Работа с картой в каршеринге

Евгений Петриёв | делимобиль

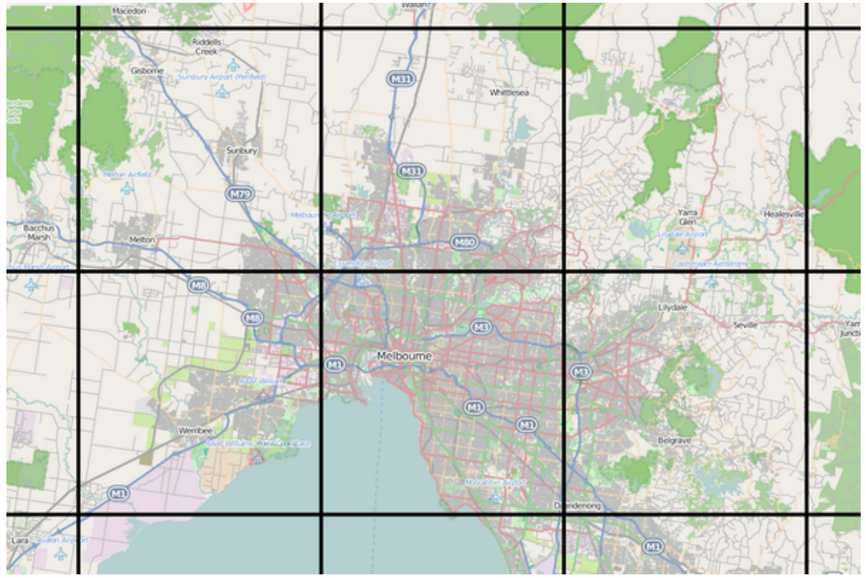
Содержание

- роль карт в приложении каршеринга
- работа с разными картографическими сервисами
- улучшение производительности карты
- можно ли построить универсальный интерфейс?
- Q&A

Терминология

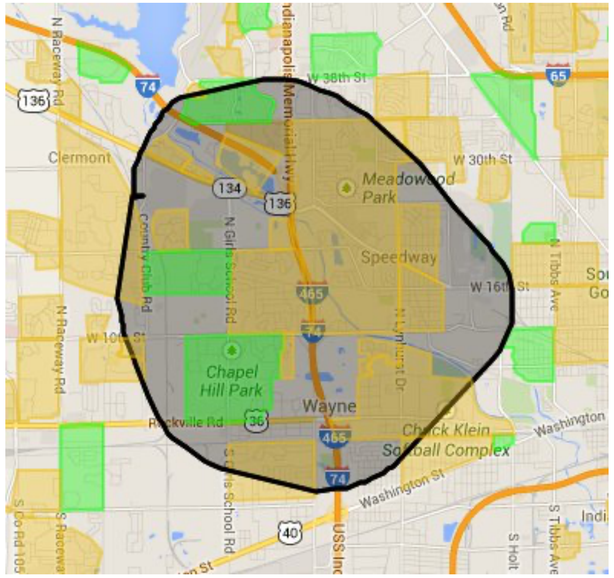
Tile

квадратное изображение из которых состоит карта



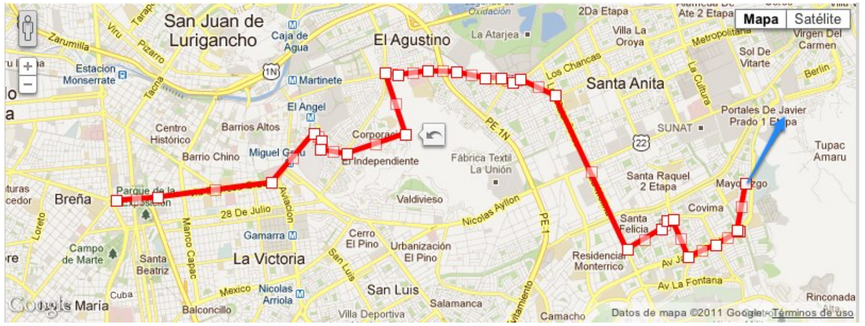
Polygon

замкнутый контур с заливкой



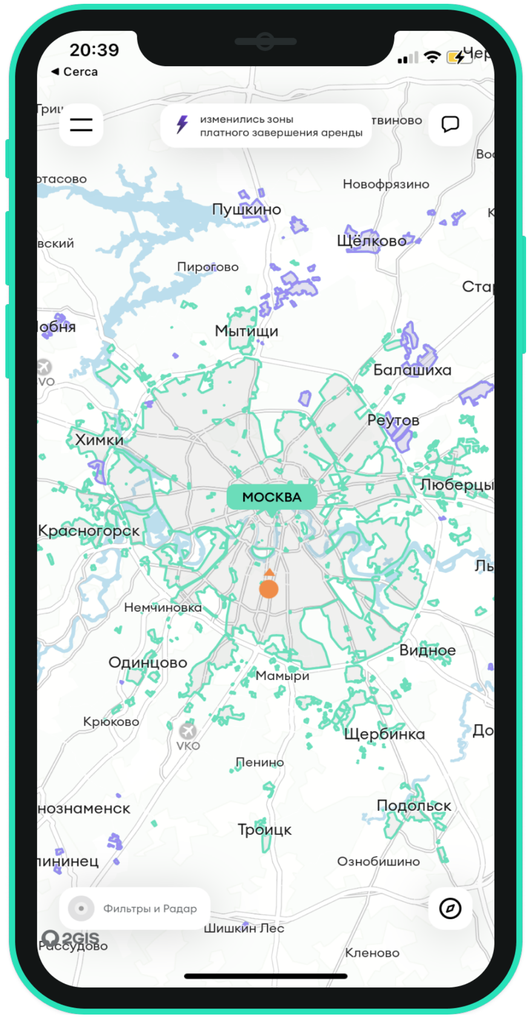
Polyline

линия, соединяющая точки на карте

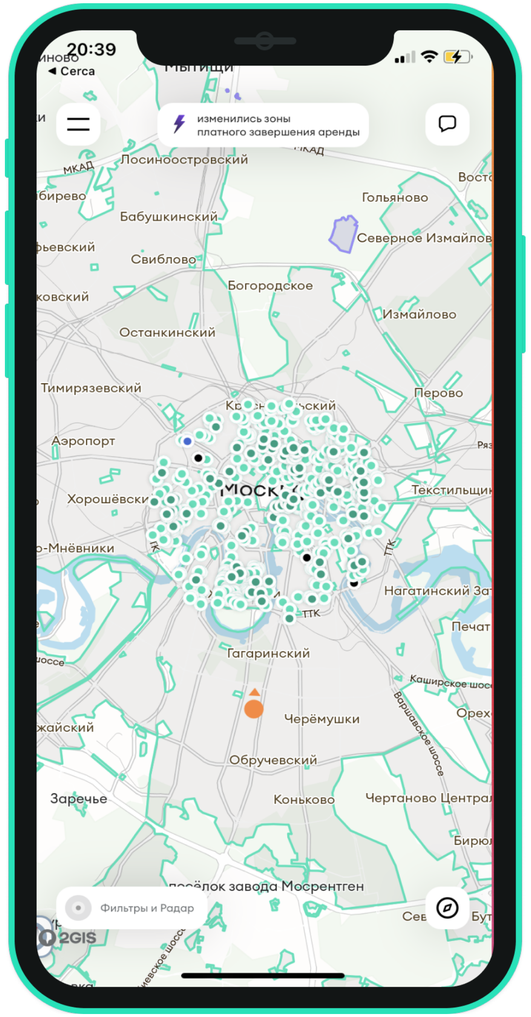


Карта в каршеринге

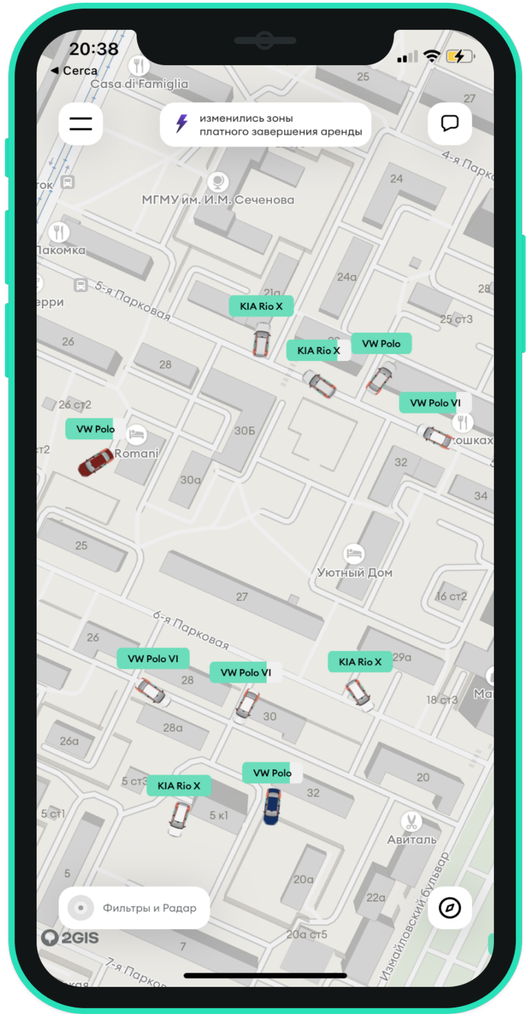
пины городов



пины автомобилей

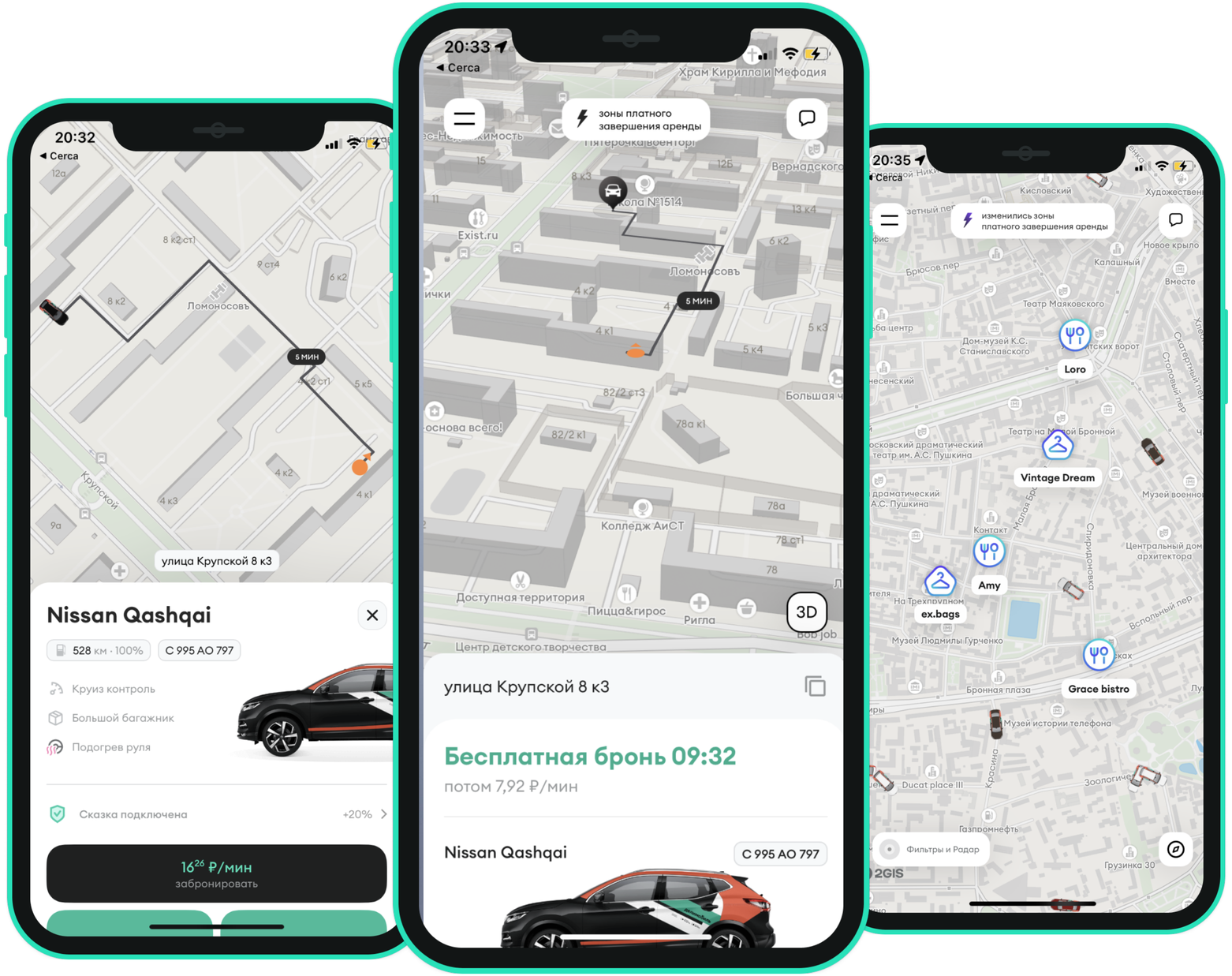


пины автомобилей



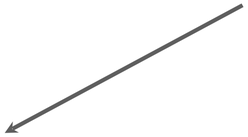
Карта в каршеринге

- маршрут
- 3D режим
- POI на карте



Google maps

GoogleMap



MapView

```
val marker = map.addMarker(MarkerOptions()
        .position(markerInfo.position)
        .icon(markerInfo.icon)
        .rotation(markerInfo.rotation)
        .zIndex(MarkerZs.CAR_Z)
        .title(markerInfo.snippet))
```

https://github.com/googlemaps/android-maps-utils

```
let marker = GMSMarker(position:
markerInfo.position)
marker.icon = markerInfo.icon
marker.rotation = marketInfo.rotation
marker.zIndex = MarkerZs.CAR_Z
marker.title = markerInfo.snippet
marker.map = gMapView
```

https://github.com/googlemaps/google-maps-ios-utils

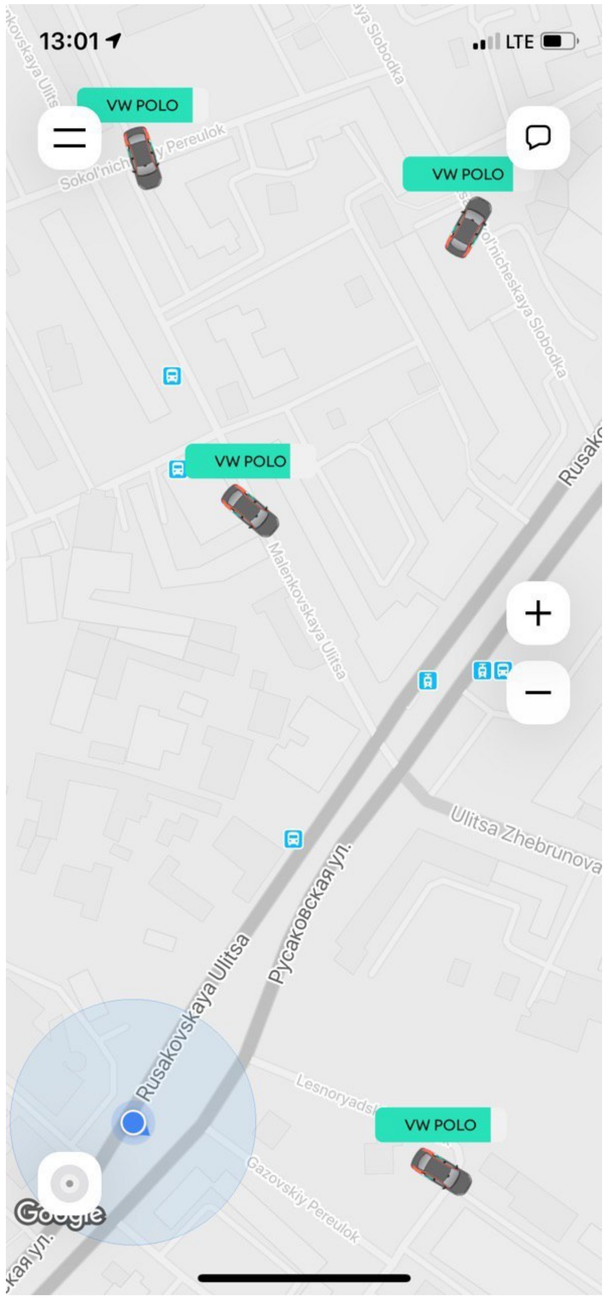
Google maps

Плюсы

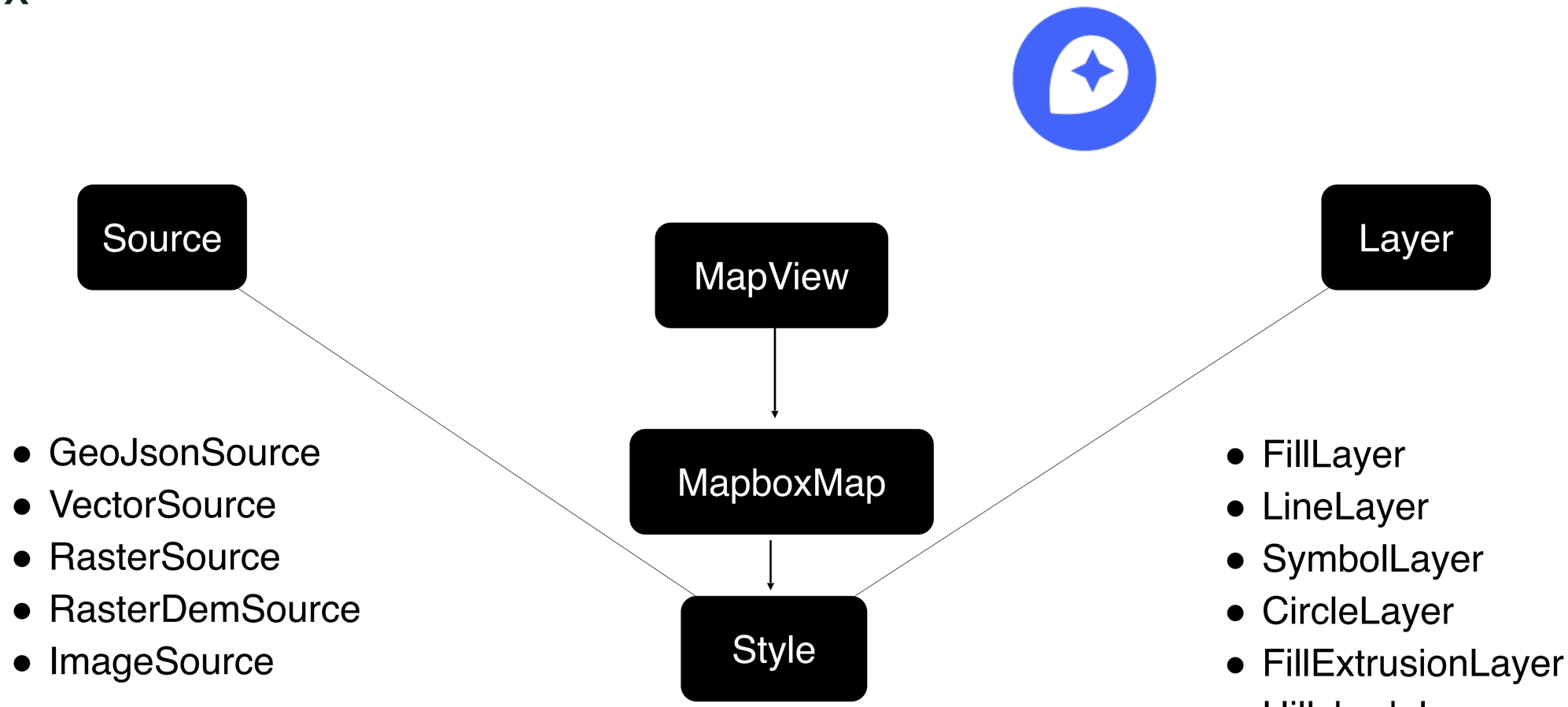
- самая популярная карта на планете
- легкая интеграция

Минусы

- тормозит на слабых девайсах
- не работает без Google Play Services
- Не хватает детализации в регионах



Mapbox


Source
MapView
Layer
MapboxMap
Style
GeoJsonSource
VectorSource
RasterSource
RasterDemSource
ImageSource
FillLayer
LineLayer
SymbolLayer
CircleLayer
FillExtrusionLayer
HillshadeLayer
HeatmapLayer
RasterLayer
SkyLayer
BackgroundLayer

Mapbox

источники данных карты

```
const val SOURCE_ZONE_DRIVING = "deli-zone-driving"
const val SOURCE_ZONE_PARKING = "deli-zone-parking"
const val SOURCE_ZONE_PAID = "deli-zone-paid"
const val SOURCE_ZONE_PINS = "deli-zone-pins"
const val SOURCE_REGIONS = "deli-regions"
const val SOURCE_GAS_STATIONS = "deli-gas-stations"
const val SOURCE_CARS = "deli-cars"
const val SOURCE_FUEL = "deli-cars-fuel"
const val SOURCE_LOCKS = "deli-cars-locks"
const val SOURCE_ROUTE = "deli-route"
const val SOURCE_ROUTE_DISTANCE = "deli-route-distance"
const val SOURCE_FIX_AREA = "deli-fix-area"
const val SOURCE_CAR_HIGHLIGHT = "deli-cars-highlight"
const val SOURCE_TRIP_PINS = "deli-trip-pins"
```

соответствующие им слои

```
const val LAYER_REGIONS = "deli-regions"
const val LAYER_ZONE_DRIVING_STROKE = "deli-zone-driving-stroke"
const val LAYER_ZONE_PARKING_FILL = "deli-zone-parking-fill"
const val LAYER_ZONE_PARKING_STROKE = "deli-zone-parking-stroke"
const val LAYER_ZONE_PAID_FILL = "deli-zone-paid-fill"
const val LAYER_ZONE_PINS = "deli-zone-pins"
const val LAYER_ZONE_PAID_STROKE = "deli-zone-paid-stroke"
const val LAYER_GAS_STATIONS = "deli-gas-stations"
const val LAYER_CARS = "deli-cars"
const val LAYER_FUEL = "deli-fuel"
const val LAYER_LOCKS = "deli-locks"
const val LAYER_ROUTE = "deli-route"
const val LAYER_ROUTE_DISTANCE = "deli-route-distance"
const val LAYER_FIX_AREA = "deli-fix-area"
const val LAYER_FIX_STROKE = "deli-fix-stroke"
const val LAYER_CAR_HIGHLIGHT = "deli-car-highlight"
const val LAYER_TRIP_PINS = "deli-trip-pins"
```

Mapbox

добавление слоя на карту

```
map.onStyle {
        val layerId = "some_layer_id"
        val sourceId = "some_source_id"

        val source = GeoJsonSource(sourceId)

        val featureCollection = FeatureCollection.fromFeatures(
            listOf(
                Feature.fromGeometry(Point.fromLngLat(37.705, 55.791)),
                Feature.fromGeometry(Point.fromLngLat(14.270, 40.851))
            )
        )

        source.setGeoJson(featureCollection)

        val layer = SymbolLayer(layerId, sourceId)

        addLayer(layer)
    }
```

Mapbox

подготовка объектов

```
val markersFeatures = markers.options.map {
        val pos = it.position
        val p = Point.fromLngLat(pos.lon, pos.lat)
        it.image?.let(icons::add)
        Feature.fromGeometry(p).apply {
            addStringProperty(PROPERTY_MARKER_ID, it.id)
            addStringProperty(PROPERTY_MARKER_ICON, it.image?.tag)
            addNumberProperty(PROPERTY_MARKER_ROTATION, it.rotation ?: 0f)
            addNumberProperty(PROPERTY_MARKER_ALPHA, it.alpha)
            addProperty(PROPERTY_MARKER_VERTICAL_OFFSET, JsonArray(2)
                .apply {
                    add(0f)
                    add(it.verticalOffset)
                }
            )

        }
    }
```

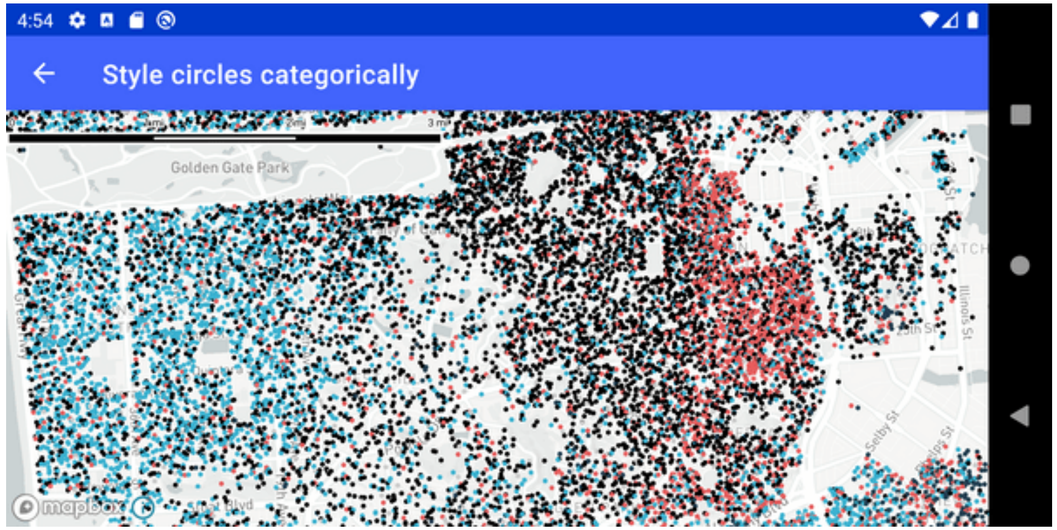
Mapbox

добавление объектов со свойствами

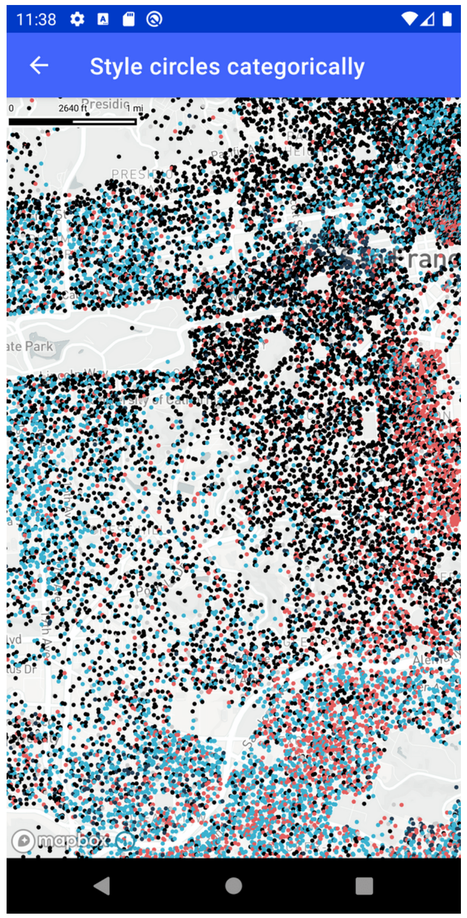
```
map.onStyle {
        icons.forEach {
            fetchIcon(it)
        }

        getSourceAs<GeoJsonSource>(sourceId)
            ?.setGeoJson(FeatureCollection.fromFeatures(markersFeatures))

        getLayer(layerId)?.setProperties(
            iconImage(Expression.get(PROPERTY_MARKER_ICON)),
            iconOpacity(Expression.get(PROPERTY_MARKER_ALPHA)),
            iconRotate(Expression.get(PROPERTY_MARKER_ROTATION)),
            iconOffset(Expression.get(PROPERTY_MARKER_VERTICAL_OFFSET)),
            iconIgnorePlacement(true),
            iconRotationAlignment(Property.ICON_ROTATION_ALIGNMENT_VIEWPORT)
        )
    }
```

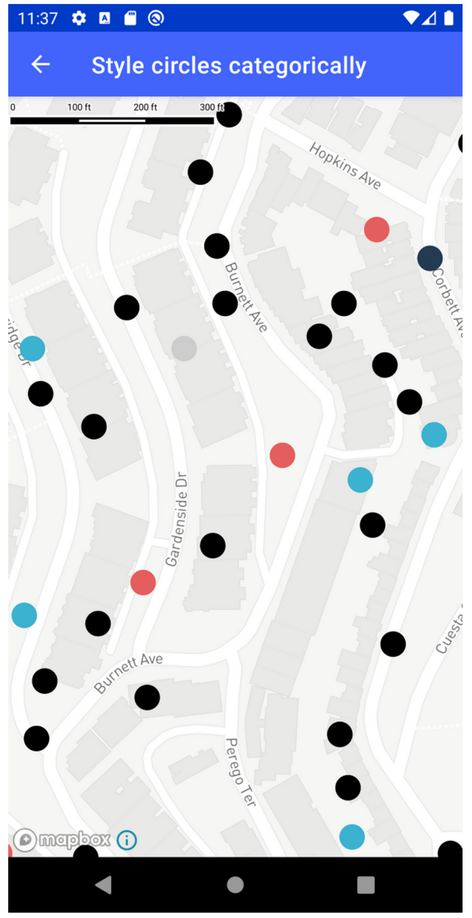
Mapbox Expressions

property expression



camera expression





Mapbox фейлы

нативный крэш в библиотеке mapbox

Crashed: Thread #1

SIGABRT 0x0000000000000000

```
Crashed: Thread :  SIGABRT  0x0000000000000000
#00 pc 0x7bf04f206c libc.so
#01 pc 0x7bf04f203c libc.so
#02 pc 0x7ae7109998 libmapbox-gl.so
#03 pc 0x7ae7109b08 libmapbox-gl.so
#04 pc 0x7ae71070c8 libmapbox-gl.so
#05 pc 0x7ae71066f4 libmapbox-gl.so
#06 pc 0x7ae7106650 libmapbox-gl.so
#07 pc 0x7ae6e2a808 libmapbox-gl.so
#08 pc 0x7ae6e2f6e0 libmapbox-gl.so
#09 pc 0x7ae6e3f9e0 libmapbox-gl.so
#10 pc 0x7ae6e37e1c libmapbox-gl.so
```

Mapbox фейлы

NO

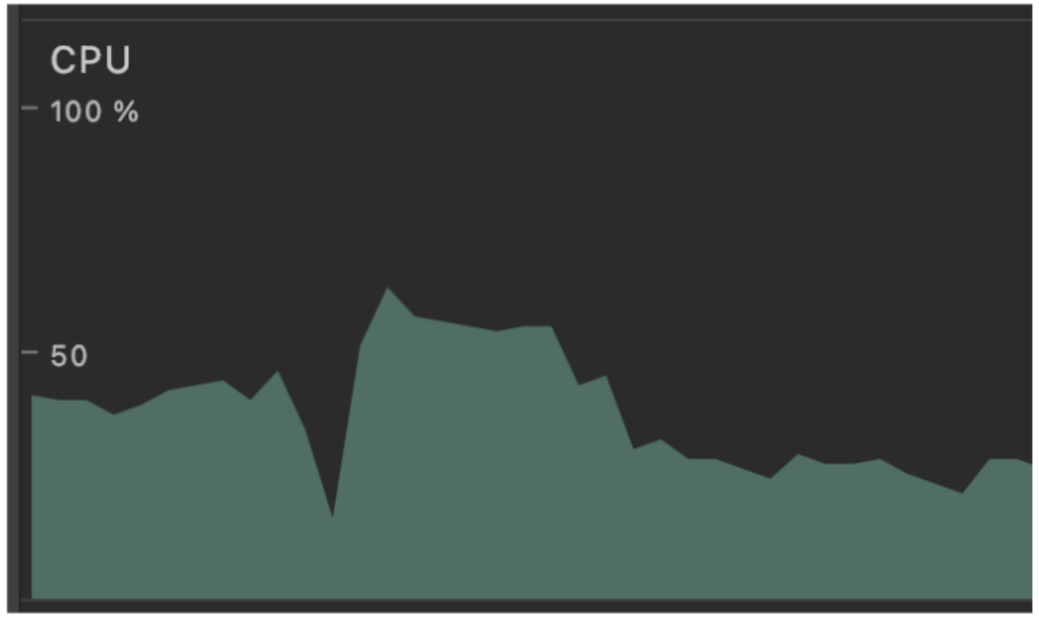
```
class MapFragment : Fragment() {

        ...

        override fun onDestroy() {
            super.onDestroy()
            vMap.onDestroy()
        }

        ...
    }
```

YES

```
class MapFragment : Fragment() {

        ...

        override fun onDestroyView() {
            super.onDestroyView()
            vMap.onDestroy()
        }

        ...
    }
```

Mapbox фейлы

Карта открыта



Закрыли карту

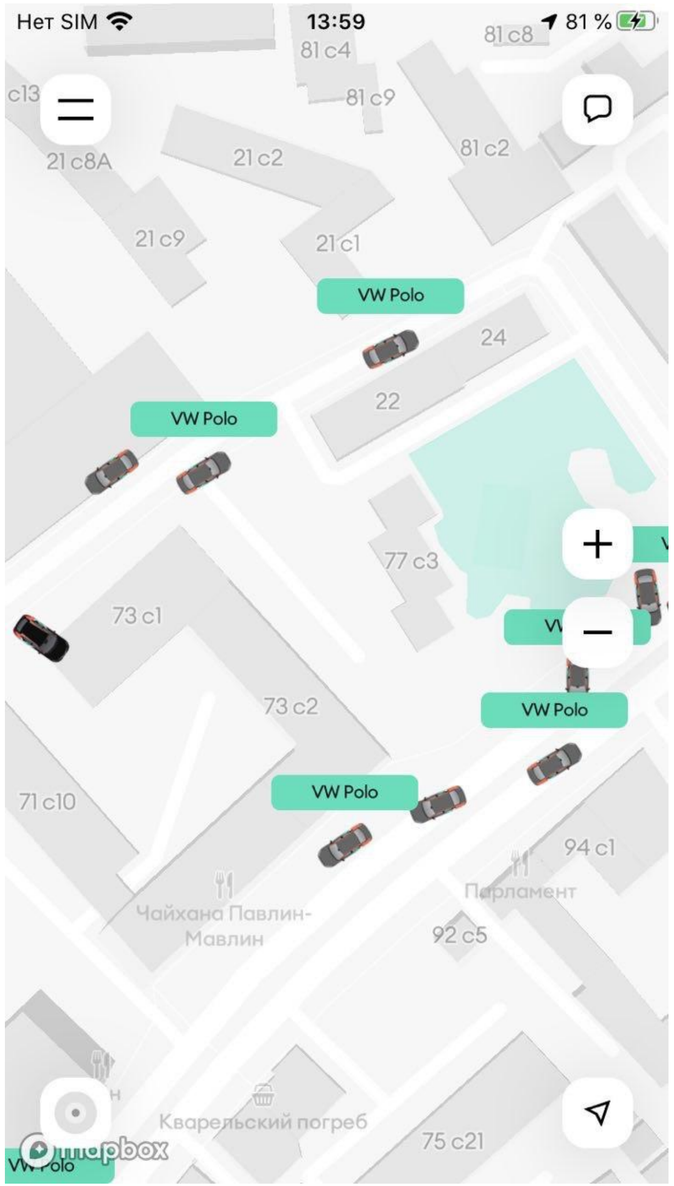
Mapbox

Плюсы

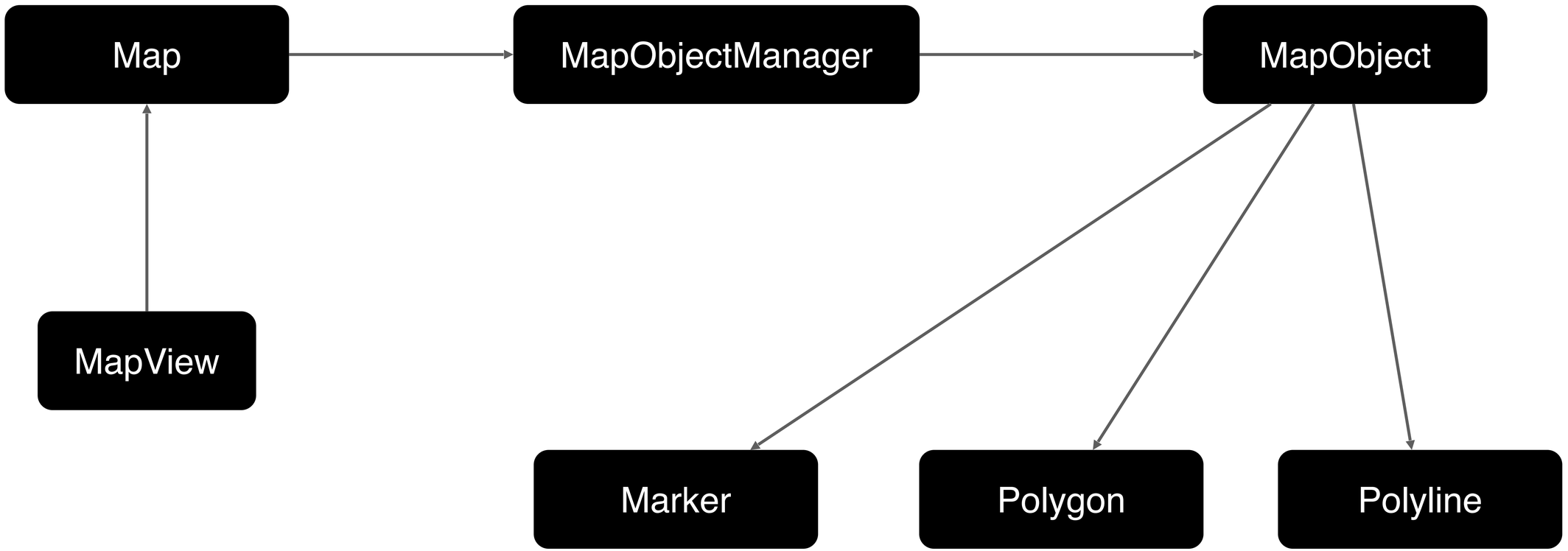
- не тормозит на слабых девайсах
- работает на Huawei
- мощный редактор стилей
- детально

Минусы

- высокий порог входа
- нагрузка на CPU
- авторизация при скачивании sdk



2GIS


Map
MapObjectManager
MapObject
MapView
Marker
Polygon
Polyline

2GIS

добавление объекта на карту

```
data class MarkerInfo(
    val lat: Double,
    val lon: Double,
    val rotation: Float,
    val bitmap: Bitmap,
    val zIndex: Int)

    val mapObjectManager = MapObjectManager(map = map, layerId = layerId)
    val marker = Marker(
        MarkerOptions(
            position = GeoPointWithElevation(
                latitude = markerInfo.lat,
                longitude = markerInfo.lon
            ),
            iconMapDirection = MapDirection(markerInfo.rotation.toDouble()),
            icon = imageFromBitmap(sdkContext, markerInfo.bitmap),
            zIndex = ZIndex(markerInfo.zIndex)
        )
    )
    mapObjectManager.addObject(marker)
```

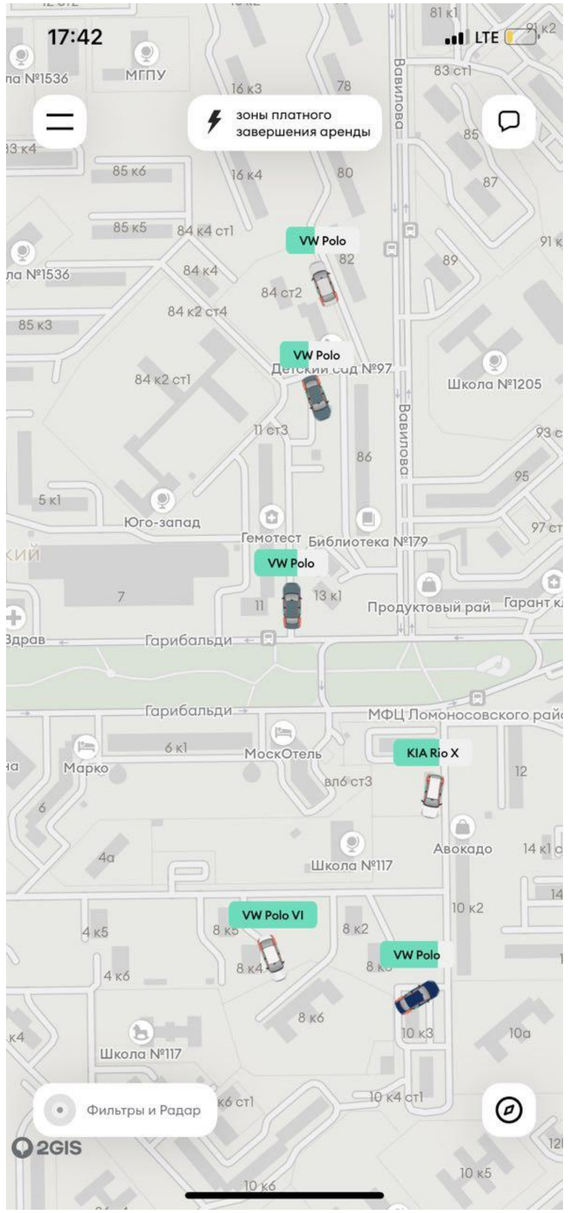
2GIS

Плюсы

- отечественное
- хорошая детализация
- локальная поддержка

Минусы

- старый Xcode
- ждём поддержку Metal
- мелкие баги
- не все фичи ещё поддерживаются



Оптимизации

- Минимизируем обращения к карте (idle, throttle)
- Ограничиваем кол-во объектов на карте
- Подготовка данных в другом потоке
- Кэширование объектов и изображений

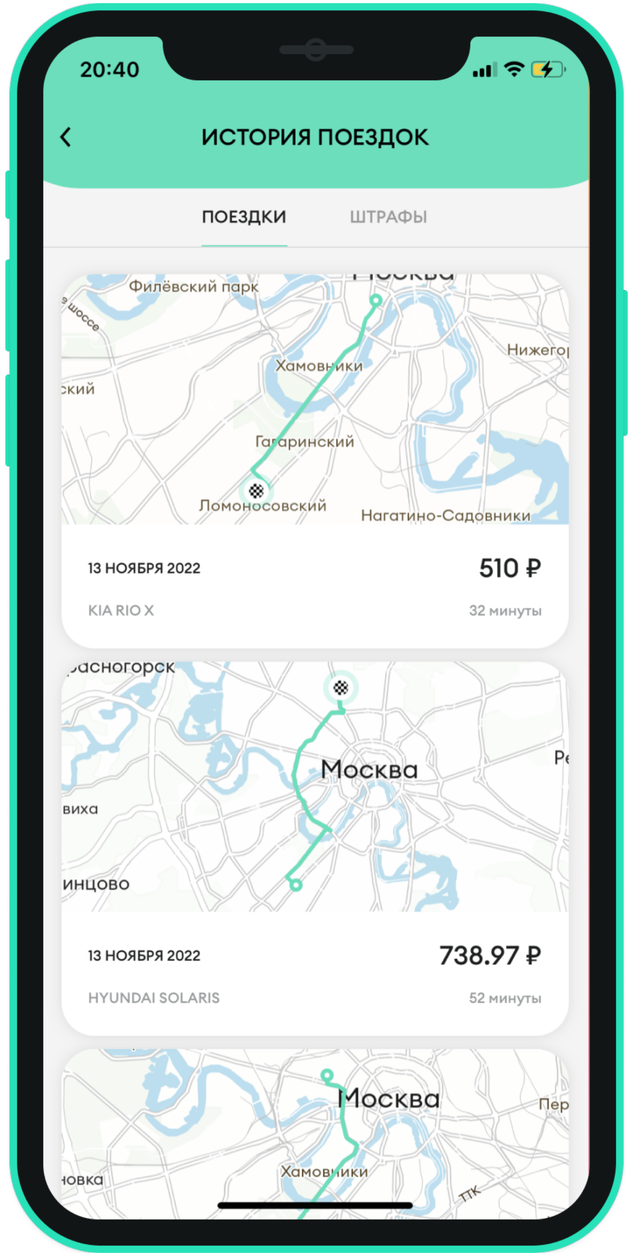
Карта в списках

takeSnapshot

скриншот фрагмента карты с нужными объектами

Static API

используется для получения изображения с картой



DistanceBetween

формула расчета расстояния между двумя точками // используется в Android SDK

```
fun distanceBetween(startLat: Double,
      startLon: Double, endLat: Double,
      endLon: Double): Float {
    var lat1 = startLat
    var lon1 = startLon
    var lat2 = endLat
    var lon2 = endLon
    val MAXITERS = 20
    // Convert lat/long to radians
    lat1 *= Math.PI / 180.0
    lat2 *= Math.PI / 180.0
    lon1 *= Math.PI / 180.0
    lon2 *= Math.PI / 180.0
    val a = 6378137.0 // WGS84 major axis
    val b = 6356752.3142 // WGS84 semi-major axis
    val f = (a - b) / a
    val aSqMinusBSqOverBSq = (a * a - b * b) / (b * b)
    val L = lon2 - lon1
    var A = 0.0
    val U1 = atan((1.0 - f) * tan(lat1))
    val U2 = atan((1.0 - f) * tan(lat2))
    val cosU1 = cos(U1)
    val cosU2 = cos(U2)
    val sinU1 = sin(U1)
    val sinU2 = sin(U2)
    val cosU1cosU2 = cosU1 * cosU2
    val sinU1sinU2 = sinU1 * sinU2
    var sigma = 0.0
    var deltaSigma = 0.0
    var cosSqAlpha = 0.0
    var cos2SM = 0.0
    var cosSigma = 0.0
    var sinSigma = 0.0
    var cosLambda = 0.0
    var sinLambda = 0.0
    var lambda = L // initial guess
    for (iter in 0 until MAXITERS) {
        val lambdaOrig = lambda
        cosLambda = cos(lambda)
        sinLambda = sin(lambda)
        val t1 = cosU2 * sinLambda
        val t2 = cosU1 * sinU2 - sinU1 * cosU2 * cosLambda
        val sinSqSigma = t1 * t1 + t2 * t2 // (14)
        sinSigma = Math.sqrt(sinSqSigma)
        cosSigma = sinU1sinU2 + cosU1cosU2 * cosLambda // (15)
        sigma = Math.atan2(sinSigma, cosSigma) // (16)
        val sinAlpha = if (sinSigma == 0.0) 0.0 else cosU1cosU2
             * sinLambda / sinSigma // (17)
        cosSqAlpha = 1.0 - sinAlpha * sinAlpha
        cos2SM = if (cosSqAlpha == 0.0) 0.0 else cosSigma - 2.0
             * sinU1sinU2 / cosSqAlpha // (18)
        val uSquared = cosSqAlpha * aSqMinusBSqOverBSq // defn
        A = 1 + uSquared / 16384.0 * (4096.0 + uSquared *
             (-768 + uSquared * (320.0 - 175.0 * uSquared)))
        val B = uSquared / 1024.0 *  (256.0 + uSquared *
             (-128.0 + uSquared * (74.0 - 47.0 * uSquared)))
        val C = f / 16.0 * cosSqAlpha * (4.0 + f * (
                4.0 - 3.0 * cosSqAlpha)) // (10)
        val cos2SMSq = cos2SM * cos2SM
        deltaSigma = B * sinSigma * (cos2SM + B / 4.0 * (cosSigma *
             (-1.0 + 2.0 * cos2SMSq) - B / 6.0 * cos2SM *
             (-3.0 + 4.0 * sinSigma * sinSigma) * (-3.0 + 4.0 * cos2SMSq)))
        lambda = L + (1.0 - C) * f * sinAlpha * (sigma + C *
             sinSigma * (cos2SM + C * cosSigma * (-1.0 + 2.0 * cos2SM *
cos2SM)))
        val delta = (lambda - lambdaOrig) / lambda
        if (abs(delta) < 1.0e-12) { break }
    }
    return (b * A * (sigma - deltaSigma)).toFloat()
}
```

Теорема Пифагора

```
let longFactor = 0.3

let dist1 = (lat1 - lat2) * (lat1 - lat2) + (long1 - long2) *

(long1 - long2) * longFactor
```

Смена вендора карт

Я построю абстракции и буду легко переключаться между разными картами

https://github.com/delimobil/deli_mapbox

У них так много отличий, переключиться обратно не получится(

https://github.com/delimobil/deli_2gis

Q&A

Tg: @epetriyov