Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Уфимский государственный нефтяной технический университет

Кафедра вычислительной техники и инженерной кибернетики

КУРСОВАЯ РАБОТА

по дисциплине

Разработка мобильных приложений

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ НАВИГАЦИИ ПО СТУДЕНЧЕСКОМУ ГОРОДКУ УГНТУ

Выполнили студенты группы БПО-21-01 К.Р. Хасиятулин

С.С. Газизова

Принял ст. преподаватель Е.В. Дружинская

Дата представления работы: \_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Результат: \_\_\_\_\_\_\_\_\_\_

Уфа, 2024 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc168339490)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc168339491)

[1.1 Предметная область 5](#_Toc168339492)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 6](#_Toc168339493)

[2.1 Функциональное моделирование 6](#_Toc168339494)

[2.1.1 Диаграмма вариантов использования 6](#_Toc168339495)

[2.1.2 Диаграмма деятельности 7](#_Toc168339496)

[2.3 Логическое моделирование 9](#_Toc168339497)

[2.4 Технологии 11](#_Toc168339498)

[ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 12](#_Toc168339499)

[3.1 Карта студенческого городка 12](#_Toc168339500)

[3.1.1 Навигационное меню 20](#_Toc168339501)

[3.1.1.1 Категория «Маршруты» 24](#_Toc168339502)

[3.1.1.2 Категория «Здания» 31](#_Toc168339503)

[3.1.1.3 Границы студенческого городка 31](#_Toc168339504)

[ЗАКЛЮЧЕНИЕ 34](#_Toc168339505)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 35](#_Toc168339506)

[ПРИЛОЖЕНИЕ А 36](#_Toc168339507)

[Ссылка на репозиторий GitHub 36](#_Toc168339508)

# ВВЕДЕНИЕ

**Актуальность.** В современном мире мобильные приложения становятся неотъемлемой частью повседневной жизни. Одной из наиболее востребованных функций таких приложений является навигация, особенно в больших и сложных пространствах, таких как студенческие городки. Для студентов, преподавателей и гостей университетского кампуса крайне важно быстро и точно находить нужные здания, аудитории, административные офисы и другие объекты инфраструктуры. Традиционные способы навигации, такие как печатные карты или статические электронные схемы, часто оказываются неудобными и устаревшими.

Использование мобильного приложения "Навигатор по студенческому городку", основанного на платформе Yandex MapKit, позволяет обеспечить интерактивность, актуальность данных и удобство использования. Таким образом, создание такого приложения отвечает потребностям современного студенческого сообщества в эффективной и удобной навигации.

**Целью курсовой работы** является разработка и реализация мобильного приложения "Навигатор по студенческому городку" с использованием Yandex MapKit. Приложение должно обеспечивать пользователей точной и актуальной картой университетского кампуса, предлагать оптимальные маршруты к нужным объектам.

В соответствии с целью было поставлено несколько **задач**, которые необходимо решить:

* Исследовать функциональные возможности Yandex MapKit и выбрать оптимальные для реализации проекта.
* Разработать структуру данных для представления объектов инфраструктуры студенческого городка.
* Реализовать интерфейс приложения, обеспечивающий интуитивно понятное взаимодействие пользователя с картой и навигационными функциями.
* Обеспечить обновление данных в реальном времени для поддержания актуальности информации.

**Объектом исследования** является разработка приложения на платформе Android, **предметом исследования** – мобильное приложение для навигации USPTU Map.

**Методы.** В курсовой работе были использованы такие методы, как анализ литературы по теме исследования, а также эмпирические (практические), такие как программирование и тестирование разработанного приложения.

**Структура.** Курсовая работа состоит из введения, трех глав, заключения, списка использованных источников и приложения.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Перед тем, как приступить к проектированию мобильного приложения, необходимо проанализировать предметную область, что позволит глубже понять потребности целевой аудитории и определить конечный функционал продукта.

## 1.1 Предметная область

Предметная область включает в себя множество процессов, связанных с определением местоположения, нахождением объектов студенческого городка и формированием маршрута.

Описанные выше процессы предметной области являются взаимосвязанными, а также имеют ряд недостатков при самостоятельном выполнении.

Реализация мобильного приложения USPTU Map предоставляет удобный функционал и возможности для выполнения описанных процессов максимально эффективным способом на текущий момент развития технологий геодезии.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

## 2.1 Функциональное моделирование

### 2.1.1 Диаграмма вариантов использования

Диаграмма вариантов использования позволяет смоделировать основные функциональные требования, рассмотренные в предыдущей главе.

Основным действующим лицом является потенциальный пользователь, которые осуществляет взаимодействие с приложением. Каждому пользователю доступен следующий функционал:

1. Определение местоположения:

* местоположение определяется при запуске приложения;
* процесс определения местоположения свидетельствует о том, что пользователь находится на территории студенческого городка.

2. Определение объектов карты и формирование маршрутов:

* объекты студенческого городка нанесены на карту и выделены специальным образом;
* маршруты формируется относительно двух объектов карты;
* маршруты формируются исключительно на территории студенческого городка;
* при попытке формирования маршрута за пределами студенческого городка, пользователь будет направлен в другие картографические сервисы;
* местоположение пользователя является динамическим объектом карты и так же подходит для формирования маршрута.

Итоговая диаграмма вариантов использования приложения «USPTU Map» представлена на рисунке 1.

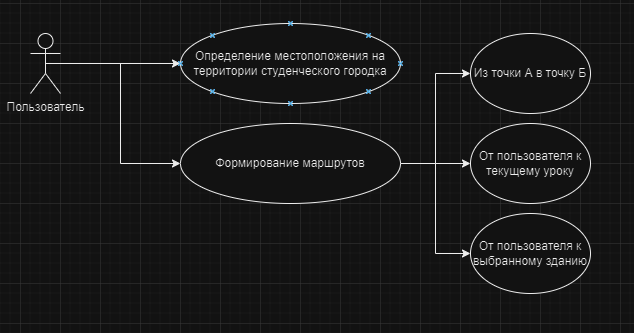


Рисунок 1 – Диаграмма вариантов использования мобильного приложения USPTU Map

### 2.1.2 Диаграмма деятельности

Диаграмма деятельности отражает динамические аспекты поведения системы. Отразим основные модули приложения с помощью них.

При входе в приложение программа регистрирует активную сессию в сервисе Yandex MapKit по предоставленному API ключу. Затем приложение посылает запрос на определение местоположения пользователя. Получив координаты, на карте создается метка, имеющая иконку, отображающая местоположение пользователя. Вместе с этим, индикатор уведомляет пользователя об определении местоположения.

После вышеописанных действий, пользователю становятся доступна функция формирования маршрутов. Пользователь выбирает объекты или конечный объект, после чего приложение отправляет запрос используя API. Сервис исследует наилучший маршрут от объекта до объекта и возвращает в приложение координаты точек для построения геометрической фигуры в виде полилинии, которая и является маршрутом на карте. Приложению остается нарисовать полилинию на карте из полученных точек координат.

Исключением для функции построении маршрутов является событие, когда пользователь находится за пределами студенческого городка. В этом случае, пользователю будет предложено воспользоваться другими картографическими сервисами, позволяющие ему добраться до студенческого городка.

Рассмотренный процесс приложения представлен на рисунке 2.

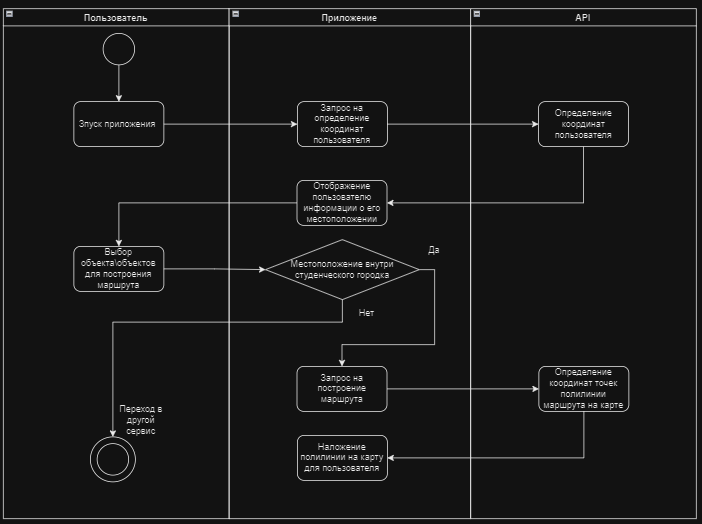


Рисунок 2 – Диаграмма деятельности типичной последовательности процесса формирования маршрута

## 2.3 Логическое моделирование

Приложение включает в себя несколько пакетов, каждый из которых отвечает за отдельный функционал:

1. map\_operations. Пакет отвечает за работу с элементами и процессами карты.
   1. placemark\_and\_polygon. Вложенный пакет, содержащий инструменты для создания и редактирования меток и полигонов на карте.
      1. strategies. Вложенный пакет, наделяющий инструментарий placemark\_and\_polygon паттерном проектирования «Стратегия». Данный паттерн позволяет удобно добавлять различные свойства к меткам и полигонам, с помощью классов наследуемых от StrategyMapObject. Стратегии, применяемые к объектам, описываются во вложенных пакетах placemark и polygon.
      2. Класс PlacemarkFactory. Наследуется от StrategyMapObject обеспечивает привязку стратегий к меткам.
      3. Класс PolygonFactory. Наследуется от StrategyMapObject обеспечивает привязку стратегий к полигонам.
      4. Интерфейс SettingsPlacemarkPolygon. Описывает некоторые общие функции для работы с полигонами и метками.
   2. Класс MapBordersHolder. Удерживает позицию камеры пользователя в обозначенной территории на карте.
   3. Класс MapOperations. Класс операций над картой, исключающий обратные вызовы, слушатели или функции с возвратом данных.
   4. Класс RouteFactory. Класс предоставляющий функционал формирования маршрутов. Данный класс является одним из ключевых для проекта, так как в нем реализовано взаимодействие с API Yandex MapKit.
   5. Класс UserLocation и интерфейс UserLocationUpdateListener. Данная связка классов позволяет отслеживать местоположение пользователя в реальном времени. Вместе с этим, класс UserLocation взаимодействует с классом RouteFactory в случае, формирования динамического маршрута относительно местоположения пользователя.
2. project\_objects. Пакет, содержащий базовые сущности проекта и географические координаты объектов на карте.
   1. base\_entities. Пакет, предоставляющий такие сущности как AcademicBuilding, Building, Dormitory. Сущность Building является основным классом, от которого наследуется AcademicBuilding и Dormitory. Также в пакете есть класс ParcelablePoint, позволяющий объекту точки, содержащей координаты долготы и широты, использовать парсерилизацию.
   2. coordinates. Пакет, содержащий географические координаты в виде объектов в пакете base\_entities и полигонов.
   3. Класс Parceling Tools. Инструментарий, содержащий работу с парсерилизацией.
3. system. Пакет, содержащий один класс ConstansProject. Класс хранит константные значения проекта.
4. user\_map\_interface. Пакет содержит две активности и один адаптер.
   1. Класс MainActivity. Главная активность приложения. В активности осуществляются все подписки на коллбэки, запускаются все процессы, связанные с API, а также происходит первоначальная настройка карты.
   2. Элемент NavigationView. Находится на MainActivity. Содержит интуитивно понятный интерфейс управления функционалом приложения.
   3. Класс RadioAdapter и активность TwoPointsRouteActivity. Активность для формирования маршрута для двух статичных объектов. Маршруты строятся между зданиями студенческого городка.
5. Класс SomeTools. Содержит незначительные функции, упрощающие работу с приложением.

## 2.4 Технологии

Для разработки мобильного приложения используется Android Studio, которая поддерживает различные языки программирования и предоставляет инструменты для создания приложений на платформе Android. Одним из её преимуществ является встроенный эмулятор, позволяющий тестировать создаваемое приложение. Android Studio включает библиотеки, предоставляющие API платформы Android, а также утилиты для создания и сборки приложений, управления виртуальными устройствами и выполнения других задач.

В качестве языка программирования выбран Kotlin, который совместим с Java и не снижает производительность приложения и не увеличивает размер файлов. Одним из ключевых преимуществ Kotlin является уменьшение объема служебного кода, что делает его более удобным для чтения и использования.

Для отображения карт и построения маршрутов используется Yandex MapKit. Библиотека, предоставляемая Яндексом, позволяет интегрировать в приложение функции картографии и навигации, обеспечивая высокую точность и удобство использования.

Для работы с асинхронными задачами и потоками данных используется библиотека Kotlin Coroutines, которая упрощает управление асинхронными операциями и многопоточностью в приложении.

Эти технологии и библиотеки позволяют создать функциональное и удобное приложение, обеспечивающее пользователю возможность легко ориентироваться на территории студенческого городка с помощью интерактивных карт и маршрутов.

# ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Карта студенческого городка

При запуске приложения, пользователю отображается основная активность – карта студенческого городка (рис. 3).

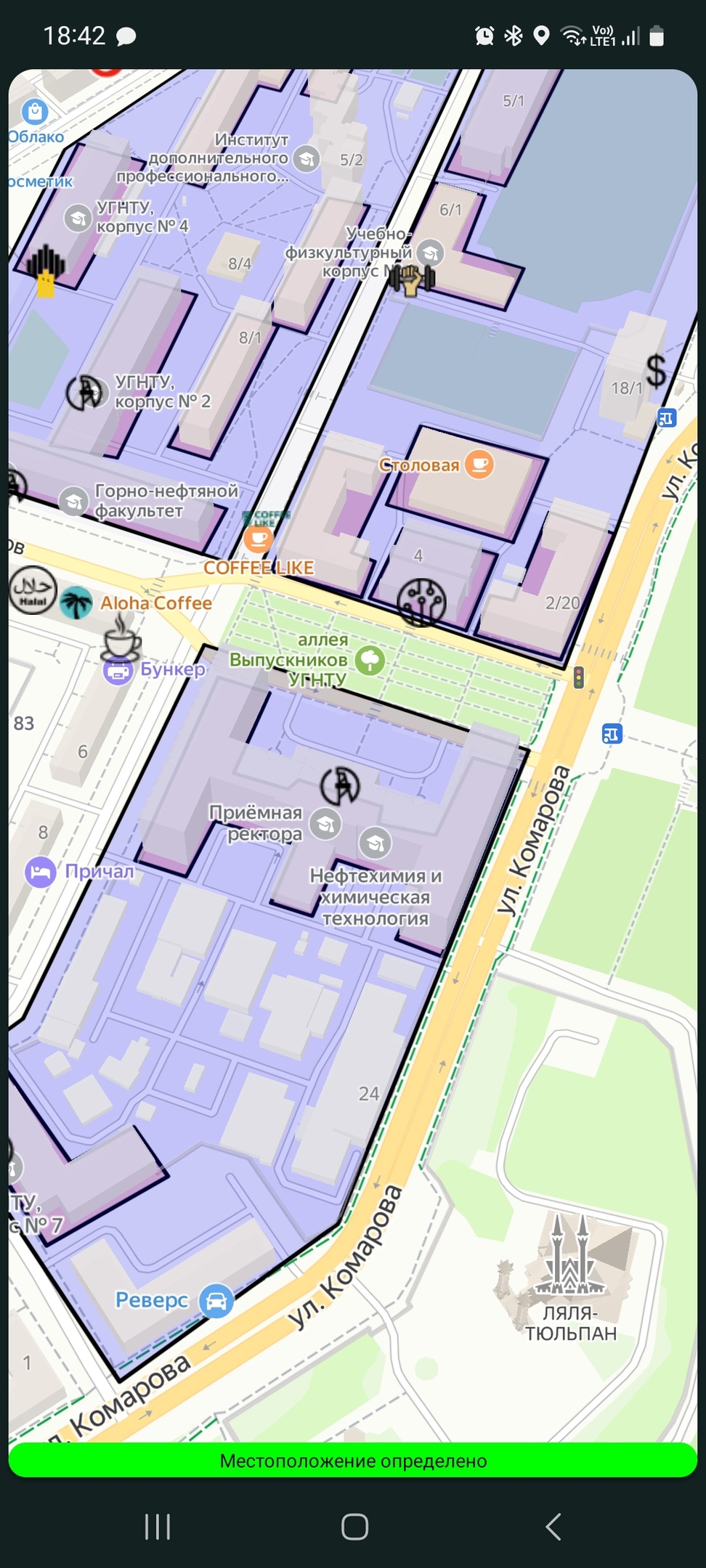


Рисунок 3 – Карта студенческого городка

Снизу основной активности расположен индикатор, уведомляющий о статусе местоположения.

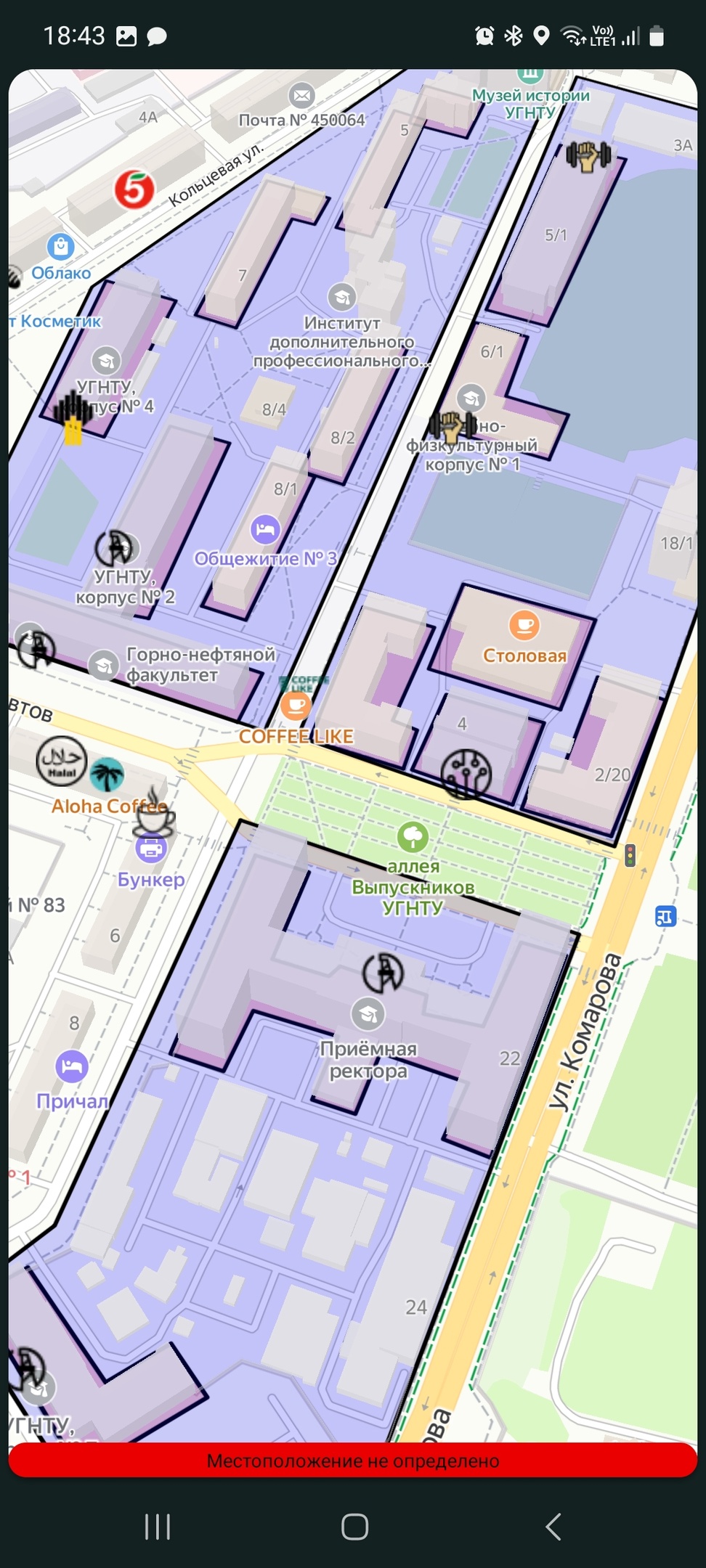


Рисунок 4 – Индикатор, уведомляющий о статусе местоположения

Листинг 1 – Основные операции MainActivity

private fun initializationMaps() {

MapKitFactory.setApiKey(ConstantsProject.API\_KEY\_YANDEX\_MAPS)

MapKitFactory.initialize(this@MainActivity)

TransportFactory.initialize(this@MainActivity)

MapPoints.initialize(this@MainActivity)

}

/\*\*

\* Проверка прав программы, запуск отслеживания местоположения пользователя

\*/

private fun checkAndRequestLocationPermissions() {

if (ContextCompat.checkSelfPermission(this@MainActivity, android.Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED

|| ContextCompat.checkSelfPermission(this@MainActivity, android.Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(

this@MainActivity,

arrayOf(android.Manifest.permission.ACCESS\_FINE\_LOCATION, android.Manifest.permission.ACCESS\_COARSE\_LOCATION),

1

)

} else {

//Разрешения уже предоставлены, можно начинать отслеживание местоположения

userLocation.setLocationListener(this@MainActivity)

userLocation.initUserLocation(ImageProvider.fromResource(

bindingMainActivity.root.context,

R.drawable.walkperson\_placemark

))

}

}

/\*\*

\* Проверка прав программы, запуск отслеживания местоположения пользователя

\*/

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults)

if (requestCode == 1) {

if ((grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION\_GRANTED)) {

//Разрешения предоставлены, начинаем отслеживание местоположения

userLocation.setLocationListener(this@MainActivity)

userLocation.initUserLocation(ImageProvider.fromResource(

bindingMainActivity.root.context,

R.drawable.walkperson\_placemark

))

} else {

//Разрешения отклонены, показываем пользователю объяснение необходимости разрешений

}

}

}

/\*\*

\* Уведомеление о статусе местоположения

\*/

override fun onLocationUpdated(location: Location) = with(bindingMainActivity) {

runOnUiThread {

cardViewStatus.setCardBackgroundColor(Color.GREEN)

textViewStatus.text = getString(R.string.location\_is\_determined)

}

}

/\*\*

\* Комплект лаунчеров текущего Activity

\*/

private var twoPointsActivityLauncher: ActivityResultLauncher<Intent>? = null

private fun launchersPack() {

//возвращение ответа из 2PointsActivity activity

twoPointsActivityLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {

if(it.resultCode == RESULT\_OK) {

// Получение данных из интента

val data: Intent? = it.data

val value1String = data?.getStringExtra(INTENT\_KEY1)

val value2String = data?.getStringExtra(INTENT\_KEY2)

if (value1String != null && value2String != null) {

val value1ByteArray = Base64.decode(value1String, Base64.DEFAULT)

val value2ByteArray = Base64.decode(value2String, Base64.DEFAULT)

val value1 = value1ByteArray.toParcelable(Building.CREATOR)

val value2 = value2ByteArray.toParcelable(Building.CREATOR)

if (value1 != null && value2 != null) {

routeFactory.requestRoute2Points(value1.coordinatesExit.toMapKitPoint(), value2.coordinatesExit.toMapKitPoint()) {

}

}

}

}

}

}

Для удобства, посредством наложения полигонов, на карте выделены зоны, относящиеся к студенческому городку. Здания студенческого городка помечены специальными иконками (рис. 5).

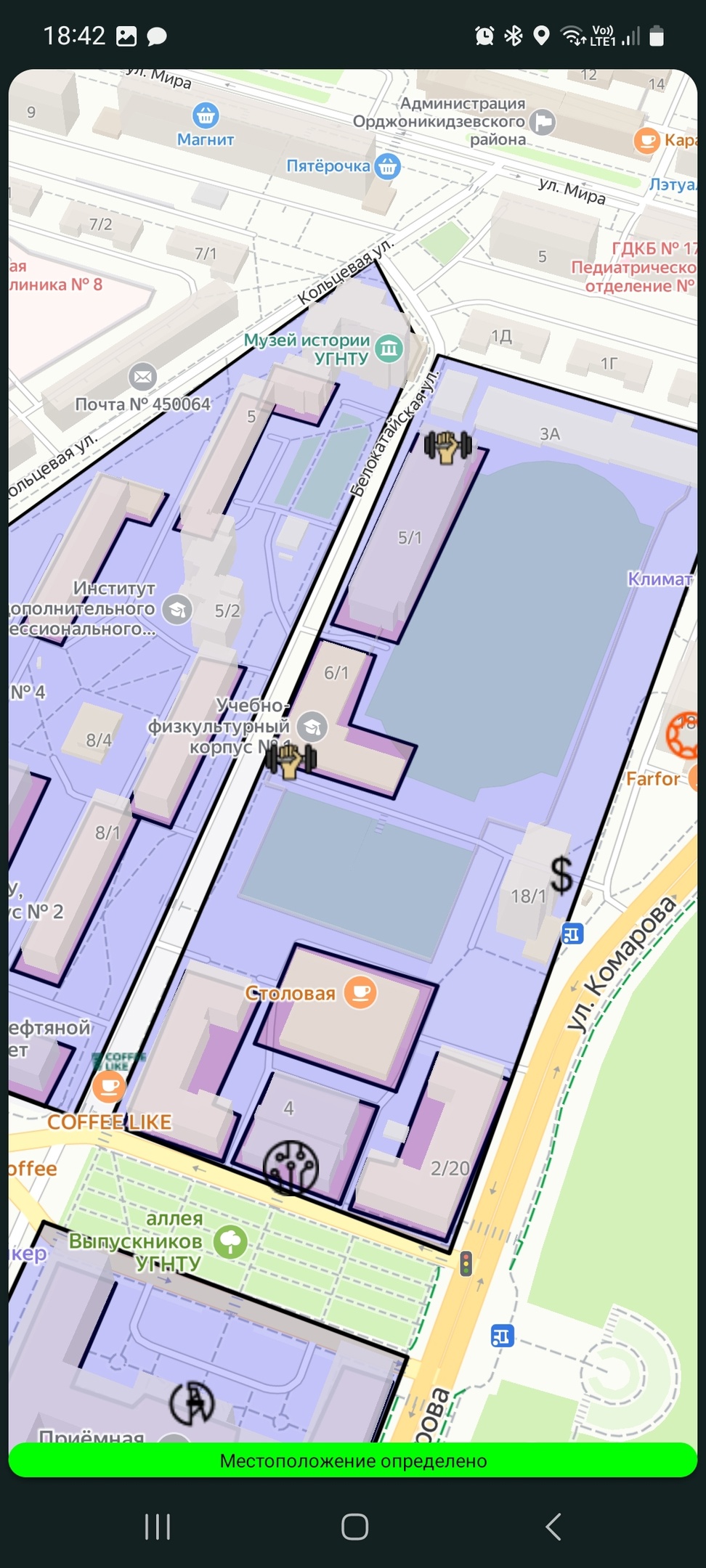
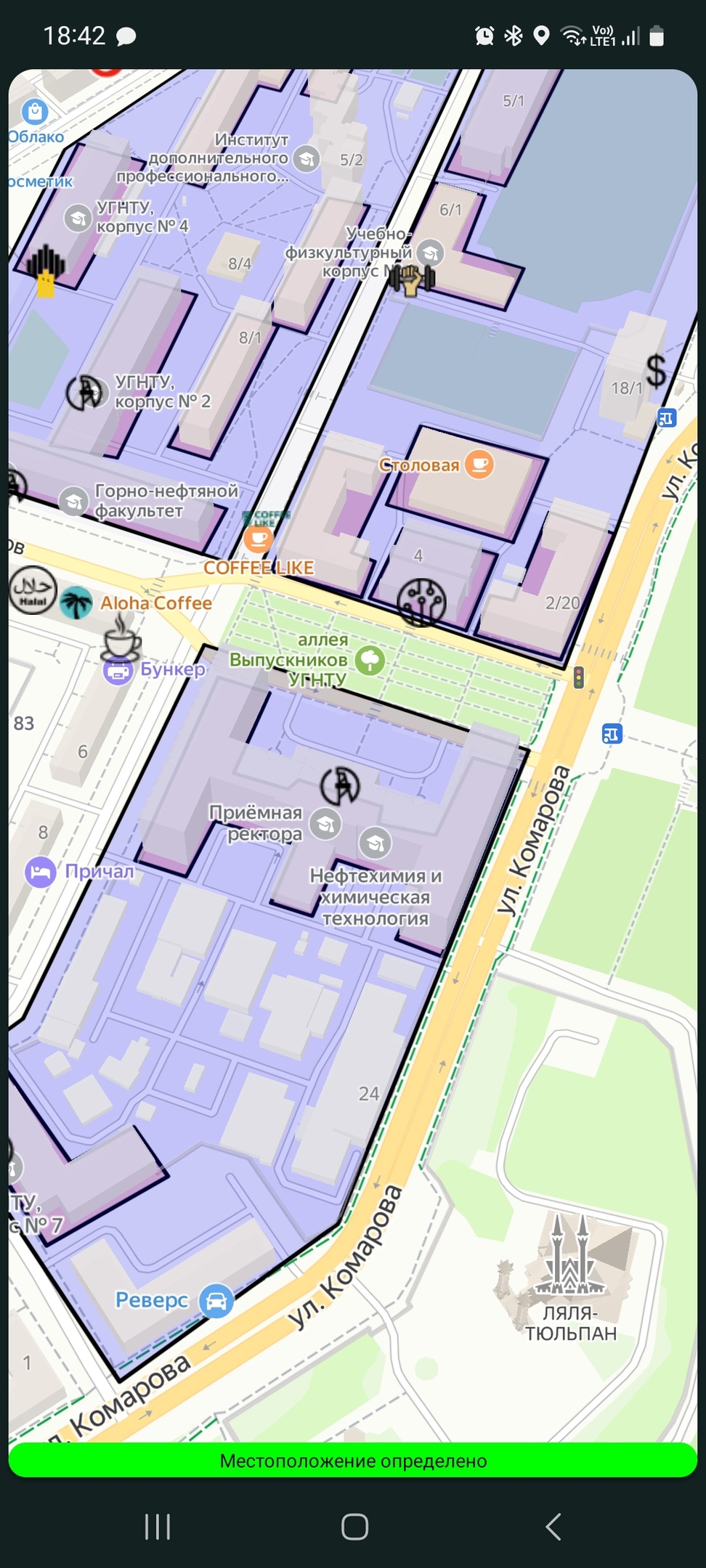
 

Рисунок 5 – Метки и полигоны на карте

Листинг 2 – MapOperations

class MapOprations(private val binding: ActivityMainBinding) {

private val placemarkFactory: PlacemarkFactory = PlacemarkFactory(binding.mapViewMain)

private val polygonFactory: PolygonFactory = PolygonFactory(binding.mapViewMain)

private var cameraPositionMain = CameraPosition(

MapPoints.CENTER\_USPTU\_CITY\_COORD,

/\* zoom = \*/ 17.0f,

/\* azimuth = \*/ 150.0f,

/\* tilt = \*/ 30.0f

)

fun getCameraPosition(): CameraPosition = with(binding) {

return mapViewMain.map.cameraPosition

}

fun startPointMaps() = with(binding) {

mapViewMain.map.isZoomGesturesEnabled = false

mapViewMain.map.move(

cameraPositionMain,

Animation(com.yandex.mapkit.Animation.Type.SMOOTH, 1F),

null

)

}

fun customPlacemarksOfMap() {

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[0], R.drawable.ugntu2\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[1], R.drawable.ugntu2\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[2], R.drawable.ugntu2\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[3], R.drawable.rosneft\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[4], R.drawable.ugntu2\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[5], R.drawable.ugntu2\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[6], R.drawable.techpark\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[7], R.drawable.sport\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CORPUSES[8], R.drawable.sport\_placemark)

//кафе

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[0], R.drawable.halal\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[1], R.drawable.halal\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[2], R.drawable.food\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[3], R.drawable.shaurma\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[4], R.drawable.dodo\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[5], R.drawable.kfc\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[6], R.drawable.fujiyama\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[7], R.drawable.bakery\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[8], R.drawable.farfor\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[9], R.drawable.coffee\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[10], R.drawable.aloha\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[11], R.drawable.coffeelike\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CAFE[12], R.drawable.mvk\_placemark)

//чилл зоны

placemarkFactory.addPlacemarkOnMap("", MapPoints.CHILL[0], R.drawable.library\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.CHILL[1], R.drawable.colizeum\_placemark)

//продуктовые магазины

placemarkFactory.addPlacemarkOnMap("", MapPoints.PRODUCTS[0], R.drawable.dollar\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.PRODUCTS[1], R.drawable.kb\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.PRODUCTS[2], R.drawable.monetka\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.PRODUCTS[3], R.drawable.pyatyorochka\_placemark)

placemarkFactory.addPlacemarkOnMap("", MapPoints.PRODUCTS[4], R.drawable.dollar\_placemark)

}

fun polygonsOfMap() = with(binding) {

PolygonsMapPoints.apply {

polygonFactory.addPolygonOnMap(firstCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(elevenCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(ufkOneCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(ufkTwoCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(thirdCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(secondCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(fourthCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(sevenCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(eighthCorpus, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(firstDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(secondDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(thirdDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(fourthDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(fiveDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(sixDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(tenDormitory, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(dinningRoom, ContextCompat.getColor(root.context, R.color.light\_red))

polygonFactory.addPolygonOnMap(firstCorpusArea, ContextCompat.getColor(root.context, R.color.light\_blue))

polygonFactory.addPolygonOnMap(secondCorpusArea, ContextCompat.getColor(root.context, R.color.light\_blue))

polygonFactory.addPolygonOnMap(thirdCorpusArea, ContextCompat.getColor(root.context, R.color.light\_blue))

polygonFactory.addPolygonOnMap(fourthCorpusArea, ContextCompat.getColor(root.context, R.color.light\_blue))

}

}

}

Листинг 3 – PolygonFactory и Placemark Factory

class PolygonFactory(private val mapView: MapView) : SettingsPlacemarkPolygon {  
 private val mapObjects: MapObjectCollection = mapView.map.mapObjects.addCollection()  
  
 fun addPolygonOnMap(points: List<Point> = listOf(Point(50.0, 50.0)), color: Int = Color.RED) : PolygonManager {  
 // Создание полигона  
 val polygon = Polygon(LinearRing(points), listOf())  
  
 // Добавление полигона на карту и настройка его внешнего вида  
 val polygonMapObject = mapObjects.addPolygon(polygon)  
 polygonMapObject.fillColor = color // Пример: полупрозрачный красный цвет  
 polygonMapObject.strokeColor = Color.BLACK // Цвет границы  
 polygonMapObject.strokeWidth = 2.0f // Ширина границы  
  
 return PolygonManager(polygonMapObject, mapObjects)  
 }  
  
 override fun deleteMapObjects() { // удаление всех полигонов  
 mapObjects.clear()  
 }  
}

class PlacemarkFactory(private val mapView: MapView): SettingsPlacemarkPolygon {  
 private val mapObjects = mapView.map.mapObjects.addCollection()  
  
 fun addPlacemarkOnMap(title: String = "", point: Point = MapPoints.CENTER\_USPTU\_CITY\_COORD, icon: Int = 0) : PlacemarkManager {  
 val context = mapView.context  
 val placemark = mapObjects.addPlacemark(point)  
 placemark.setIcon(com.yandex.runtime.image.ImageProvider.fromResource(context, icon)) // Установка иконки  
 placemark.setText(title)  
 return PlacemarkManager(placemark, mapObjects)  
 }  
  
 override fun deleteMapObjects() { //удаление всех пметок  
 mapObjects.clear()  
 }  
}

## 3.1.1 Навигационное меню

Навигационное меню демонстрирует ключевые возможности приложения, предоставляя пользователю выбор метода прокладки маршрута от начальной точки до конечного пункта. В меню выделены две категории: маршруты и здания.

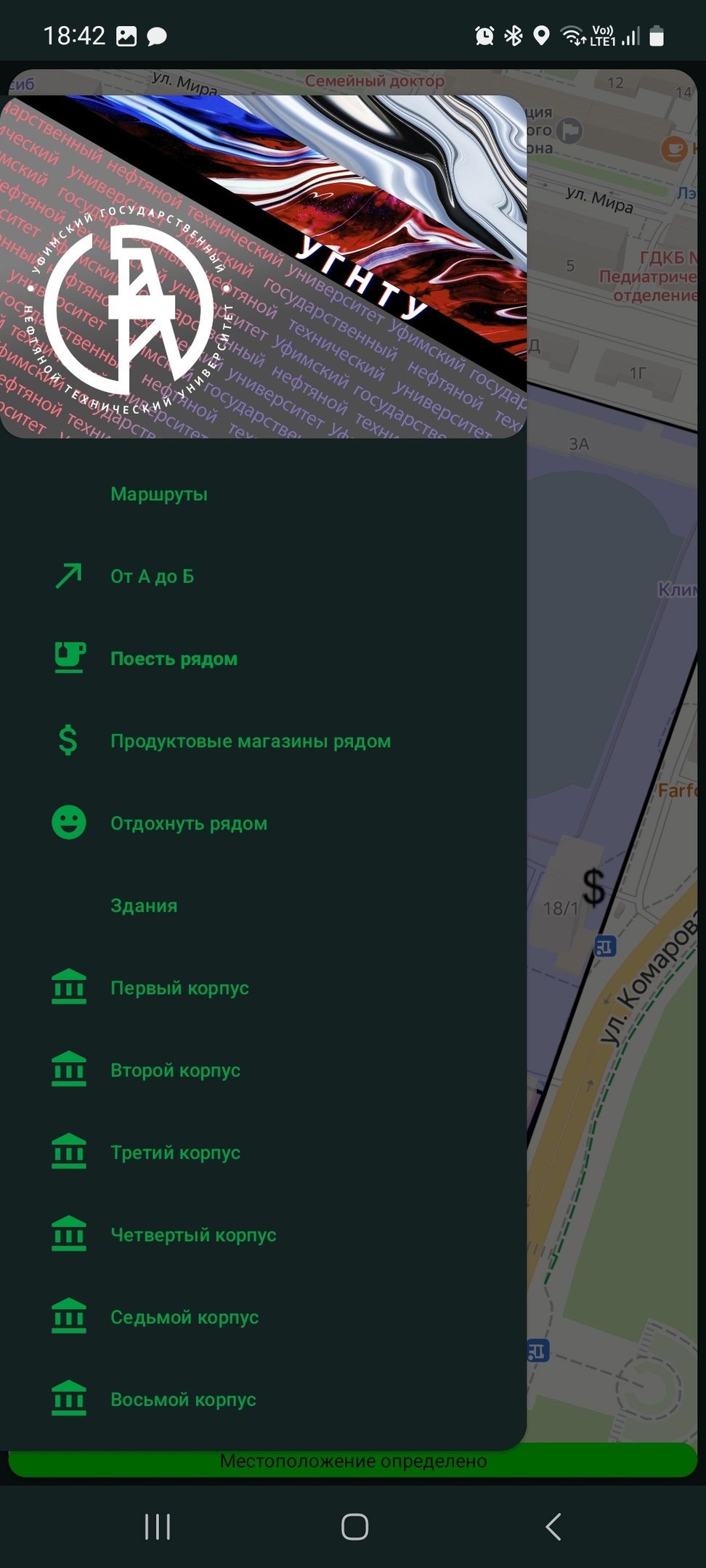


Рисунок 6 – Навигационное меню

Листинг 4 – NavigationView

/\*\*  
 \* Инициализация NaviagationView  
 \*/  
 private fun initializationNavMenu() = with(bindingMainActivity) {  
 navViewMain.setNavigationItemSelectedListener {  
 when(it.itemId) {  
 //ГРУППА - МАРШРУТЫ  
 R.id.itemMakeRoute -> {  
 routeFactory.removeAllRoutes()  
 userLocation.routingEnabled = false  
  
 val intent = Intent(this@MainActivity, TwoPointsRouteActivity::class.java)  
 twoPointsActivityLauncher?.launch(intent)  
 }  
 R.id.itemMakeLessonRoute -> {  
 //ПОЛУЧИТЬ ИНФОРМАЦИЮ О ПАРАХ  
 routeFactory.removeAllRoutes()  
 //userLocation.routingEnabled = true  
// userLocation.setEndPoint(MapPoints.academicBuildings[0].coordinatesExit.toMapKitPoint())  
//  
// val wp = WebParsing("БПО-21-01", this@MainActivity);  
// lifecycleScope.launch { wp.parseSchedule() }  
// var lessons = wp.getSchedule()  
// lessons = wp.getSchedule()  
 }  
 R.id.itemMakeRouteFood -> {  
 val building = userLocation.getNearestBuilding(MapPoints.universityCafe)!!  
 functionForBuildings(building.coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemMakeRouteProducts -> {  
 val building = userLocation.getNearestBuilding(MapPoints.universityProducts)!!  
 functionForBuildings(building.coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemMakeRouteRelax -> {  
 val building = userLocation.getNearestBuilding(MapPoints.universityChill)!!  
 functionForBuildings(building.coordinatesExit.toMapKitPoint())  
 }  
  
 //ГРУППА - ЗДАНИЯ  
 R.id.itemFirstCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[0].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemSecondCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[1].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemThirdCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[2].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemFourthCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[3].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemSevenCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[4].coordinatesExit.toMapKitPoint())  
 }  
 R.id.ItemEightCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[5].coordinatesExit.toMapKitPoint())  
 }  
 R.id.ItemElevenCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[6].coordinatesExit.toMapKitPoint())  
 }  
 R.id.ItemUfkOneCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[7].coordinatesExit.toMapKitPoint())  
 }  
 R.id.ItemUfkTwoCorpus -> {  
 functionForBuildings(MapPoints.academicBuildings[8].coordinatesExit.toMapKitPoint())  
 }  
  
 //ГРУППА - ОБЩЕЖИТИЯ  
 R.id.itemFirstDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[0].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemSecondDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[1].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemThirdDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[2].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemFourthDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[3].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemFiveDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[4].coordinatesExit.toMapKitPoint())  
 }  
 R.id.itemSixDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[5].coordinatesExit.toMapKitPoint())  
 }  
 R.id.ItemTenDormitory -> {  
 functionForBuildings(MapPoints.universityDormitories[6].coordinatesExit.toMapKitPoint())  
 }  
 }  
  
 drawerLayoutMain.closeDrawers()  
 true  
 }  
  
 // Создаем Handler для главного потока  
 Handler(Looper.getMainLooper()).postDelayed({  
 // Проверяем, не уничтожена ли уже активность  
 if (!isFinishing) {  
 drawerLayoutMain.openDrawer(GravityCompat.START)  
 }  
 }, 1500) // Задержка в 2 секунды  
 }

### 3.1.1.1 Категория «Маршруты»

Категория "Маршруты" включает в себя возможность проложить маршрут от "А до Б", то есть от одного определенного объекта к другому, а также построить маршрут от текущего местоположения пользователя до продуктового магазина, кафе или зоны отдыха.

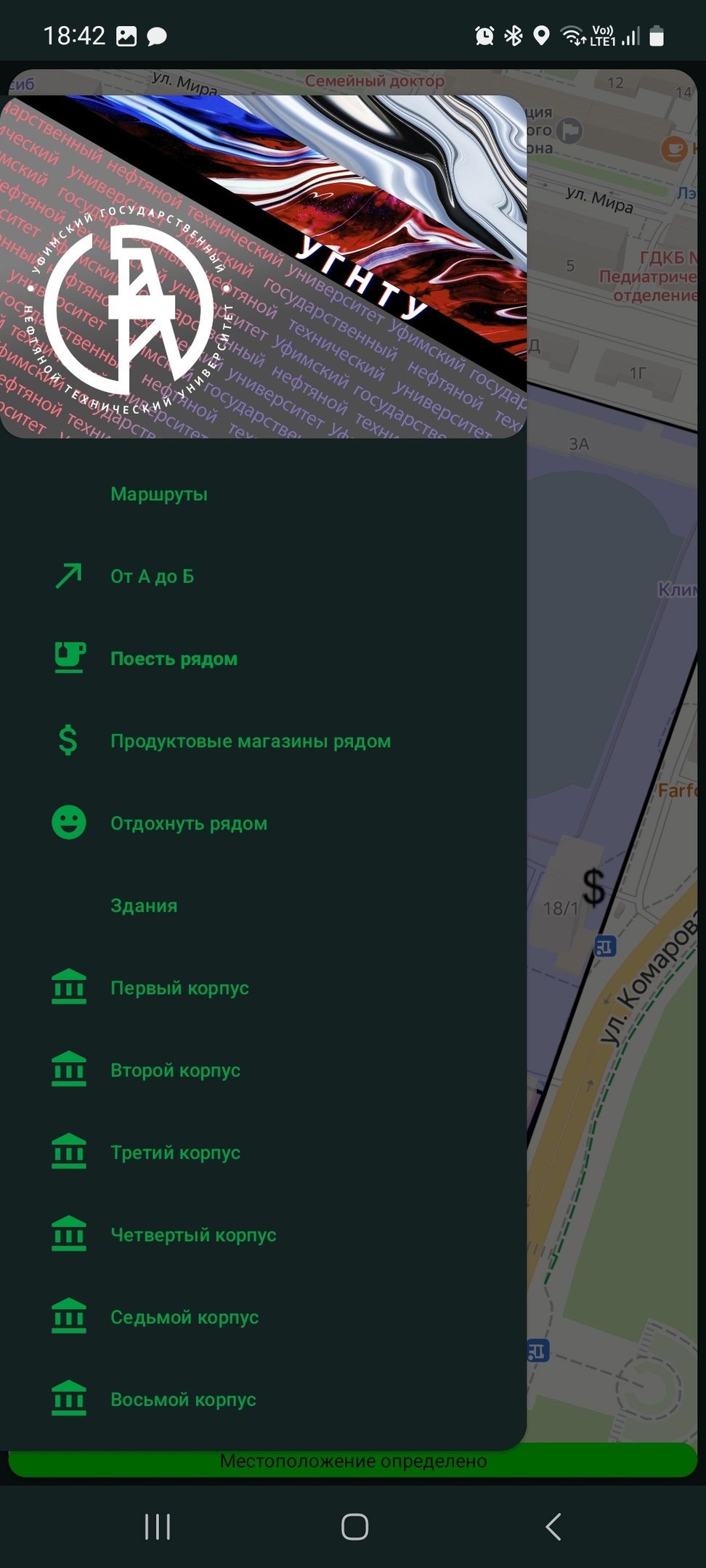


Рисунок 7 – Категория «Маршруты»

Листинг 5 – RouteFactory

/\*\*  
 \* Класс описывающий формирование маршрутов на карте  
 \*/  
class RouteFactory(private val mapView: MapView) {  
 private var routesCollection: MapObjectCollection? = mapView.map.mapObjects.addCollection()  
 private var routeOfMap: PolylineMapObject? = null  
 private var pedestrianSession: Session? = null  
  
 /\*\*  
 \* Метод построения маршрута из точки A в точку B  
 \*/  
 public fun requestRoute2Points(  
 startPoint: Point,  
 endPoint: Point,  
 onError: (com.yandex.runtime.Error) -> Unit  
 ): Session {  
 val pedestrianRouter = TransportFactory.getInstance().createPedestrianRouter()  
  
 //Создание запроса  
 val requestPoints = arrayListOf(  
 RequestPoint(startPoint, RequestPointType.WAYPOINT, null, null),  
 RequestPoint(endPoint, RequestPointType.WAYPOINT, null, null)  
 )  
  
 //Удаление предыдущего маршрута, если он существует  
 //removeCurrentRoute()  
 removeAllRoutes()  
  
 pedestrianSession = pedestrianRouter.requestRoutes(  
 requestPoints,  
 TimeOptions(), // TimeOptions без nullable параметров  
 object : Session.RouteListener {  
 override fun onMasstransitRoutes(routes: MutableList<Route>) {  
 if (routes.isNotEmpty()) {  
 // Создаем новый маршрут на карте  
 //routeOfMap = binding.mapViewMain.map.mapObjects.addPolyline(routes.first().geometry)  
 routeOfMap = routesCollection?.addPolyline(routes.first().geometry)  
 routeOfMap!!.setStrokeColor(Color.RED)  
 }  
 }  
  
 override fun onMasstransitRoutesError(error: com.yandex.runtime.Error) {  
 onError(error)  
 }  
 }  
 )  
  
 return pedestrianSession as Session  
 }  
  
 /\*\*  
 \* Удаление всех маршрутов в колекции  
 \*/  
 public fun removeAllRoutes() {  
 routesCollection?.clear() // Удаляем все маршруты из коллекции  
 }  
  
 /\*\*  
 \* Удаление текущего маршрута  
 \*/  
 private fun removeCurrentRoute() {  
 routeOfMap?.let {  
 mapView.map.mapObjects.remove(it)  
 routeOfMap = null // Обнуляем ссылку, чтобы избежать повторного использования  
 }  
 }  
  
 /\*\*  
 \* Закрытие сессии построения маршрутов  
 \*/  
 public fun cancelRouteSession() {  
 pedestrianSession?.cancel()  
 }  
}

При нажатии на кнопку «От А до Б» открывается новое окно с двумя списками. Первый список содержит объекты студенческого городка, из которых пользователь может выбрать начальную точку маршрута. Второй список аналогичен первому, но предназначен для выбора конечной точки маршрута (рис. 8).

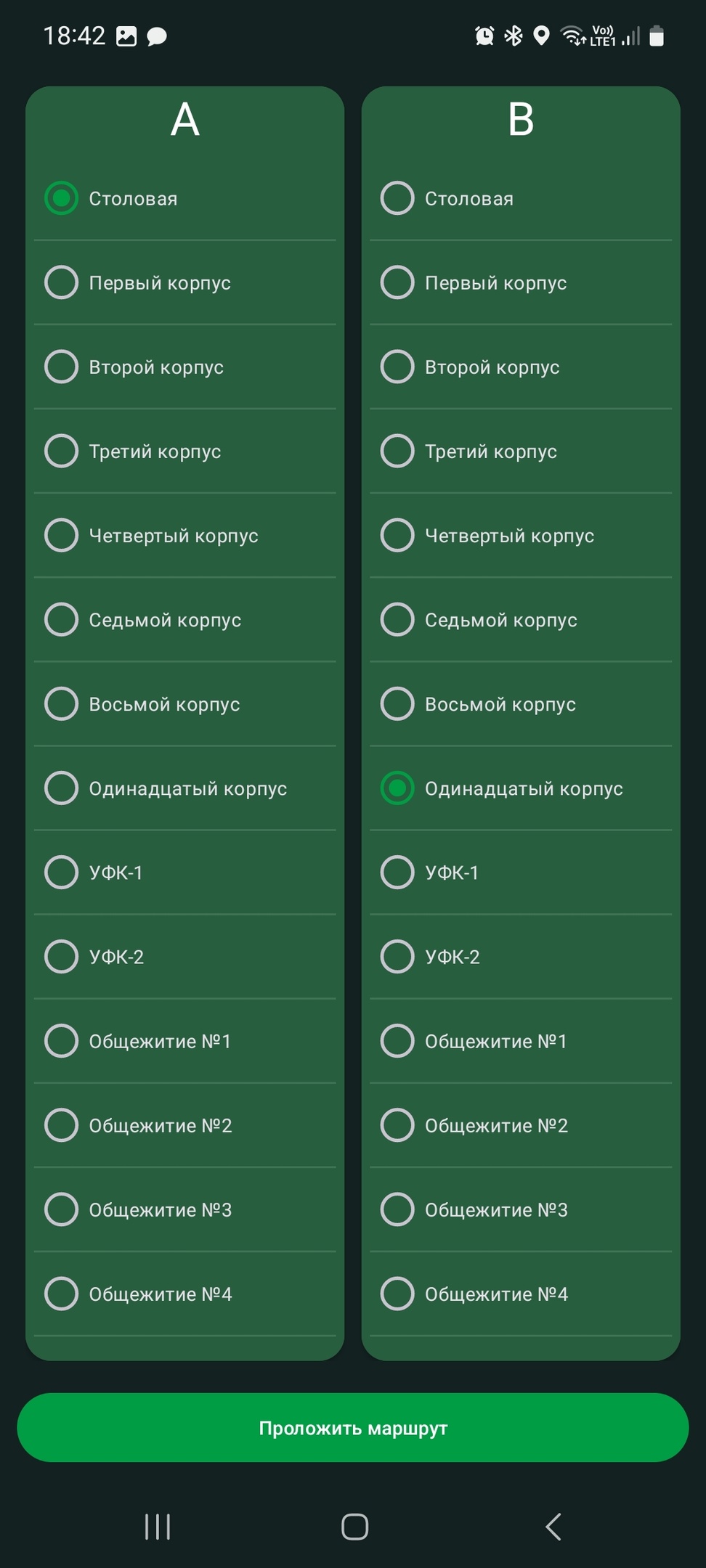
 

Рисунок 8 – Активность «От А до Б»

Листинг 6 – RadioAdapter

class RadioAdapter(  
 private val context: Context,  
 private val items: List<String>  
) : ArrayAdapter<String>(context, 0, items) {  
 private lateinit var binding: ListItemRadioButtonBinding  
  
 // Внешне управляемая переменная для отслеживания выбранной позиции  
 var selectedPosition: Int = -1  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {  
 if (convertView == null) {  
 binding = ListItemRadioButtonBinding.inflate(LayoutInflater.from(context), parent, false)  
 } else {  
 binding = ListItemRadioButtonBinding.bind(convertView)  
 }  
  
 // Устанавливаем текст радиокнопки  
 binding.radioButtonOption.text = items[position]  
  
 // Отмечаем радиокнопку, если текущая позиция соответствует выбранной  
 binding.radioButtonOption.isChecked = position == selectedPosition  
  
 return binding.root  
 }  
}

Листинг 7 – TwoPointrRouteActivity

class TwoPointsRouteActivity : AppCompatActivity() {  
 private lateinit var bindingActivityTwoPointsRoute: ActivityTwoPointsRouteBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 bindingActivityTwoPointsRoute = ActivityTwoPointsRouteBinding.inflate(layoutInflater)  
 setContentView(bindingActivityTwoPointsRoute.root)  
  
 init()  
 }  
  
 private lateinit var adapterLeft: RadioAdapter  
 private lateinit var adapterRight: RadioAdapter  
 private var selectedItemLeft = -1  
 private var selectedItemRight = -1  
 private fun init() = with(bindingActivityTwoPointsRoute) {  
 // Получаем массив строк из ресурсов  
 //val options = resources.getStringArray(R.array.buildings)  
 val options: ArrayList<String> = arrayListOf()  
 combinedBuildingList.forEach { options.add(it.name.toString()) }  
  
 //Инициализируем адаптер  
 adapterLeft = RadioAdapter(this@TwoPointsRouteActivity, options.toList())  
 adapterRight = RadioAdapter(this@TwoPointsRouteActivity, options.toList())  
  
 //Находим ListView и устанавливаем для него адаптер  
 listViewLeft.adapter = adapterLeft  
 listViewRight.adapter = adapterRight  
  
 // Устанавливаем слушателя для обработки выбора элементов - нихуя не работает (посмотри с помощью точки останова)  
 listViewLeft.setOnItemClickListener { parent, view, position, id ->  
 adapterLeft.selectedPosition = position  
 adapterLeft.notifyDataSetChanged()  
  
 // Получаем выбранный элемент  
 selectedItemLeft = position  
 if(selectedItemLeft != -1 && selectedItemRight != -1) buttonAccept.isEnabled = true  
 }  
 listViewRight.setOnItemClickListener { parent, view, position, id ->  
 adapterRight.selectedPosition = position  
 adapterRight.notifyDataSetChanged()  
  
 // Получаем выбранный элемент  
 selectedItemRight = position  
 if(selectedItemLeft != -1 && selectedItemRight != -1) buttonAccept.isEnabled = true  
 }  
 }  
  
 fun buttonAcceptOnClick(view: View) {  
 val value1ByteArray = combinedBuildingList[selectedItemLeft].toByteArray()  
 val value2ByteArray = combinedBuildingList[selectedItemRight].toByteArray()  
 val value1String = Base64.encodeToString(value1ByteArray, Base64.DEFAULT)  
 val value2String = Base64.encodeToString(value2ByteArray, Base64.DEFAULT)  
  
 val returnIntent = Intent().apply {  
 putExtra(INTENT\_KEY1, value1String)  
 putExtra(INTENT\_KEY2, value2String)  
 }  
  
 setResult(Activity.RESULT\_OK, returnIntent)  
 finish()  
 }  
}

### Категория «Здания»

В навигационном меню, в категории "Здания", пользователь может выбрать объект студенческого городка, к которому нужно проложить маршрут от его текущего местоположения.

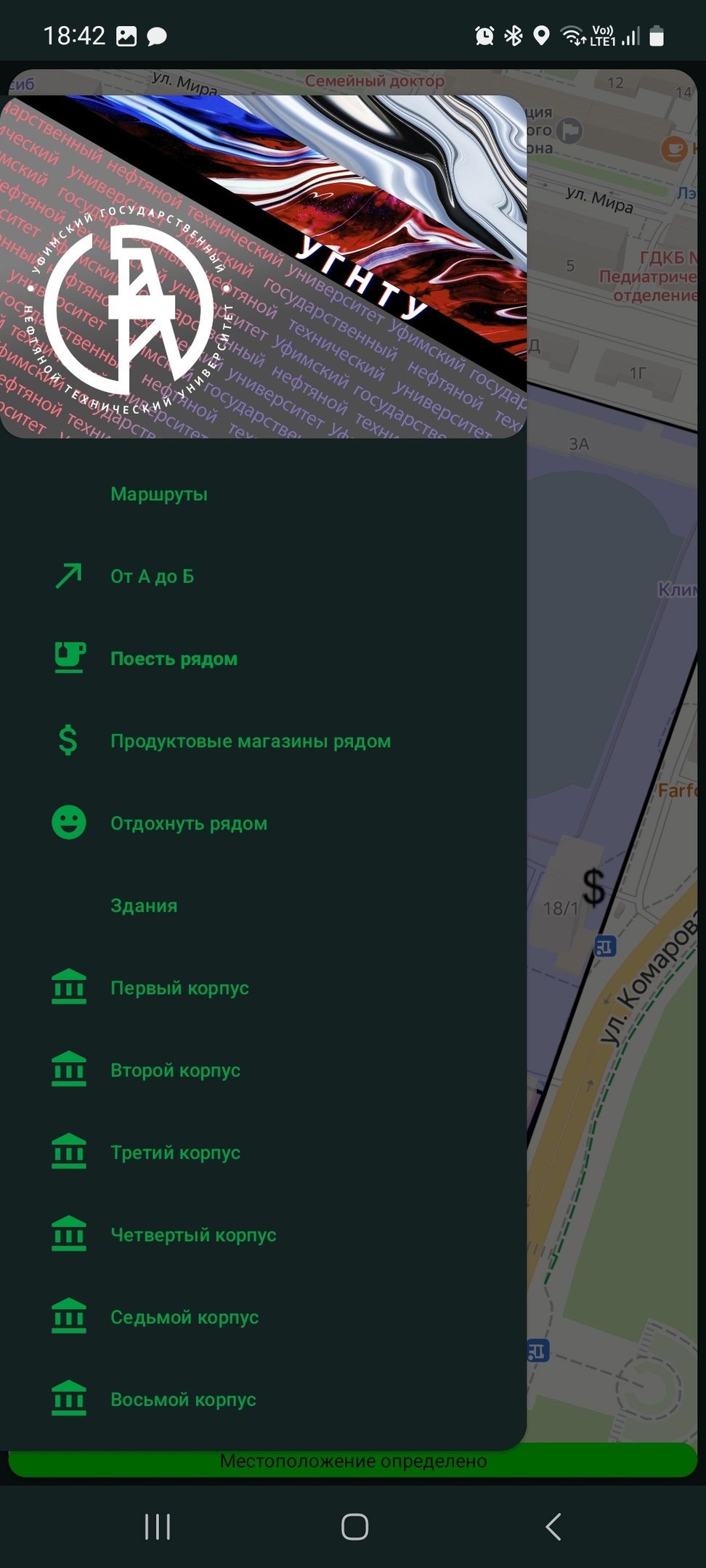


Рисунок 9 – Категория «Здания»

### Границы студенческого городка

Если местоположение пользователя было определено за пределами студенческого городка, то приложение предложит воспользоваться другими навигационными сервисами, для прокладки маршрута до студенческого городка.

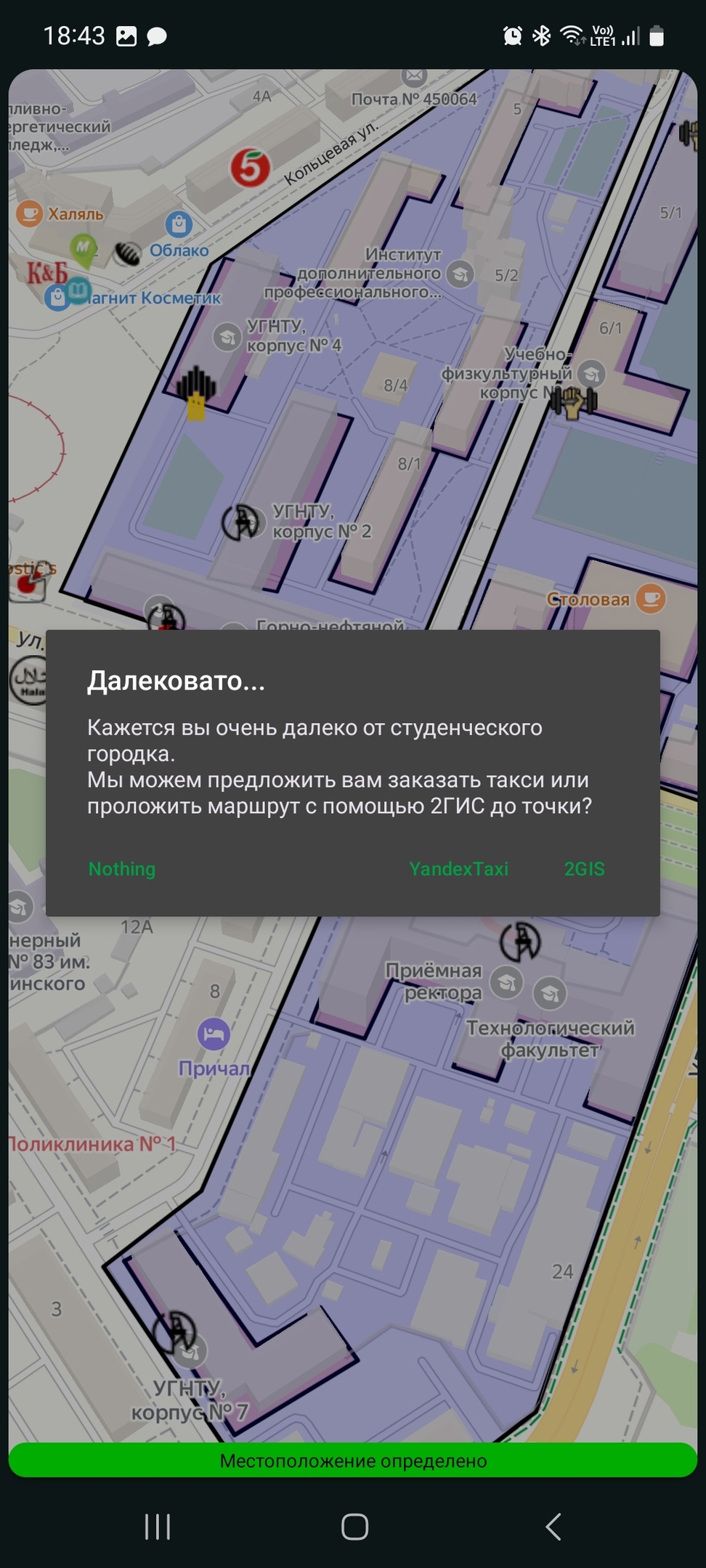


Рисунок 10 – Уведомление о местоположении за пределами студенческого городка

Листинг 8 – chekDistance

/\*\*

\* Проверка больше ли расстояние, чем distanceTo от пользователя до точки

\*/

public fun checkDistance(

target: Point = Point(0.0, 0.0),

distanceTo: Float = 1000F

): Boolean {

// Создаём объект Location для пользоваетля и выбранной точки

val userAndroidLocation = AndroidLocation("").apply {

latitude = userLocation?.position?.latitude ?: 0.0

longitude = userLocation?.position?.longitude ?: 0.0

}

val targetAndroidLocation = AndroidLocation("").apply {

latitude = target.latitude

longitude = target.longitude

}

// Расстояние между местоположением пользователя и выбранной точкой в метрах

val distance = userAndroidLocation.distanceTo(targetAndroidLocation)

// Проверяем, больше ли расстояние чем distanceTo

return if (distance > distanceTo) {

true

} else {

false

}

}

# ЗАКЛЮЧЕНИЕ

В рамках курсовой работы было разработано мобильное приложение «Навигатор по студенческому городку», содержащее функционал для эффективной навигации по территории университетского кампуса. Ссылка на репозиторий с готовым проектом расположена в Приложении А.

Были проанализированы технологии создания мобильных приложений, выбрана операционная система Android для работы программного продукта и среда разработки Android Studio.

В ходе выполнения курсовой работы были решены следующие задачи:

* проведен анализ предметной области;
* изучены основные концепции разработки мобильных приложений с использованием языка программирования Kotlin;
* разработаны функциональная и логическая модели;
* реализовано мобильное приложение, упрощающее навигацию по студенческому городку.

Для реализации навигационных функций приложения использовалась библиотека Yandex MapKit, обеспечивающая высокую точность и удобство использования карт и маршрутов. Библиотека Yandex MapKit интегрирована с основным функционалом приложения, предоставляя пользователям возможность прокладывать маршруты между различными объектами кампуса.

Асинхронные задачи и потоки данных управляются с использованием библиотеки Kotlin Coroutines, что упрощает управление асинхронными операциями и многопоточностью в приложении.

Так, в процессе работы удалось достичь поставленной цели – создать мобильное приложение под платформу Android для навигации по студенческому городку, используя Yandex MapKit.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация [Yandex MapKit](https://yandex.ru/dev/mapkit/doc/ru/android/generated/getting_started) [Электронный ресурс] // Карты и Навигация – Подключение и использование на Android. URL: <https://yandex.ru/dev/mapkit/doc/ru/android/generated/getting_started> (дата обращения: 22.02.2024).
2. Парамонов, И. В. Разработка мобильных приложений для платформы Android: учебное пособие / И. В. Парамонов; Яросл. гос. ун-т им. П. Г. Демидова. — Ярославль: ЯрГУ, 2013. — 88 с.
3. Разработка мобильных приложений: электронное учеб.-метод. пособие для студентов специальностей 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)», 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем» / сост. Н. В. Пацей. – Минск: БГТУ, 2020. – 265 с.
4. Корутины в Kotlin [Электронный ресурс] // Руководство по асинхронному программированию. URL: <https://kotlinlang.org/docs/coroutines-guide.html> (дата обращения: 15.03.2024).
5. RecyclerView [Электронный ресурс]. –– 2020. –– URL: <https://developer.android.com/guide/topics/ui/layout/recyclerview> (дата обращения: 02.04.2024).
6. Документация Android Studio [Электронный ресурс] // Официальный сайт Android Developers. URL: <https://developer.android.com/studio> (дата обращения: 14.02.2024)

# ПРИЛОЖЕНИЕ А

# Ссылка на репозиторий GitHub

<https://github.com/Naillin/USPTU_Map.git>