingletone.getFirebaseInstance().getReference(*LIGHTS\_REFERENCE\_FIREBASE*)  
 lightsRef.child(light.*identifier*).setValue(light)  
}

### Расчет рекомендуемой скорости

При получении настроек, надо сделать расчет скорости, чтобы давать пользователю рекомендации. Считается расстояние до светофора, его полный цикл, и время, прошедшее с начала зеленого света.

**val** distance = light.**location**.distanceTo(**lastLoc**)  
  
**val** cycle = light.**settings**.*greenCycle* + light.**settings**.*redCycle***val** diff = (((Date().*time* - light.**settings**.*startOfMeasurement*) / 1000) % cycle).toInt()  
  
**var** timeToGreen = findOptimalTimeToGreen(light, diff, distance, cycle)

Время до зеленого света ищется максимально близкое к 5 км/ч – средней скорости пешехода.

Скорость считается по стандартной формуле

**private fun** calculateRecommendedSpeed(distance: Float, timeToGreen: Int): Double {  
 **return** distance / timeToGreen.toDouble()  
}

### Автоматическое обновление информации для пользователя

Вся информация на главном экране интерактивная и постоянно меняется. Этого удалось достичь с помощью RxJava.

**subscription** = Observable.timer(*SECOND\_IN\_MILLIS*, TimeUnit.**MILLISECONDS**)  
 .observeOn(AndroidSchedulers.mainThread())  
 .repeat(timeToGreen.toLong())  
 .subscribe(**{** Log.v(**TAG**, **"time to green ${**timeToGreen - 1**}"**)  
 **view**.setTimeToGreen(--timeToGreen)  
 **view**.setRecommendedSpeed(…)  
 **if** (timeToGreen == 1) {  
 **view**.hideSuggestions()  
 }  
 **}**)

Timer работает в background тредах, но каждую секунду посылает на *mainThread* оповещения, то есть мы сможем изменять UI приложения, так как он может быть изменен только с основного треда, ведь он не является thread safe.

По такому же принципу построено обновление дистанции и текущий свет в информации о светофоре.

## Описание и обоснование метода организации входных и выходных данных

### Описание метода организации входных и выходных данных

Входными данными для данного продукта являются локация пользователя и настройки светофоров: зеленый цикл в секундах, красный цикл в секундах.

Выходные данные – клиентская часть приложения.

### Обоснование метода организации входных и выходных данных

Локация пользователя используется для поиска ближайших светофоров. Настройки светофоров позволяют давать рекомендации по скорости движения. Клиентская часть приложения позволяет пользователю читать рекомендательную информацию о его безостановочном движении.

## Описание и обоснование метода выбора технических и программных средств

Для работы необходим следующий состав технических средств:

1. Смартфон с операционной системой Android
2. GPS-навигация в смартфоне

Для работы программы необходим следующий состав программных средств:

1. Операционная система Android 4.2 или выше.
2. Библиотека Google Play Services v2 для доступа к Firebase.

# ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

## Предполагаемая потребность

Приложение должно заинтересовать жителей городов, в которых есть много перекрестков со светофорами. Программа поможет сэкономить время простоя на красном сигнале, а также позволит повысить безопасность движения, т.к. помогает избежать ситуации пересекания проезжей части на мигающий зеленый свет.

## Экономические преимущества продукта по сравнению с отечественными или зарубежными аналогами и образцами.

Аналогов не обнаружено.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Android Code That Scales, With MVP [Электронный ресурс] – Режим доступа: https://engineering.remind.com/android-code-that-scales/, свободный
2. Firebase Realtime Database [Электронный ресурс] – Режим доступа: https://firebase.google.com/docs/database/, свободный
3. Retrofit [Электронный ресурс] – Режим доступа: http://square.github.io/retrofit/, свободный
4. RxJava – Reactive Extensions for the JVM – a library for composing asynchronous and event-based programs using observable sequences for the Java VM. [Электронный ресурс] – Режим доступа: https://github.com/ReactiveX/RxJava, свободный
5. API v0.6 – OpenStreetMap Wiki [Электронный ресурс] – Режим доступа: https://wiki.openstreetmap.org/wiki/API\_v0.6, свободный
6. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
14. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

# ПРИЛОЖЕНИЕ 1

**ДИАГРАММА КЛАСОВ**

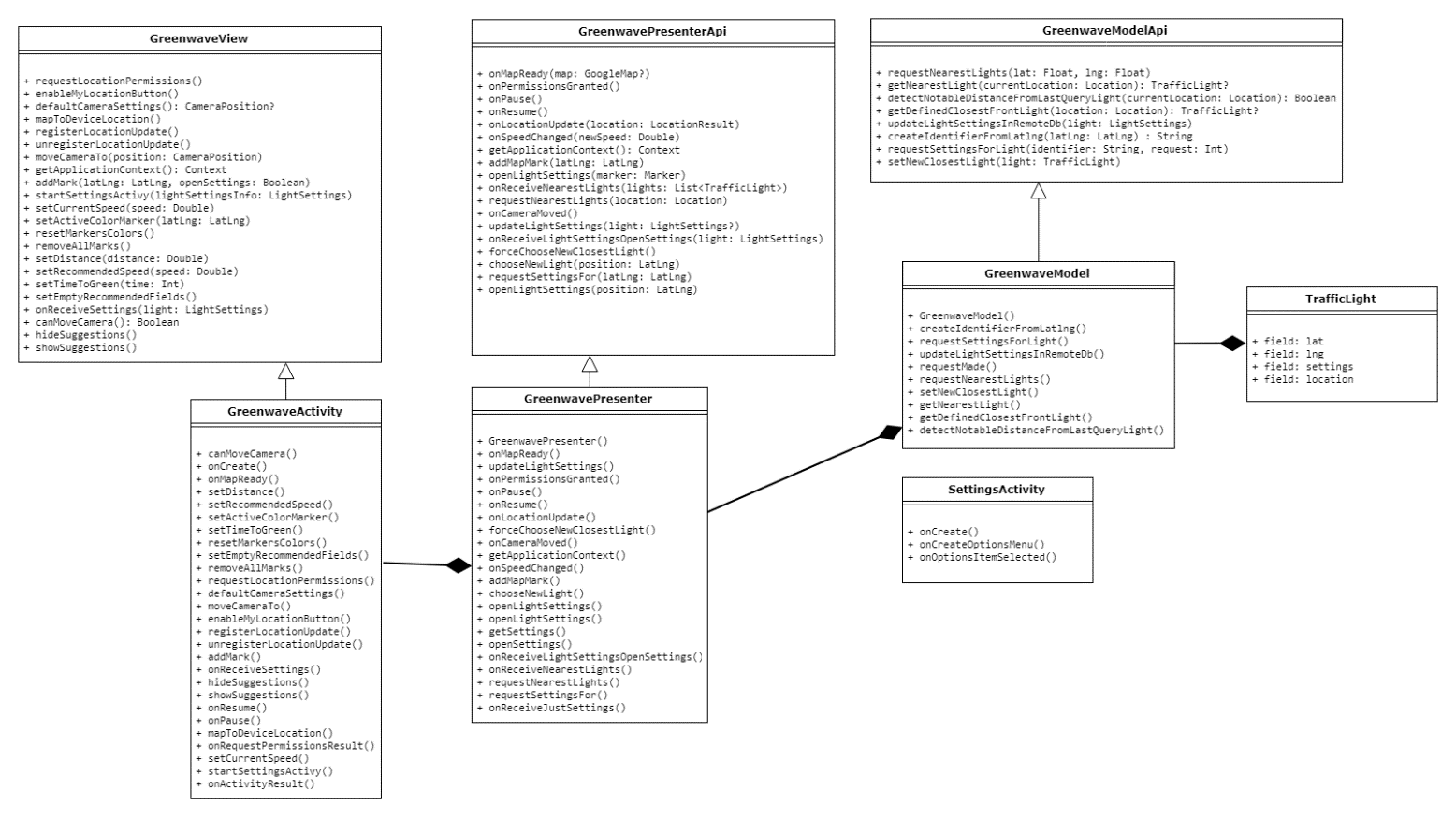


Рис. 4 - Диаграмма классов

# **ПРИЛОЖЕНИЕ** 2

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 2.1

Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Класс | Назначение |
| GreenwaveActivity | Класс основного компонента приложения |
| GreenwaveView | Интерфейс основного отображения |
| GreenwavePresenterApi | Интерфейс для Presenter в схеме MVP |
| GreenwavePresenter | Отвечает за логику приложения |
| GreenwaveModelApi | Интерфейс модели в схеме MVP |
| GreenwaveModel | Отвечает за данные в приложении |
| LightSettings | Настройки светофоров |
| Storage | Хранение настроек в памяти телефона |
| TrafficLight | Объект светофора |
| SettingsActivity | Активити настроек светофоров |
| CameraMovementLogicHelper | Помощник движений камеры |
| GreenwaveConstants | Все константы |
| MeanSpeed | Помощник в расчете средней скорости |

Таблица 2.2

Описание полей, методов и свойств класса GreenwaveActivity

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | Модификатор доступа | | | Тип | | Описание | |
| presenter | private final | | | interface nick.greenwave.GreenwavePresenterApi | | Презентер в схеме MVP | |
| TAG | private final | | | class java.lang.String | | Тег для логирования | |
| locationCallback | private | | | class com.google.android.gms.location.LocationCallback | | Callback для получения новой позиции | |
| fusedLocationClient$delegate | private final | | | interface kotlin.Lazy | | Клиент для Location Api | |
| requestNearestLights$delegate | private final | | | interface kotlin.Lazy | | Запрос ближайших светофоров | |
| requestClosestLight$delegate | private final | | | interface kotlin.Lazy | | Запрос одного ближайшего светофора | |
| speedView$delegate | private final | | | interface kotlin.Lazy | | Отображение скорости | |
| recommendedSpeed$delegate | private final | | | interface kotlin.Lazy | | Отображение рекомендуемой скорости | |
| recommendedLabel$delegate | private final | | | interface kotlin.Lazy | | Надпись "рекомендуемая скорость" | |
| recommendedUnits$delegate | private final | | | interface kotlin.Lazy | | Единицы измерения скорости | |
| remainingDistance$delegate | private final | | | interface kotlin.Lazy | | Отображение оставшейся дистанции | |
| remainingDistanceLabel$delegate | private final | | | interface kotlin.Lazy | | Надпись "distance" | |
| remainingDistanceUnits$delegate | private final | | | interface kotlin.Lazy | | Единицы измерения дистанции | |
| lightInfoView$delegate | private final | | | interface kotlin.Lazy | | Отображение информации о светофоре | |
| redCycle$delegate | private final | | | interface kotlin.Lazy | | Отображение красного цикла | |
| greenCycle$delegate | private final | | | interface kotlin.Lazy | | Отображение зеленого цикла | |
| current$delegate | private final | | | interface kotlin.Lazy | | Отображение текущей скорости | |
| select$delegate | private final | | | interface kotlin.Lazy | | Кнопка Выбрать светофор | |
| settings$delegate | private final | | | interface kotlin.Lazy | | Кнопка Настройки светофора | |
| lastMarkerClicked | private | | | class com.google.android.gms.maps.model.Marker | | Последний нажатый маркер | |
| followUser | private | | | boolean | | Следовать за положением пользователя | |
| stopSelectedMarkerTimer | private | | | boolean | | Флаг для остановки таймера выбранного маркера | |
| timeToGreen$delegate | private final | | | interface kotlin.Lazy | | Отображение времени до зеленого света | |
| timeToGreenLabel$delegate | private final | | | interface kotlin.Lazy | | Надпись Время до зеленого света | |
| timeToGreenUnits$delegate | private final | | | interface kotlin.Lazy | | Единицы измерения времени до зеленого света | |
| map | private | | | class com.google.android.gms.maps.GoogleMap | | Карта | |
| mCameraPosition | private | | | class com.google.android.gms.maps.model.CameraPosition | | Позиция камеры | |
| markers | private final | | | class java.util.ArrayList | | Все добавленные маркеры | |
| selectedMarkerCurrentLight | private | | | interface io.reactivex.disposables.Disposable | | Автообновление информационного окна | |
| locationUpdateRequest$delegate | private final | | | interface kotlin.Lazy | | Запрос на обновление локации | |
| Методы | | | | | | | |
| Имя | | Модификатор доступа | Тип | | Аргументы | | Назначение |
| removeAllMarks | | public | void | |  | | убрать все маркеры |
| onCreate | | protected | void | | Bundle | | Создание окна |
| addMark | | public | void | | LatLng, boolean | | Добавить маркер |
| moveCameraTo | | public | void | | CameraPosition | | Подвинуть камеру |
| canMoveCamera | | public | boolean | |  | | Можно ли двигать камеру |
| onMapReady | | public | void | | GoogleMap | | При готовности карты |
| hideSuggestions | | public | void | |  | | Скрыть рекомендации |
| registerLocationUpdate | | public | void | |  | | Начать получать обновление локации |
| setEmptyRecommendedFields | | public | void | |  | | Занулить все рекомендации |
| requestLocationPermissions | | public | void | |  | | запрос на использование геолокации |
| mapToDeviceLocation | | public | void | |  | | Карту на позицию пользователя |
| setActiveColorMarker | | public | void | | LatLng | | Выбрать маркер активным |
| onRequestPermissionsResult | | public | void | | int, String[], int[] | | При получении разрешения на использование геолокации |
| onReceiveSettings | | public | void | | LightSettings | | При получнии настроек |
| defaultCameraSettings | | public | class com.google.android.gms.maps.model.CameraPosition | | Стандартные настройки камеры | | defaultCameraSettings |
| createLocationCallback | | private final | void | |  | | Создать запрос на локацию |
| showLightInfoView | | private final | boolean | | Marker | | Отобразить инфо о светофоре |
| hideLightInfoView | | private final | void | |  | | Скрыть инфо о светофоре |
| getDeviceLocation | | private final | class com.google.android.gms.tasks.Task | |  | | Получить локацию пользователя |
| unregisterLocationUpdate | | public | void | |  | | Перестать получать локацию |
| startSettingsActivy | | public | void | | LightSettings | | Открыть настройки |
| enableMyLocationButton | | public | void | |  | | Включить кнопку Моя позиция |
| resetMarkersColors | | public | void | |  | | Сбросить цвета маркеров |
| showSuggestions | | public | void | |  | | Показать рекомендации |
| setCurrentSpeed | | public | void | | double | | Установить текущую скорость |
| onResume | | protected | void | |  | | При возвращении в состояние Resumed |
| onPause | | protected | void | |  | | При переходе в состояние Paused |
| onActivityResult | | protected | void | | int, int, Intent | | При получении результата другой Activity |

Таблица 2.3

Описание полей методов и свойств класса GreenwaveView

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| canMoveCamera | public abstract | boolean |  | Можно ли двигать камеру |
| moveCameraTo | public abstract | void | CameraPosition | Подвинуть камеру |
| setTimeToGreen | public abstract | void | int | Установить время до зеленого |
| setCurrentSpeed | public abstract | void | double | Установить текущую скорость |
| setDistance | public abstract | void | double | Установить дистанцию |
| addMark | public abstract | void | LatLng, boolean | Добавить маркер |
| removeAllMarks | public abstract | void |  | Убрать все маркеры |
| showSuggestions | public abstract | void |  | Показать рекомендации |
| hideSuggestions | public abstract | void |  | Скрыть рекомендации |
| requestLocationPermissions | public abstract | void |  | Запросить разрешение на использование геолокации |
| unregisterLocationUpdate | public abstract | void |  | Перестать получать локацию |
| startSettingsActivy | public abstract | void | LightSettings | Открыть настройки |
| resetMarkersColors | public abstract | void |  | Сбросить цвета маркеров |
| defaultCameraSettings | public abstract | CameraPosition |  | Стандартные настройки камеры |
| getApplicationContext | public abstract | class android.content.Context |  | Получить Application Context |
| mapToDeviceLocation | public abstract | void |  | Карту к локации пользователя |
| setRecommendedSpeed | public abstract | void | double | Установить рекмендуемую скорость |
| registerLocationUpdate | public abstract | void |  | Получать локацию |
| onReceiveSettings | public abstract | void | LightSettings | При получении настроек |
| setEmptyRecommendedFields | public abstract | void |  | Занулить рекомендации |
| setActiveColorMarker | public abstract | void | LatLng | Установить активный светофор |
| enableMyLocationButton | public abstract | void |  | Включить кнопку Моя позиция |

Таблица 2.4

Описание полей методов и свойств класса GreenwavePresenterApi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| onSpeedChanged | public abstract | void | double | При смене скорости |
| onLocationUpdate | public abstract | void | LocationResult | При смене локации |
| onCameraMoved | public abstract | void |  | При движении камеры |
| addMapMark | public abstract | void | LatLng | Добавить маркер |
| chooseNewLight | public abstract | void | LatLng | Выбрать новый светофор |
| onReceiveLightSettingsOpenSettings | public abstract | void | LightSettings | При получении настроек светофора |
| onPause | public abstract | void |  | При переходе в Paused |
| onResume | public abstract | void |  | При переходе в Resumed |
| onMapReady | public abstract | void | GoogleMap | При готовности карты |
| forceChooseNewClosestLight | public abstract | void |  | Выбрать новый светофор специально |
| getApplicationContext | public abstract | class android.content.Context |  | Получить Application Context |
| requestNearestLights | public abstract | void | Location | Запросит ближайшие светофоры |
| onPermissionsGranted | public abstract | void |  | При получении разрешений |
| updateLightSettings | public abstract | void | LightSettings | Обновить настройки светофора |
| openLightSettings | public abstract | void | LatLng | Открыть настройки |
| openLightSettings | public abstract | void | Marker | Открыть настройки |
| onReceiveNearestLights | public abstract | void | List | При получении ближайших светофоров |
| requestSettingsFor | public abstract | void | LatLng | Запросить настройки |

Таблица 2.5

Описание полей методов и свойств класса GreenwavePresenter

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Описание | |
| lastLoc | | private | | class android.location.Location | | последняя позиция | |
| lastSpeed | | private | | double | | последняя скорость | |
| model | | private final | | interface nick.greenwave.GreenwaveModelApi | | Модель в MVP | |
| movementHelper | | private final | | class utils.CameraMovementLogicHelper | | помощник движения камеры | |
| needUpdateNearestLight | | private | | boolean | | Флаг для обновления ближайших светофоров | |
| meanSpeedHelper | | private | | class utils.MeanSpeed | | Средняя скорость | |
| subscription | | private | | interface io.reactivex.disposables.Disposable | | Автообновление интерфейса | |
| TAG | | private final | | class java.lang.String | | Тег для логирования | |
| userLightsStorage$delegate | | private final | | interface kotlin.Lazy | | Обращение к пользовательским светофорам | |
| view | | private final | | interface nick.greenwave.GreenwaveView | | View в MVP | |
| Методы | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| onSpeedChanged | public | | void | | double | | При смене скорости |
| getSettings | public | | void | | LightSettings | | Получить настройки |
| setDistanceTo | private final | | void | | TrafficLight | | Установить дистанцию |
| onMapReady | public | | void | | GoogleMap | | При готовности карты |
| onResume | public | | void | |  | | При переходе в Resumed |
| onPause | public | | void | |  | | При переходе в Paused |
| onCameraMoved | public | | void | |  | | При движении камеры |
| onLocationUpdate | public | | void | | LocationResult | | При смене локации |
| requestNearestLights | public | | void | | Location | | Запросит ближайшие светофоры |
| detectTimeToUpdateClosestLight | private final | | void | |  | | Определить время до обновления |
| forceChooseNewClosestLight | public | | void | |  | | Выбрать новый светофор специально |
| updateLightSettings | public | | void | | LightSettings | | Обновить настройки светофора |
| onPermissionsGranted | public | | void | |  | | При получении разрешений |
| moveCameraToLastLocation | private final | | void | |  | | Камеру к последней локации |
| onReceiveOneClosestLight | private final | | void | | TrafficLight | | Получить 1 светофор |
| updateClosestLightIfNeeded | private final | | void | |  | | Обновить настройки светофора |
| updateDistanceToClosestLight | private final | | void | |  | | Обновить дистанцию |
| calculateRecommendedSpeed | private final | | double | | float, int | | Расчитать рекомендуему скорость |
| getApplicationContext | public | | class android.content.Context | |  | | Получить Application Context |
| findOptimalTimeToGreen | private final | | int | | TrafficLight, int, float, int | | Найти оптимальное время до зеленого света |
| addMapMark | public | | void | | LatLng | | Добавить маркер |
| chooseNewLight | public | | void | | LatLng | | Выбрать новый светофор |
| getView | public final | | interface nick.greenwave.GreenwaveView | |  | | Получить View в схеме MVP |
| openSettings | public | | void | | LightSettings | | Открыть настройки |
| requestSettingsFor | public | | void | | LatLng | | Запросить настройки светофора |
| openLightSettings | public | | void | | Marker | | Открыть настройки |
| openLightSettings | public | | void | | LatLng | | Открыть настройки |
| onReceiveJustSettings | public | | void | | LightSettings | | При получении настроек светофора |
| onReceiveNearestLights | public | | void | | List | | При получении ближайших светофоров |
| getUserLightsStorage | private final | | class nick.greenwave.data.Storage | |  | | Получить пользовательские светофоры |
| onReceiveLightSettingsOpenSettings | public | | void | | LightSettings | | При получении настроек светофора |

Таблица 2.6

Описание полей методов и свойств класса GreenwaveModelApi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| detectNotableDistanceFromLastQueryLight | public abstract | boolean | Location | Пройдено ли большое расстояние |
| getNearestLight | public abstract | class nick.greenwave.data.TrafficLight | Location | Получить ближайший светофор |
| requestSettingsForLight | public abstract | void | String, int | Запросить настройки |
| requestNearestLights | public abstract | void | float, float | Запросить светофоры |
| updateLightSettingsInRemoteDb | public abstract | void | LightSettings | Обновить настройки в удаленной БД |
| getDefinedClosestFrontLight | public abstract | class nick.greenwave.data.TrafficLight | Location | Получить ближайший светофор |
| createIdentifierFromLatlng | public abstract | class java.lang.String | LatLng | Создать идентификатор для БД |
| setNewClosestLight | public abstract | void | TrafficLight | Установить ближайший светофор |

Таблица 2.7

Описание полей методов и свойств класса GreenwaveModel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поля | | | | | |
| Имя | Модификатор доступа | | Тип | | Описание |
| TAG | private final | | class java.lang.String | | Тег для логирования |
| nearestLights | private | | interface java.util.List | | ближайшие светофоры |
| osmService$delegate | private final | | interface kotlin.Lazy | | Сервис OSM |
| lastQueryLightLocation | private | | class android.location.Location | | последняя локация запроса светофоров |
| closestLight | private | | class nick.greenwave.data.TrafficLight | | ближайший светофор |
| userLightsStorage$delegate | private final | | interface kotlin.Lazy | | Пользовательские светофоры |
| requestMade | private | | boolean | | Флаг был ли сделан запрос на светофоры |
| presenter | private final | | interface nick.greenwave.GreenwavePresenterApi | | Presenter в MVP |
| Методы | | | | | |
| Имя | | Модификатор доступа | Тип | Аргументы | Назначение |
| detectNotableDistanceFromLastQueryLight | | public | boolean | Location | Пройдено ли большое расстояние |
| getOsmService | | private final | interface utils.OsmService |  | Получить Сервис OSM |
| createQuery | | private final | class java.lang.String | float, float | Создать запрос для OSM API |
| replyToRequest | | private final | void | int, LightSettings | Ответить на запрос получения настроек |
| requestNearestLightsFromApi | | private final | void | float, float | Запросить ближайшие светофоры из API |
| requestSettingsForLight | | public | void | String, int | Запросить настройки |
| createIdentifierFromLatlng | | public | class java.lang.String | LatLng | Создать идентификатор для БД |
| updateLightSettingsInRemoteDb | | public | void | LightSettings | Обновить настройки в удаленной БД |
| requestNearestLights | | public | void | float, float | Получить ближайшие светофоры |
| onReceiveNearestLights | | private final | void | OsmQueryResult | при получении ответа от OSM API |
| setNewClosestLight | | public | void | TrafficLight | Установить ближайший светофор |
| getMovementVector | | private final | class kotlin.Pair | Location | Получить вектор движения |
| getDefinedClosestFrontLight | | public | class nick.greenwave.data.TrafficLight | Location | Получить ближайший светофор |
| isLightCloserThan | | private final | boolean | TrafficLight, double, Location | Ближе ли светофор чем |
| isLightInFront | | private final | boolean | TrafficLight, LatLng, Pair | Находится ли светофор перед пользователем |
| degreeToRadian | | private final | double | double | Градусы в радианы |
| getNearestLight | | public | class nick.greenwave.data.TrafficLight | Location | Получить ближайший светофор |
| getDistance | | private final | double | LatLng, LatLng | Получить дистанцию |

Таблица 2.8

Описание полей методов и свойств класса LightSettings

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Описание |
| greenCycle | public final | kotlin.Int | Зеленый цикл |
| identifier | public final | kotlin.String | Индентификатор |
| redCycle | public final | kotlin.Int | Красный цикл |
| startOfMeasurement | public final | kotlin.Long | Вреям начала зеленого света в UNIX |

Таблица 2.9

Описание полей методов и свойств класса Storage

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Описание | |
| context | private final | | android.content.Context | | | Контекс Android | |
| preferences | private final | | android.content.SharedPreferences | | | Preferences Android | |
| Методы | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Аргументы | | Назначение |
| createKey | | private final | | String | nick.greenwave.data.TrafficLight | | Создать ключ |
| createValue | | private final | | String | TrafficLight | | Создать значение |
| getAllLights() | | public final | | List<TrafficLight> |  | | Получить все светофоры |
| parseValue | | public final | | TrafficLight | String, String | | Распарсить значение |
| saveToPreferences | | public final | |  | TrafficLight | | Сохранить в Preferences |

Таблица 2.10

Описание полей методов и свойств класса TrafficLight

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Описание |
| lat | public final | Double | Latitude |
| lng | public final | Double | Longtitude |
| location | public final | Location | Локация |
| settings | public final | LightSettings | Настройки |

Таблица 2.11

Описание полей методов и свойств класса SettingsActivity

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | | Описание |
| TAG | private final | | class java.lang.String | | | | Тег для логирования |
| lightSettingsInfo | private | | class nick.greenwave.data.dto.LightSettings | | | | Настройки светофора |
| greenCycle$delegate | private final | | interface kotlin.Lazy | | | | Отображение зеленого цикла |
| redCycle$delegate | private final | | interface kotlin.Lazy | | | | Отображение красного цикла |
| currentLight$delegate | private final | | interface kotlin.Lazy | | | | Отображение текущего света |
| Методы | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Аргументы | Назначение | |
| onCreateOptionsMenu | | public | | boolean | Menu | При создании меню | |
| onOptionsItemSelected | | public | | boolean | MenuItem | При выборе пункта меню | |
| showErrorDialog | | private final | | void |  | Показать ошибку | |
| onCreate | | protected | | void | Bundle | При создании окна | |
| setGreenCycle | | private final | | void | int | Установить зеленый цикл | |
| setNewSettings | | private final | | void |  | Установить новые настройки | |
| saveNewSettings | | private final | | void |  | Сохранить новые настройки | |
| setRedCycle | | private final | | void | int | Установить красный цицкл | |

Таблица 2.12

Продолжение таблицы 2.11

Описание полей методов и свойств класса CameraMovementLogicHelper

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | | Описание |
| TAG | private final | | class java.lang.String | | | | Тег для логирования |
| isMoving | private | | boolean | | | | флаг движения |
| lastMove | private | | long | | | | Время последнего движения |
| Методы | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Аргументы | Назначение | |
| canMoveCamera | | public final | | boolean |  | Можно ли двигать камеру | |
| startMovement | | public final | | void |  | Начать движение | |

Таблица 2.13

Описание полей методов и свойств класса GreenwaveConstants

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Описание |
| SECOND\_IN\_MILLIS | public static final | long | секунды в мс |
| MINUNE\_IN\_MILLIS | public static final | long | минуты в мс |
| STANDARD\_ZOOM | public static final | float | стандартный зум |
| MEDIUM\_ZOOM | public static final | float | средний зум |
| STANDARD\_TILT | public static final | float | стандартный наклон |
| STANDARD\_LOCATION | private static final | LatLng | стандартная локация |
| LOCATION\_UPDATE\_INTERVAL | public static final | long | период обновления локации |
| LOCATION\_PERMISSION | private static final | class java.lang.String | запрос на локацию |
| LOCATION\_PERMISSION\_REQUEST | private static final | int | код для запроса |
| SPEED\_UPDATE\_INTERVAL | private static final | long | период обновления скорости |
| MAX\_LOCATIONS\_IN\_HISTORY | private static final | int | количество локаций в истории |
| LAST\_2\_SPEED\_MEASURMENT | private static final | int | последние 2 замера скорости |
| ALL\_MEAN\_SPEED\_MEASURMENT | private static final | int | код для средней скорости |
| EXTRAS\_LIGHT\_INFO | private static final | class java.lang.String | ключ для запроса настроек |
| OVERPASS\_QUERY\_DELIMETER | private static final | class java.lang.String | разделитель в запросе OSM |
| OVERPASS\_QUERY | private static final | class java.lang.String | запрос OSM |
| NEAREST\_LIGHTS\_MARGIN | private static final | float | отступ ближайших светофоров |
| NEAREST\_LIGHT\_DISTANCE | private static final | double | расстояние до ближайшего свтофора |
| MARKER\_EPSILON | private static final | float | отклонение маркера |
| NOTABLE\_DISTANCE | private static final | double | дистанция автообновления светофоров |
| LIGHTS\_REFERENCE\_FIREBASE | private static final | class java.lang.String | путь в удаленной БД |
| TIMER\_NAME\_GREEN | private static final | class java.lang.String | имя таймера для зеленого света |
| REQUEST\_FIREBASE\_CLOSEST\_LIGHT\_SETTINGS | private static final | int | код запроса настроек |
| REQUEST\_FIREBASE\_SETTINGS\_TO\_OPEN\_SETTINGS | private static final | int | код запроса настроек |
| REQUEST\_FIREBASE\_JUST\_SETTINGS | private static final | int | код запроса настроек |

Таблица 2.14

Описание полей методов и свойств класса CameraMovementLogicHelper

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | | Описание |
| speeds | private | | class java.util.ArrayList | | | | Последние скорости |
| Методы | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Аргументы | Назначение | |
| getMeanSpeed | | public final | | double |  | Получить среднюю скорость | |
| addSpeed | | public final | | void | double | Добавить скорость | |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |