Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Уральский федеральный университет**

**имени первого Президента России Б.Н.Ельцина»**   
  
Математико-механический факультет

Кафедра высокопроизводительных компьютерных технологий   
  
  
 РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ

НА БАЗЕ МОБИЛЬНЫХ ИНТЕРФЕЙСОВ

|  |  |  |
| --- | --- | --- |
| «Допустить к защите» Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_2011 г. |  | Дипломная работа  студентки V курса Баевой А.И.  Научный руководитель  кандидат физ.-мат. наук,  доцент кафедры информатики  и процессов управления  Авербух В.Л. |

Екатеринбург  
2011

**РЕФЕРАТ**

Баева А.И. РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ НА БАЗЕ МОБИЛЬНЫХ ИНТЕРФЕЙСОВ, дипломная работа: стр.107, рис. 20, прил. 1

Ключевые слова: ИНФОРМАЦИОННАЯ СИСТЕМА, ЭЛЕКТРОННАЯ КАРТА, ИНФОРМАЦИОННАЯ ПОДДЕРЖКА ПОЛЬЗОВАТЕЛЯ, ANDROID, ГЕОКОДИРОВАНИЕ, ИНФОРМАЦИОННАЯ ВИЗУАЛИЗАЦИЯ.

Объект исследования – информационные системы для мобильных устройств.

Цель работы – разработка туристической карты для мобильных устройств на платформе Android.

В процессе работы были проведены исследования в области городских информационных систем на базе мобильных интерфейсов, изучены подходы к реализации подобных систем для платформы Android.

Результатом работы стала прототипная реализация приложения (Туристическая карта) для мобильных устройств на платформе Android.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc294788263)

[1. ГОРОДСКИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И МОБИЛЬНЫЕ ИНТЕРФЕЙСЫ 8](#_Toc294788264)

[1.1 Введение в ГИС 8](#_Toc294788265)

[1.2 Организация данных в ГИС 11](#_Toc294788266)

[1.3 ГИС среди информационных технологий 17](#_Toc294788267)

[1.4 Что ГИС могут сделать для туризма 24](#_Toc294788268)

[1.5 Предметно ориентированные ГИС 25](#_Toc294788269)

[1.6 Основные концепции информационной визуализации, используемые в ГИС 25](#_Toc294788270)

[1.7 ГИС и мобильные интерфейсы 27](#_Toc294788271)

[2. ГЕОКОДИРОВАНИЕ В ИС 28](#_Toc294788272)

[2.1 Понятие геокодирования. 28](#_Toc294788273)

[2.2 Назначение геокодирования 28](#_Toc294788274)

[2.3 Методы геокодирования 29](#_Toc294788275)

[3. ОСНОВНЫЕ ТРЕБОВАНИЯ К ТУРИСТИЧЕСКОЙ КАРТЕ 33](#_Toc294788276)

[4. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ 34](#_Toc294788277)

[4.1 Операционная система Google Android 34](#_Toc294788278)

[5. ОПИСАНИЕ ПРОЕКТА ПРИЛОЖЕНИЯ 44](#_Toc294788279)

[5.1 Структура программы. 44](#_Toc294788280)

[5.2 Описание компонентов программы 45](#_Toc294788281)

[5.3 Программная реализация. 58](#_Toc294788282)

[ЗАКЛЮЧЕНИЕ 68](#_Toc294788283)

[ЛИТЕРАТУРА 69](#_Toc294788284)

[ПРИЛОЖЕНИЕ 70](#_Toc294788285)

# ВВЕДЕНИЕ

Главное направление совершенствования мобильных телефонов можно определить одним понятием: конвергенция технологий. Мобильные телефоны объединили в себе едва ли не всё, что можно и разумно объединять под корпусом одного устройства.

Глядя на сегодняшний мобильный телефон, на какую-нибудь до предела функциональную и модную модель, можно подумать, что это – предел совершенства, но вспомните, как мы смотрели, например, на «кирпич» от ericsson году так в 1999-м и думали то же самое. Наверное, глядя на телефон в 2020-м, мы с улыбкой будем вспоминать «совершенные» аппараты 2011-го.

Основные задачи телефона прошлого – совершать и принимать звонки, писать sms. Сегодня эти задачи дополнилась работой с интернет-ресурсами, прослушиванием музыки, фотосъемкой, использованием игр и приложений. Мобильный будущего, скорее всего, добавит к списку основных задач просмотр телепрограмм, управление различной техникой, функции контроля состояния здоровья своего владельца и возможно многое другое.

Некоторые из перечисленных функций уже существуют в современных телефонах, но пока они относятся к разряду экзотических, а прогресс в области их создания, скорее всего, переведёт наиболее нужные из них в область обыденных. Скорее всего, у аппаратов будущего могут появиться такие возможности, о которых мы сегодня даже не подозреваем.

В будущем поменяются технологии изготовления телефонов. Изменения технологий могут быть настолько существенными, что, увидев сотовый телефон 2016 года, не сразу поймешь, что это такое. Причём особая прелесть ожидания перемен заключается в том, что, благодаря научно-техническим революциям могут появиться такие устройства, которые сейчас не то что кажутся невозможными, а просто не рассматриваются нами, находясь «за горизонтом» нашего воображения.

Сегодня на территории нашей страны активно эксплуатируется больше 200 млн. номеров мобильных операторов, о чем еще десять лет назад отечественные провайдеры могли только мечтать. Рынок мобильных технологий развивается настолько быстро, что за его изменениями поспевают не все, даже очень крупные компании. Последние пять лет показали, что современный оператор связи должен быть максимально гибким, держать руку на пульсе мировых новинок и изобретений. Тот, кто исповедует консервативный подход к развитию, достаточно быстро начнет терять рыночные позиции, будучи вынужденным измениться или уступить место другим компаниям.

Эксперты обращают внимание потребителей на тот факт, что сейчас на рынке доступно несколько мобильных платформ, стремительную популярность среди которых в последнее время приобретает iOS и Android. На рынке США компания Apple уже давно заняла пальму первенства в сегменте дорогих и презентабельных мобильных устройств, постоянно удивляя клиента интересными новинками. На примере Apple, видно как кардинально может измениться тот или иной рынок с появлением новых «девайсов».

Перспективы развития рынка мобильных технологий весьма благоприятны, поскольку изменяется не столько сам рынок, сколько потребитель. Если вчера мечтой многих был безлимитный тариф, то сегодня с широким распространением мобильного интернета прерогативы существенно изменились. С появлением на отечественном рынке 3G решений, произошло заметное перераспределение клиентов в пользу тех компаний, которые были готовы к запуску данной услуги. Современному владельцу мобильного устройства уже недостаточно просто общаться, ему нужен круглосуточный доступ в глобальную сеть, посредством которой можно не только находить нужную информацию, но и общаться с друзьями, сообщать о своем местонахождении и многое другое.

Что бы ни говорили, а мобильность сегодня — явление глобальное, проникающее во многие сферы нашей жизни. Почти половина всех пользователей мобильных услуг проживают в пяти странах, среди которых и Россия. Ныне потребитель ищет не просто мобильный телефон, он внимательно изучает условия, соотнося их с требованиями современного рынка. Именно поэтому можно утверждать, что развитие мобильных технологий будет только ускоряться.

Ниже в дипломной работе проводится исследование в области городских информационных систем на базе мобильных интерфейсов. Будут изучены подходы к реализации подобных систем, основные методики геокодирования в информационных системах ориентированных на определенную группу пользователей. Результатом дипломной работы станет прототипная реализация приложения (туристическая карта) для мобильных устройств на платформе Android. В заключении рассматриваются перспективы данного исследования.

# ГОРОДСКИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И МОБИЛЬНЫЕ ИНТЕРФЕЙСЫ

Материал для данного раздела основан на [1], [2], [3],[4].

## 1.1 Введение в ГИС

Географическая информационная система (ГИС, геоинформационная система) - программно-аппаратный комплекс, предназначенный для сбора, управления, анализа и отображения пространственно распределенной информации. ГИС - не только и не столько информационные системы для географии, сколько информационные системы с географически организованной информацией. В простейшем варианте географические информационные системы - сочетание обычных баз данных (атрибутивной информации) с электронными картами, то есть мощными графическими средствами.

Основная идея ГИС - связь данных на карте и в базе данных. ГИС - это и аналитические средства для работы с любой координатно-привязанной информацией. В принципе, ГИС можно рассматривать как некое расширение концепции баз данных. В этом смысле ГИС фактически представляет собой новый уровень и способ интеграции и структурирования информации. ГИС предлагает совершенно новый путь развития картографии. Преодолеваются основные недостатки обычных карт - их статичность и ограниченная емкость как носителя информации. В последние десятилетия бумажные карты из-за перегруженности информацией становятся нечитабельными. ГИС же обеспечивает управление визуализацией информации. Появляется возможность выводить (на экран, на твердую копию) только те объекты или их множества, которые интересуют нас в данный момент. Фактически осуществляется переход от сложных комплексных карт к серии взаимосвязанных частных карт. При этом улучшается структурированность информации, а, следовательно, повышается эффективность ее обработки и анализа. В ГИС карта оживает и становится действительно динамическим объектом в смысле:

• изменяемости масштаба;

• преобразования картографических проекций;

• варьирования объектным составом карты;

• возможности опрашивать через карту в режиме реального времени многочисленные базы данных;

• изменения способа отображения объектов (цвет, тип линии и т.п.), в том числе и определения символогии через значения атрибутов, то есть синхронизации визуализации с изменениями в базах данных;

• легкости внесения любых изменений.

ГИС являются классом информационных систем, имеющим свои особенности. Они построены с учетом закономерностей геоинформатики и методов, применяемых в этой науке. ГИС как интегрированные информационные системы предназначены для решения различных задач науки и производства на основе использования пространственно - локализованных данных об объектах и явлениях природы и общества. Неразрывно с ГИС связаны геоинформационные технологии. Геоинформационные технологии можно определить как совокупность программно-технологических средств получения новых видов информации об окружающем мире. Геоинформационные технологии предназначены для повышения эффективности: процессов управления, хранения и представления информации, обработки и поддержки принятия решений.

ГИС имеет ряд особенностей, которые необходимо учитывать при изучении этих систем. Одна из особенностей ГИС и геоинформационных технологий состоит в том, что они являются элементами информатизации общества. Это заключается во внедрении ГИС и геоинформационных технологий в науку, производство, образование и применение в практической деятельности получаемой информации об окружающей реальности. Геоинформационные технологии являются новыми информационными технологиями, направленными на достижение различных целей, включая информатизацию производственно-управленческих процессов. Другой особенностью ГИС является то, что как информационные системы они являются результатом эволюции этих систем и поэтому включают в себя основы построения и функционирования информационных систем. ГИС как система включает множество взаимосвязанных элементов, каждый из которых связан прямо или косвенно с каждым другим элементом, а два любые подмножества этого множества не могут быть независимыми не нарушая целостность, единство системы.

Автоматизированной информационной системой (АИС) называют организационно-техническую систему, использующую автоматизированные информационные технологии в целях обучения, информационно-аналитического обеспечения научно-инженерных работ и процессов управления. В соответствии с данным определением ГИС попадает в класс автоматизированных информационных систем.

Также особенностью ГИС является то, что она является интегрированной информационной системой. Интегрированные системы построены на принципах интеграции технологий различных систем. Они зачастую применяются настолько в разных областях, что их название часто не определяет все их возможности и функции. По этой причине не следует связывать ГИС с решением задач только геодезии или географии. "ГЕО" в названии геоинформационных систем и технологий определяет объект исследований, а не предметную область использования этих систем. Необходимо рассмотреть место ГИС среди других автоматизированных систем, что требует дать краткую классификацию этих систем. Выбирая различные аспекты рассмотрения автоматизированных информационных систем можно дать различные их различные классификации.

По принадлежности к конкретной предметной области можно подразделить информационные системы на три класса: технические, экономические, информационно-аналитические. К техническим относят автоматизированные системы научных исследований (АСНИ), системы автоматизированного проектирования (САПР), гибкие производственные системы (ГПС), робототехнические комплексы (РТК) и др. Информационно-аналитические автоматизированные системы включают: автоматизированные справочно-информационные системы (АСИС), базы данных (БД), экспертные системы (ЭС), статистические информационные системы (СТИС) и т.п. Примером экономических систем могут служить автоматизированные системы управления (АСУ), бухгалтерские информационные системы (БУ-ИС), банковские информационные системы (БИС), биржевые информационные системы (БИС), маркетинговые информационные системы (МИС) и др.

Особенностью ГИС как интегрированной системы является то, что она интегрирует технологии трех перечисленных выше классов систем: технических, информационно-аналитических и экономических. Следовательно, ГИС могут быть использованы как любая из этих систем.

## 1.2 Организация данных в ГИС

Тематические данные хранятся в ГИС в виде таблиц, поэтому проблем с их хранением и организацией в базах данных не возникает. Наибольшие проблемы представляет хранение и визуализация графических данных. Основой визуального представления данных при помощи ГИС-технологий служат специальные графические модели. Они подразделяются на векторные и растровые модели. В общем случае модели пространственных (координатных) данных могут иметь векторное или растровое (ячеистое) представление, содержать или не содержать топологические характеристики. Этот подход позволяет классифицировать модели по трем типам: растровая модель; векторная нетопологическая модель; векторная топологическая модель. Все эти модели взаимно преобразуемы. Тем не менее, при получении каждой из них необходимо учитывать их особенности. В ГИС форме представления координатных данных соответствуют два основных подкласса моделей - векторные и растровые (ячеистые или мозаичные). Возможен класс моделей, которые содержат характеристики, как векторов, так и мозаик. Они называются гибридными моделями. Между векторными и растровыми изображениями имеется различие, характерное именно для ГИС. Растровые изображения отображают поля данных, т.е. носят полевой характер. Векторные изображения в ГИС, как правило, отображают геоинформационные объекты, т.е. носят объектный характер.

Если векторная модель дает информацию о том, где расположен тот или иной объект, то растровая - информацию о том, что расположено в той или иной точке территории. Это определяет основное назначение растровых моделей - непрерывное отображение поверхности.

Растровые модели

В растровых моделях дискретизация осуществляется наиболее простым способом - весь объект (исследуемая территория) отображается в пространственные ячейки, образующие регулярную сеть. При этом каждой ячейке растровой модели соответствует одинаковый по размерам, но разный по характеристикам (цвет, плотность) участок поверхности объекта. В ячейке модели содержится одно значение, усредняющее характеристику участка поверхности объекта. В теории обработки изображений эта процедура известна под названием пикселизация.

Основное назначение растровых моделей - непрерывное отображение поверхности. В растровых моделях в качестве атомарной модели используют двухмерный элемент пространства - пиксель (ячейка). Упорядоченная совокупность атомарных моделей образует растр, который, в свою очередь, является моделью карты или геообьекта. Растровые модели позволяют отображать полутона и цветовые оттенки. Как правило, каждый элемент растра или каждая ячейка должны иметь лишь одно значение плотности или цвета. Это применимо не для всех случаев. Например, когда граница двух типов покрытий может проходить через центр элемента растра, элементу дается значение, характеризующее большую часть ячейки или ее центральную точку. Ряд систем позволяет иметь несколько значений для одного элемента растра.

Для растровых моделей существует ряд характеристик: разрешение, ориентация, зоны, значение, положение. Разрешение - минимальный линейный размер наименьшего участка отображаемого пространства (поверхности), отображаемый одним пикселем. Пиксели обычно представляют собой прямоугольники или квадраты, реже используются треугольники и шестиугольники. Более высоким разрешением обладает растр с меньшим размером ячеек. Высокое разрешение подразумевает обилие деталей, множество ячеек, минимальный размер ячеек. Значение - элемент информации, хранящийся в элементе растра (пикселе) или легенде. Положение обычно задается упорядоченной парой координат (номер строки и номер столбца), которые однозначно определяют положение каждого элемента отображаемого пространства в растре.

Проводя сравнение векторных и растровых моделей, отметим удобство векторных для организации и работы со взаимосвязями объектов. Тем не менее, используя простые приемы, например, включая взаимосвязи в таблицы атрибутов, можно организовать взаимосвязи и в растровых системах. Необходимо остановиться на вопросах точности отображения в растровых моделях. В растровых форматах в большинстве случаев неясно, относятся ли координаты к центральной точке пикселя или к одному из его углов. Поэтому точность привязки элемента растра определяют как 1/2 ширины и высоты ячейки.

Растровые модели имеют следующие достоинства:

* Растр не требует предварительного знакомства с явлениями, данные собираются с равномерно расположенной сети точек, что позволяет в дальнейшем на основе статистических методов обработки получать объективные характеристики исследуемых объектов. Благодаря этому растровые модели могут использоваться для изучения новых явлений, о которых не накоплен материал. В силу простоты этот способ получил наибольшее распространение;
* Растровые данные проще для обработки по параллельным алгоритмам и этим обеспечивают более высокое быстродействие по сравнению с векторными;
* Некоторые задачи, например создание буферной зоны, много проще решать в растровом виде;
* Многие растровые модели позволяют вводить векторные данные, в то время как обратная процедура весьма затруднительна для векторных моделей;
* Процессы растеризации много проще алгоритмически, чем процессы векторизации, которые зачастую требуют экспертных решений.

Векторные модели

Векторные модели строятся на векторах, занимающих часть пространства в отличие от занимающих все пространство растровых моделей. Это определяет их основное преимущество - требование на порядки меньшей памяти для хранения и меньших затрат времени на обработку и представление, а главное более высокая точность позиционирования и представления данных.

При построении векторных моделей объекты создаются путем соединения точек прямыми линиями, дугами окружностей, полилиниями. Площадные объекты - ареалы задаются наборами линий. Векторные модели используются преимущественно в транспортных, коммунальных, маркетинговых приложениях ГИС.

Системы ГИС, работающие в основном с векторными моделями, получили название векторных ГИС. В реальных ГИС имеют дело не с абстрактными линиями и точками, а с объектами, содержащими линии и ареалы, занимающими пространственное положение, а также со сложными взаимосвязями между ними. Поэтому полная векторная модель данных ГИС отображает пространственные данные как совокупность следующих основных частей: геометрические (метрические) объекты (точки, линии и полигоны); атрибуты - признаки, связанные с объектами; связи между объектами.

Векторные модели (объектов) используют в качестве элементарной модели последовательность координат, образующих линию. Линией называют границу, сегмент, цепь или дугу. Основные типы координатных данных в классе векторных моделей определяются через базовый элемент линия следующим образом. Точка определяется как выродившаяся линия нулевой длины, линия - как линия конечной длины, а площадь представляется последовательностью связанных между собой отрезков. Каждый участок линии может являться границей для двух ареалов либо двух пересечений (узлов). Отрезок общей границы между двумя пересечениями (узлами) имеет разные названия, которые являются синонимами в предметной области ГИС. Специалисты по теории графов предпочитают слову линия термин ребро, а для обозначения пересечения употребляют термин вершина. Национальным стандартом США официально санкционирован термин цепь. В некоторых системах (arclnfo, geodraw) используется термин дуга. В отличие от обычных векторов в геометрии дуги имеют свои атрибуты. Атрибуты дуг обозначают полигоны по обе стороны от них. По отношению к последовательному кодированию дуги эти полигоны именуются левый и правый. Понятие дуги {цепи, ребра) является фундаментальным для векторных ГИС векторные модели получают разными способами. Один из наиболее распространенных - векторизация сканированных (растровых) изображений. Векторизация - процедура выделения векторных объектов с растрового изображения и получение их в векторном формате. Для векторизации необходимо высокое качество (отчетливые линии и контуры) растровых образов. Чтобы обеспечить требуемую четкость линий иногда приходится заниматься улучшением качества изображения.

При векторизации возможны ошибки, исправление которых осуществляется в два этапа:

1) корректировка растрового изображения до его векторизации;

2) корректировка векторных объектов.

В векторных форматах набор данных определен объектами базы данных. Векторная модель может организовывать пространство в любой последовательности и дает "произвольный доступ" к данным. В векторной форме легче осуществляются операции с линейными и точечными объектами, например, анализ сети - разработка маршрутов движения по сети дорог, замена условных обозначений. В растровых форматах точечный объект должен занимать целую ячейку. Это создает ряд трудностей, связанных с соотношением размеров растра и размера объекта. Что касается точности векторных данных, то здесь можно говорить о преимуществе векторных моделей перед растровыми, так векторные данные могут кодироваться с любой мыслимой степенью точности, которая ограничивается лишь возможностями метода внутреннего представления координат. Обычно для представления векторных данных используется 8 или 16 десятичных знаков (одинарная или двойная точность).

## 1.3 ГИС среди информационных технологий

Первым вопросом человека, не знакомого с географическими информационными системами (ГИС), будет, конечно, "а зачем мне это нужно?".

На первый взгляд достаточно очевидным является только применение ГИС в подготовке и распечатке карт и, может быть, в обработке аэро- и космических снимков. Реальный же спектр применений ГИС гораздо шире, и чтобы оценить его, нам стоит взглянуть на применение компьютеров вообще, тогда место ГИС будет представляться гораздо яснее.

Компьютеры дают не только большее удобство выполнения известных операций с документами, они являются носителем нового направления человеческой деятельности - информационных технологий, и современное общество основано в значительной степени на них.

Информацией в нашем понимании следует называть все, что может быть представлено в виде букв, цифр и изображений. Так вот, все методы, техники, приемы, средства, системы, теории, направления и т.д., которые нацелены на сбор, переработку и использование информации, вместе называются информационными технологиями. И ГИС – одна из них.

Эта технология объединяет традиционные операции при работе с базами данных, такими, как запрос и статистический анализ, с преимуществами полноценной визуализации и географического (пространственного) анализа, которые предоставляет карта. Эти возможности отличают ГИС от других информационных систем и обеспечивают уникальные возможности для ее применения в широком спектре задач, связанных с анализом и прогнозом явлений и событий окружающего мира, с осмыслением и выделением главных факторов и причин, а также их возможных последствий, с планированием стратегических решений и текущих последствий предпринимаемых действий.

Создание карт и географический анализ не являются чем-то абсолютно новым. Однако технология ГИС предоставляет новый, более соответствующий современности, более эффективный, удобный и быстрый подход к анализу проблем и решению задач, стоящих перед человечеством в целом, и конкретной организацией или группой людей, в частности. Она автоматизирует процедуру анализа и прогноза.

До начала применения ГИС лишь немногие обладали искусством обобщения и полноценного анализа географической информации с целью обоснованного принятия оптимальных решений, основанных на современных подходах и средствах.

В настоящее время ГИС - это многомиллионная индустрия, в которую вовлечены сотни тысяч людей во всем мире. ГИС изучают в школах, колледжах и университетах. Эту технологию применяют практически во всех сферах человеческой деятельности - будь то анализ таких глобальных проблем как перенаселение, загрязнение территории, сокращение лесных угодий, природные катастрофы, так и решение частных задач, таких как поиск наилучшего маршрута между пунктами - экскурсионный и экстремальный туризм, подбор оптимального расположения нового офиса, поиск дома по его адресу, прокладка трубопровода на местности, различные муниципальные задачи.

Как же удается с помощью одной технологии решать столь разные задачи? Чтобы это понять, рассмотрим последовательно устройство, работу и примеры применения ГИС.

Составные части ГИС

Работающая ГИС включает в себя пять ключевых составляющих: аппаратные средства, программное обеспечение, данные, исполнители и методы.

* **Аппаратные средства.**

Это компьютер, на котором запущена ГИС. В настоящее время ГИС работают на различных типах компьютерных платформ, от централизованных серверов до отдельных компьютеров и мобильных устройств.

* **Программное обеспечение ГИС.**

По ГИС содержит функции и инструменты, необходимые для хранения, анализа и визуализации географической (пространственной) информации. Ключевыми компонентами программных продуктов являются: инструменты для ввода и оперирования географической информацией; система управления базой данных (dbms или субд); инструменты поддержки пространственных запросов, анализа и визуализации (отображения); графический пользовательский интерфейс (gui или гип) для легкого доступа к инструментам и функциям.

* **Данные.**

Это вероятно наиболее важный компонент ГИС. Данные о пространственном положении (географические данные) и связанные с ними табличные данные могут собираться и подготавливаться самим пользователем, либо приобретаться у поставщиков на коммерческой или другой основе. В процессе управления пространственными данными ГИС интегрирует пространственные данные с другими типами и источниками данных, а также может использовать СУБД, применяемые многими организациями для упорядочивания и поддержки имеющихся в их распоряжении данных

* **Исполнители.**

Широкое применение технологии ГИС невозможно без людей, которые работают с программными продуктами и разрабатывают планы их использования при решении реальных задач. Пользователями ГИС могут быть как технические специалисты, разрабатывающие и поддерживающие систему, так и конечные пользователи, которым ГИС помогает решать текущие каждодневные дела и проблемы.

* **Методы.**

Успешность и эффективность (в том числе экономическая) применения ГИС во многом зависит от правильно составленного плана и правил работы, которые составляются в соответствии со спецификой задач и работы каждой организации.

Задачи, которые решает ГИС.

ГИС общего назначения, в числе прочего, обычно выполняет пять процедур (задач) с данными: ввод, манипулирование, управление, запрос и анализ, визуализацию.

* **Ввод.**

Для использования в ГИС данные должны быть преобразованы в подходящий цифровой формат. Процесс преобразования данных с бумажных карт в компьютерные файлы называется оцифровкой.

* **Манипулирование.**

Часто для выполнения конкретного проекта имеющиеся данные нужно дополнительно видоизменить в соответствии с требованиями вашей системы. Например, географическая информация может быть в разных масштабах (осевые линии улиц имеются в масштабе 1: 100 000, границы округов переписи населения - в масштабе 1: 50 000, а жилые объекты - в масштабе 1: 10 000). Для совместной обработки и визуализации все данные удобнее представить в едином масштабе и одинаковой картографической проекции. Гис-технология предоставляет разные способы манипулирования пространственными данными и выделения данных, нужных для конкретной задачи.

* **Управление.**

В небольших проектах географическая информация может храниться в виде обычных файлов. Но при увеличении объема информации и росте числа пользователей для хранения, структурирования и управления данными эффективнее применять системы управления базами данных (субд), специальные компьютерные средства для работы с интегрированными наборами данных (базами данных). В ГИС наиболее удобно использовать реляционную структуру, при которой данные хранятся в табличной форме. При этом для связывания таблиц применяются общие поля. Этот простой подход достаточно гибок и широко используется во многих, как ГИС, так и не ГИС приложениях.

* **Запрос и анализ.**

При наличии ГИС и географической информации вы сможете получать ответы как на простые вопросы, так и на более сложные, требующие дополнительного анализа, запросы. Запросы можно задавать как простым щелчком мышью на определенном объекте, так и посредством развитых аналитических средств.

С помощью ГИС можно выявлять и задавать шаблоны для поиска, проигрывать сценарии по типу "что будет, если…". Современные ГИС имеют множество мощных инструментов для анализа, среди них наиболее значимы два: анализ близости и анализ наложения.

Для проведения анализа близости объектов относительно друг друга в ГИС применяется процесс, называемый буферизацией. Он помогает ответить на вопросы типа: сколько домов находится в пределах 100 м от этого водоема? Сколько покупателей живет не далее 1 км от данного магазина? Какова доля доходов от туристического бизнеса в казну города с данного участка побережья? Есть ли участки зон отдыха недостаточно обслуживаемые сервисными службами? Процесс наложения включает интеграцию данных, расположенных в разных тематических слоях. В простейшем случае это операция отображения, но при ряде аналитических операций данные из разных слоев объединяются физически. Наложение, или пространственное объединение, позволяет, например, интегрировать данные о почвах, уклоне, растительности и землевладении со ставками земельного налога.

* **Визуализация.**

Для многих типов пространственных операций конечным результатом является представление данных в виде карты или графика. Карта - это очень эффективный и информативный способ хранения, представления и передачи географической (имеющей пространственную привязку) информации. Раньше карты создавались на столетия. Гис предоставляет новые удивительные инструменты, расширяющие и развивающие искусство и научные основы картографии. С ее помощью визуализация самих карт может быть легко дополнена отчетными документами, трехмерными изображениями, графиками, таблицами, диаграммами, фотографиями и другими средствами, например, мультимедийными.

Связанные технологии.

Гис тесно связана с рядом других типов информационных систем. Ее основное отличие заключается в способности манипулировать и проводить анализ пространственных данных. Хотя и не существует единой общепринятой классификации информационных систем, приведенное ниже описание должно помочь дистанциировать ГИС от настольных картографических систем (desktop mapping), систем САПР (cad), дистанционного зондирования (remote sensing), систем управления базами данных (субд или dbms) и технологии глобального позиционирования (gps).

Системы настольного картографирования используют картографическое представление для организации взаимодействия пользователя с данными. В таких системах все основано на картах, карта является базой данных. Большинство систем настольного картографирования имеет ограниченные возможности управления данными, пространственного анализа и настройки. Соответствующие пакеты работают на настольных компьютерах - pc, macintosh и младших моделях unix рабочих станций.

## 1.4 Что ГИС могут сделать для туризма

Пожалуй, главным козырем ГИС является наиболее "естественное" (для человека) представление как собственно пространственной информации, так и любой другой информации, имеющей отношение к объектам, расположенным в пространстве (т.н. атрибутивной информации). Способы представления атрибутивной информации различны: это может быть числовое значение с датчика, таблица из базы данных (как локальной, так и удаленной) о характеристиках объекта, его фотография, или реальное видеоизображение, наконец, звуковая запись.

Таким образом, ГИС могут помочь везде, где используется пространственная информация и информация об объектах, находящихся в определенных местах пространства.

Если же посмотреть на некоторые области и экономический эффект применения ГИС, то они могут делать пространственные запросы и проводить анализ. Способность ГИС проводить поиск в базах данных и осуществлять пространственные запросы позволила многим компаниях заработать миллионы долларов.

Так, например, на создание туристско–ориентированной геоинформационной системы города пинава (pinawa) и окружающих его территорий (канада) было затрачено $82,500 за 3 года, а за один год система приносит $5 000 000.

Гис помогает сократить время получения ответов на запросы клиентов; выявлять территории подходящие для требуемых мероприятий; выявлять взаимосвязи между различными параметрами (например, почвами, климатом и урожайностью с/х культур);

## 1.5 Предметно ориентированные ГИС

Зачастую имеющие статус государственных, региональные ГИС (regional gis), субрегиональные ГИС и локальные, или местные ГИС (local gis).

Гис различаются предметной областью информационного моделирования, к примеру, городские ГИС, или муниципальные ГИС, МГИС (urban gis), природоохранные ГИС (environmental gis), туристические т.п.

Среди них особое наименование, как особо широко распространенные, получили земельные информационные системы. Проблемная ориентация ГИС определяется решаемыми в ней задачами (научными и прикладными), среди них инвентаризация ресурсов (в том числе кадастр), анализ, оценка, мониторинг, управление и планирование, поддержка принятия решений. Интегрированные ГИС, ИГИС (integrated gis, igis) совмещают функциональные возможности ГИС и систем цифровой обработки изображений (данных дистанционного зондирования) в единой интегрированной среде.

Научные, технические, технологические и прикладные аспекты проектирования, создания и использования ГИС изучаются геоинформатикой.

## 1.6 Основные концепции информационной визуализации, используемые в ГИС

Информационная визуализация - это процесс формирования ментальной модели данных, посредством чего обеспечивается представление об их внутренней структуре. Задача информационной визуализации − представление и выявление (возможно скрытых) взаимоотношений, структур и отдельных характеристик изучаемых данных.

В связи с этим можно рассмотреть набор операций информационной визуализации, состоящий из следующих пунктов:

− общий обзор;

− изменение размера фрагмента (зуминг);

− фильтрация;

−- детализация по запросу;

− установление зависимостей по определенному признаку;

− извлечение данных по заданному признаку;

− сохранение истории (то есть запись последовательности операций).

Общий обзор дает краткий обзор всей совокупности данных. Он также позволяет видеть полную совокупность плюс подробное представление интересующего фрагмента. За счет зуминга можно увеличить или уменьшить масштаб фрагмента изображения, выбранного в общем обзоре. Фильтрация отбрасывает неинтересные элементы, позволяя пользователям управлять содержанием экрана. При детализации пользователь выбирает элемент или группу элементов и получает детальную информацию о них. Для установления зависимости по признаку пользователь назначает некий признак (атрибут), который позволяет отобрать нужные элементы. Упорядочение элементов массива также строится на основе установления связи с некоторыми признаками (дата записи, алфавитный порядок). Извлечение данных по признаку позволяет извлекать подмножество элементов, удовлетворяющих назначенному признаку (возраст, пол, хобби) и работать с ними. И, наконец, сохранение истории позволяет в случае необходимости последовательно отменить ряд предыдущих команд и вернуться к первоначальному состоянию, например, к первичному тексту документа.

## 1.7 ГИС и мобильные интерфейсы

Мобильные ГИС являются важным компонентом развитых геоинформационных систем и предназначены для применения непосредственно вне помещения. Они объединяют gps-приемник, мобильный компьютер и программное обеспечение, позволяющие определять местоположение на местности, визуализировать пространственные данные, обращаться к географическим базам данных в реальном времени, осуществлять сбор и анализ данных непосредственно на изучаемых в поле объектах. Эти системы быстро развиваются и применяются в широком спектре задач, связанных с работами на местности.

В настоящий период отчетливо наметилась тенденция интеграции спутниковых систем позиционирования с мобильными компьютерами. На базе объединения gps-приемника в одном корпусе с мощным карманным компьютером, работающим под управлением специализированного программного обеспечения, создаются высокоэффективные мобильные ГИС.

# ГЕОКОДИРОВАНИЕ В ИС

В данном разделе рассматриваются операции геокодирования, которые позволяют создавать новые графические объекты карты на основании табличных данных. В зависимости от выбранного метода геокодирования, при создании новых объектов может также использоваться имеющаяся информация о существующих объектах карты.

Материал раздела основан на [3],[4].

## 2.1 Понятие геокодирования.

Геокодирование – это процедура автоматизированного создания объектов карты на основании атрибутивных данных, содержащихся в некоторой таблице. В зависимости от характера используемых данных различаются координатное геокодирование, геокодирование по объектам и адресное геокодирование.

## 2.2 Назначение геокодирования

При решении различных прикладных задач с помощью ГИС нередко возникает задача размещения на карте объектов, построенных на основе информации, заданной в виде таблицы. В простейшем случае такая таблица может в явном виде содержать координаты некоторых точек, в которых требуется разместить объекты. В более сложных случаях, помимо исходной таблицы, может использоваться и другая информация, имеющаяся в гис, такая как координаты существующих объектов и табличная информация об объектах.

Процедура автоматизированного создания объектов карты на основании атрибутивных данных, содержащихся в некоторой таблице, называется геокодированием.

В выполнении геокодирования, как правило, участвуют, как минимум, два компонента гбд:

* Таблица, содержащая исходные данные для геокодирования (геокодируемая таблица);
* Точечный тип объектов одного из слоев карты (геокодируемый тип). К этому типу будут принадлежать объекты, создаваемые при геокодировании.

## 2.3 Методы геокодирования

**Координатное геокодирование** требует для своего выполнения, чтобы в двух полях геокодируемой таблицы содержались значения координат x и y. При выполнении геокодирования для каждой записи таблицы создается точечный объект с указанными координатами.

При выполнении координатного геокодирования можно также выполнить трансформацию координат создаваемых объектов. Это полезно в том случае, если координаты, содержащиеся в таблице, заданы в системе координат, отличающейся от системы координат данной карты.

Координатное геокодирование – это самый простой из методов геокодирования, который в то же время позволяет наиболее точно разместить создаваемые объекты на карте.

**Геокодирование по объектам** основано на привязке создаваемых точечных объектов к положению существующих объектов какого-либо типа. Такой метод геокодирования применяется, например, в том случае, если геокодируемая таблица содержит список адресов интересующих пользователя объектов (магазинов, станций обслуживания и т.п.).

Определение положения объекта по его адресу – это задача, которую каждому человеку не раз приходилось решать на практике. Для ее решения достаточно иметь подробный план города, где показаны все здания и указаны их адреса. Тогда положение искомого объекта можно определить с точностью до здания.

В соответствии с этой аналогией, для геокодирования по объектам необходимо, кроме исходной геокодируемой таблицы и геокодируемого типа, указать еще два компонента гбд, участвующих в операции:

* Ссылочный тип – тип объектов, к положению которых на карте будут привязываться создаваемые объекты (например, в качестве ссылочного можно использовать тип объектов «здание»);
* Ссылочную таблицу – таблицу, записи которой связаны с объектами ссылочного типа и в то же время содержат данные, позволяющие с каждой записью геокодируемой таблицы связать запись ссылочной таблицы. Таким образом, через посредство ссылочной таблицы можно для каждой записи геокодируемой таблицы найти соответствующий объект ссылочного типа и использовать этот объект для определения положения создаваемого объекта на карте.

Если ссылочный тип – «здания», то в качестве ссылочной таблицы можно использовать таблицу «паспорта зданий», записи которой содержат, в частности, адреса соответствующих зданий.

По сравнению с координатным геокодированием, применение геокодирования по объектам дает меньшую точность, но зато не требует явного указания координат создаваемых объектов.

**Адресное геокодирование** основано на приближенном вычислении положения создаваемого объекта относительно заданного линейного объекта.

Продолжая аналогию с использованием плана города, можно представить себе такой план, на котором не указаны адреса конкретных зданий, но для каждой улицы на каждом квартале нанесены минимальное и максимальное значение адресов зданий этого квартала. Если предположить, что здания в некотором квартале имеют номера от 100 до 120, то объект, который находится в здании с номером 118, должен быть размещен вблизи конца квартала.

Для выполнения адресного геокодирования необходимо, кроме исходной геокодируемой таблицы и геокодируемого типа, указать еще два компонента гбд, участвующих в операции:

* Ссылочный тип – линейный тип объектов, относительно которых будет вычисляться положение на карте создаваемых объектов. В качестве ссылочного типа можно, например, использовать тип объектов «сегменты улиц», который содержит отрезки улиц, соответствующие отдельным кварталам;
* Ссылочную таблицу сегментов – таблицу, записи которой, во-первых, связаны с объектами ссылочного типа, во-вторых, содержат данные, позволяющие с каждой записью геокодируемой таблицы связать запись ссылочной таблицы и вычислить положение создаваемого объекта.

Если ссылочный тип – «сегменты улиц», то в качестве ссылочной таблицы можно использовать таблицу «сегменты», записи которой содержат код улицы и диапазон адресов зданий соответствующего сегмента.

Точность определения координат объектов при адресном геокодировании зависит от того, насколько сильно различаются по размерам объекты, соответствующие различным адресам. Если в приведенном выше примере здание номер 120 занимает половину всего квартала, то положение здания номер 118 будет вычислено весьма неточно.

Для любого из описанных методов геокодирования имеется дополнительная возможность установить связи между создаваемыми объектами и записями таблиц. В качестве связанных таблиц могут выступать как сама геокодируемая таблица, так и другие таблицы гбд. В последнем случае информация из геокодируемой таблицы используется для определения записи связанной таблицы, присоединяемой к создаваемому объекту.

# ОСНОВНЫЕ ТРЕБОВАНИЯ К ТУРИСТИЧЕСКОЙ КАРТЕ

Целью работы является разработка туристической карты для мобильных устройств, работающих на платформе Android.

Данное приложение должно быть «электронным помощником» туриста, используя которое он смог бы без труда познакомиться с достопримечательностями города.

Туристическая карта должна быть проста в использовании, но в то же время она должна содержать всю информацию, которая необходима как туристу, впервые прибывшему в Екатеринбург, так и тому, кто хочет лучше узнать свой город.

Основываясь на исследованиях, приведенных в предыдущих главах, я пришла к выводу, что приложение должно иметь следующие особенности:

* Мобильность — можно воспользоваться приложением, где бы вы ни находились и в любое время.
* Полноценная, постоянно обновляемая карта города.
* Возможность добавления на карту тематических слоев, содержащих дополнительную информацию.
* Возможность определить свое местоположение.
* Наличие информации об объектах туризма, а именно справочник, в котором данные структурированы определенным образом.
* Возможность поиска нужного объекта по различным критериям, просмотра информации о нем.
* Возможность отображения на карте как отдельного объекта, так и всей группы, к которой данный объект принадлежит.

# ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ

Во время проведения исследований было решено разрабатывать наше приложение для мобильных устройств работающих под управлением платформы android, т.к. данная платформа по мнению аналитиков и экспертов захватывает рынок мобильных устройств постепенно вытесняя с него общепризнанных лидеров.

Материал взят из [5], [6], [7], [8].

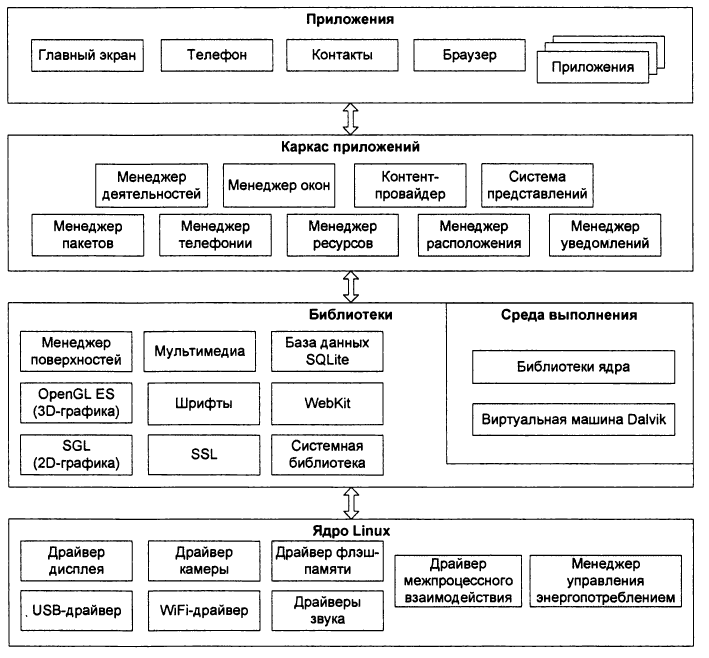
## 4.1 Операционная система Google Android

Система Android – это программный стек для мобильных устройств, который включает операционную систему, программное обеспечение промежуточного слоя (middleware), а также основные пользовательские приложения (e-mail-клиент, календарь, карты, браузер, контакты и др.).

Уровень ядра

Ядро является слоем абстракции между оборудованием и остальной частью программного стека. На этом уровне располагаются основные службы типа управления процессами, распределения памяти и управления файловой системой.

Ядро android основано на ядре Linux 2.6, но сама система Android не является Linux-системой в чистом виде, имеет некоторые отличия и содержит дополнительные расширения ядра, специфичные для Android, - свои механизмы распределения памяти, взаимодействие между процессами и др.



**Рис. 1** Архитектура системы Android

Основные компоненты уровня ядра:

* Драйвер межпроцессорного взаимодействия (ipc driver);
* Драйвер управления питанием (android power management);
* Набор драйверов для оборудования, входящего в состав мобильного устройства.

Рассмотрим теперь кратко основные компоненты ядра android.

**Драйвер IPC.** Приложения и службы могут работать в защищенных отдельных процессах, которые должны общаться между собой и иметь доступ к общим данным. Платформа android обеспечивает механизм ipc (inter-process communication), который является основным механизмом взаимодействия между процессами.

Драйвер IPC обеспечивает следующую функциональность:

* Взаимодействие процессов;
* Создание и обработку пулов потоков в процессах;
* Подсчет и отображение ссылок на объекты в других процессах;
* Синхронные запросы между процессами.

**Управление энергопотреблением.** Система управления энергопотреблением (android power management) разработана на основе стандартного драйвера управления питанием linux, но оптимизирована для мобильных устройств с учетом их специфических особенностей.

Основная функция системы управления энергопотреблением – экономное использование батареи мобильного устройства. Драйвер переводит системы в «спящий режим» с минимальным потреблением мощности процессором, если приложения и службы не используются.

**Драйверы оборудования.** Программный стек android разработан с учетом необходимой гибкости, включая работу со многими дополнительными компонентами, имеющимися в мобильных устройствах. Эти компоненты в значительной степени полагаются на доступность определенных аппаратных средств на данном устройстве. Они предоставляют дополнительную фунциональность для мобильных устройств (сенсорный экран, камера, gps, акселерометр и т.д.).

Уровень библиотек

Следующий уровень над ядром Linux является набором библиотек c/c++ типа openGL, webkit, freetype, ssl, библиотеки поддержки libc, базы данных SQLite и мультимедиабиблиотек (media framework). Системная библиотека базируется на berkeley software distribution (bsd) и разработана для мобильных устройств на основе Linux.

Следующий уровень над ядром Linux включает набор библиотек c/c++, используемых различными компонентами ос. Для разработчиков доступ к функциям этих библиотек реализован через использование application framework – каркаса приложений. Библиотеки этого уровня по своему функциональному назначению можно разделить на следующие группы:

* Системная библиотека с;
* Менеджер поверхностей;
* Функциональные библиотеки с/с++.

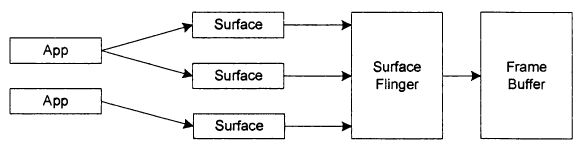
**Системная библиотека libc.** Компания Google разработала собственную с-библиотеку (libc) – bionic. Это было необходимо по следующим причинам:

* Библиотека будет загружаться в каждый процесс и, следовательно, должна иметь маленький размер. Библиотека bionic имеет размер около 200 кбайт, что в два раза меньше размера glibc;
* Ограниченная мощность центрального процессора мобильного устройства. Это означает, что библиотека должна быть оптимизирована для максимального быстродействия.

библиотека bionic имеет встроенную поддержку важных для android системных служб и регистрацию системных событий. Библиотека bionic не поддерживает определенные функциональности, например исключения с++, и несовместима с gnu libc и стандартом posix.

**Менеджер поверхностей.** Система android использует композитный менеджер поверхностей, похожий на compiz (композитный менеджер для x window system, использующий для ускорения 3d-графику opengl). Вместо того чтобы рисовать непосредственно в буфер экрана, команды рисунка входят за кадром в битовые массивы, которые потом объединяются с другими битовыми массивам, чтобы сформировать изображение, которое видит пользователь. Это позволяет системе создавать все виды интересных эффектов, например прозрачные окна и причудливые переходы.

Менеджер поверхностей обрабатывает весь рендеринг поверхностей на фреймовый буфер. Менеджер может объединить 2d- и 3d-поверхности и поверхности от нескольких приложений. Поверхности передаются как буферы компоновкой ipc-запросов. Менеджер поверхностей использует двойную буферизацию, используя транспонирования страницы. Системные интеграторы могут подключать аппаратное 2d-ускорение, используя плагины khronos. Обработка графической информации менеджером поверхностей представлена на следующем рисунке



**Рис. 2.** Обработка графической информации менеджером поверхностей

**Функциональные библиотеки.** Android включает ряд библиотек с/с++, используемых различными компонентами системы. Далее приводятся основные функциональные библиотеки системы.

* Мультимедиа (media framework). Эти библиотеки ответственны за регистрацию и воспроизведение аудио- и видеоформатов. Основаны на packetvideo opencore и предназначены для поддержки популярных аудио- и видеоформатов (mpeg4, h.264, mp3 и др.).
* SQLite – процессор баз данных, доступный всем приложениям. SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (api) библиотеки sqlite. Такой подход уменьшает накладные расходы, время отклика и упрощает разработку приложений для работы с данными. Sqlite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа.
* OpenGL es – движок для работы с 3d-графикой, основанный на api-версии openGL es 1.0. OpenGL es – это упрощенная версия спецификации openGL, позволяющая мобильным устройствам работать с тяжеловесными в графическом отношении приложениями. Библиотека использует аппаратный 3d-акселератор (если он доступен на этом устройстве) или встроенное высоко оптимизированное трехмерное программное обеспечение для работы с растровой графикой.
* Freetype – библиотека шрифтов, предназначенная для работы с растровыми и векторными шрифтами.
* Webkit – библиотека, предназначенная для функционирования встроенного в android web-браузера. Webkit поддерживает css, javascript, dom, ajax.
* SGL – движок для работы с 2d-графикой. Android также поддерживает собственную графическую 2d-библиотеку skia, которая написана на языке c и с++ (skia также формирует ядро популярного браузера google chrome).
* SSL – библиотека предназначена для работы с сокетами, основанная на протоколе ssl: sslv3.0 или tslv1.2.

Среда выполнения

Среда выполнения обеспечивает библиотеки ядра dalvik virtual machine (виртуальная машина dalvik), которые предоставляют требуемую функциональность для java-приложений.

**Dalvik virtual machine.** Прикладное программное обеспечение, запускаемое на мобильном устройстве, исполняет виртуальная машина dalvik, которая хоть и является аналогом виртуальной машины java, существенно от нее отличается. Dalvik относится к классу регистровых машин (регистры процессора используются как первичные модули хранения данных), идеально подходящих для работы на процессорах risc-архитектуры, к которым относятся и процессоры arm, применяемые в мобильных устройствах, тогда как стандартная виртуальная машина java компании sun microsystems – стековая. В результате использования регистровой виртуальной машины Google надеется на 30 процентов уменьшить количество команд по сравнению со стековыми машинами.

Созданные с помощью стандартного java-компилятора class-файлы и преобразуются в байт-код dalvik (\*.dex) транслятором dx, входящим в состав sdk. Изнутри работающий Android выглядит как набор виртуальных машин dalvik, в каждой из которых исполняется прикладная задача.

Виртуальная машина dalvik, на которой построена вся операционная система Google Android, дает разработчикам приложений удобный механизм для написаний приложений, которым не принципиален объем используемой памяти и мощность процессора.

**Core libraries.** Включает набор основных библиотек, которые предоставляют функциональность для java. Библиотеки ядра обеспечивают слой API и являются основной платформой разработки java-приложений для Android.

Уровень каркаса приложений.

Уровень каркаса приложений находится на вершине системных библиотек, функциональных библиотек и dalvik vm. На этом уровне находятся основные службы Android для управления жизненным циклом приложений, пакетами, ресурсами и т.д.

Программист имеет полный доступ к тем же API, которые используются основными приложениями. Архитектура этих приложений разработана с целью упрощения многократного использования компонентов. Любое разрабатываемое приложение может использовать возможности базовых приложений и, соответственно, любое стороннее приложение может использовать возможности вашего приложения (с учетом установленных разрешений). Этот же самый механизм позволяет многократно использовать уже разработанные компоненты.

Службы Android – это службы, которые являются основными для всех приложений, работающих на устройстве. К ним относятся:

* Менеджер деятельностей (activity manager) – управляет жизненным циклом приложений и предоставляет систему навигации по истории работы с деятельностями (стеку деятельностей);
* Менеджер пакетов (package manager) – управляет установкой и развертыванием пакетов прикладных программ, которые находятся на устройстве;
* Менеджер окон (window manager) – сохраняет окна приложения. Если разработчик предусмотрел вывод экрана, а затем переключение на другой экран, первый будет сохранен операционной системой и поставлен в режим ожидания. Это, в свою очередь, позволяет с помощью клавиши <back> мобильного устройства просматривать уже использовавшиеся экраны подобно тому, как это делается в web-браузере;
* Менеджер ресурсов (resource manager) – предназначен для доступа к строковым, графическим и другим типам ресурсов;
* Контент-провайдеры (content providers) – службы, которые позволяют приложениям получать доступ к данным других приложений, а также предоставлять сторонним приложениям доступ к своим данным;
* Система представлений (view system) – система с расширяемой функциональностью, которая служит для создания внешнего вида приложений, включающего такие компоненты, как списки, таблицы, поля ввода, кнопки, встроенный web-браузер и многое другое;
* Телефонный менеджер – обеспечивает слой api, контролирующий основную телефонную информацию, такую как сетевой тип и статус подключения, а также предоставляет различные утилиты для управления телефонными номерами;
* Менеджер местоположения – навигационные службы, которые позволяют приложениям получать периодические обновления географического местоположения устройства или запускать определенное приложение;
* Менеджер уведомлений – позволяет любому приложению отображать пользовательские уведомления в строке состояния.

Уровень приложений

Мобильное устройство Android поставляется с набором основных приложений, включая почтового клиента, программу для работы с sms, календарь, навигационные карты, браузер, контакты и др.

Что интересно, платформа Android не делает разницы между основными приложениями телефона сторонним программным обеспечением – таким образом, ключевые приложения, входящие в стандартный набор программного обеспечения, можно заменить при желании альтернативными приложениями. Программы для Android пишутся на языке java.

При разработке приложений программисты имеют полный доступ ко всей функциональности операционной системы. Архитектура приложений построена так, чтобы было легко использовать основные компоненты предоставляемые системой. Также есть возможность создавать свои компоненты и предоставлять их в открытое использование.

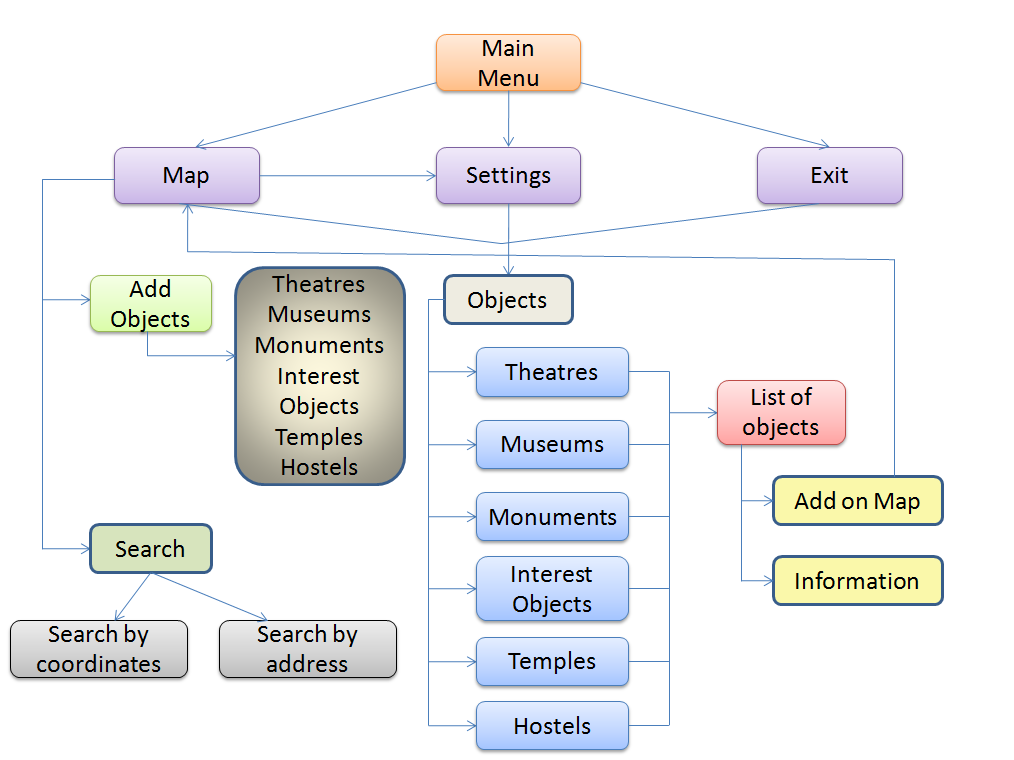
# ОПИСАНИЕ ПРОЕКТА ПРИЛОЖЕНИЯ

## 5.1 Структура программы.

Одним из компонентов android-приложения является деятельность (activity). Деятельность представляет собой визуальный пользовательский интерфейс для приложения – окно. Как правило, окно полностью заполняет экран мобильного устройства, но может иметь размеры меньше чем у экрана. Деятельность может также использовать дополнительные окна, например всплывающее диалоговое окно, которое запрашивает пользовательский ответ для основной деятельности, или окно уведомления о каком-либо событии в приложении или системе. Приложение может содержать несколько деятельностей. Каждая деятельность независима от других. При открытии новой деятельности работа предыдущей приостанавливается, а сама она вносится и сохраняется в стек деятельностей.

Деятельности могут взаимодействовать между собой. Как правило, одна из деятельностей приложения (окно которое открывается при запуске приложения) помечено как главное. Также из текущей деятельности можно запустить любую другую деятельность, даже если она определена в другом приложении.

На рисунке 3 показано дерево взаимодействия activities (деятельностей) программы.



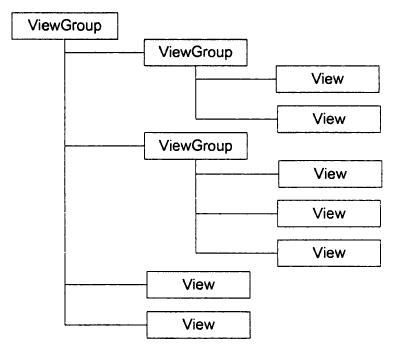
**Рис. 3** Дерево взаимодействия activities (деятельностей) программы.

## 5.2 Описание компонентов программы

На платформе android необходимо определить пользовательский интерфейс для каждой деятельности, используя иерархии узлов view и viewgroup, как показано на рисунке 4.

Объекты view – основные модули отображения интерфейса пользователя на платформе android. Объект view – структура данных свойства которой сохраняют параметры разметки и содержание для определенной прямоугольной области экрана. Как объект в интерфейсе пользователя, объект view является точкой взаимодействия пользователя и программы.

Класс viewgroup служит ядром для подклассов, называемых разметками (layouts), которые формируют расположение элементов пользовательского интерфейса на форме, использую различные виды архитектуры разметки – фреймовый, линейный, табличный и относительный.



**Рис. 4** пример дерева представлений для деятельностей

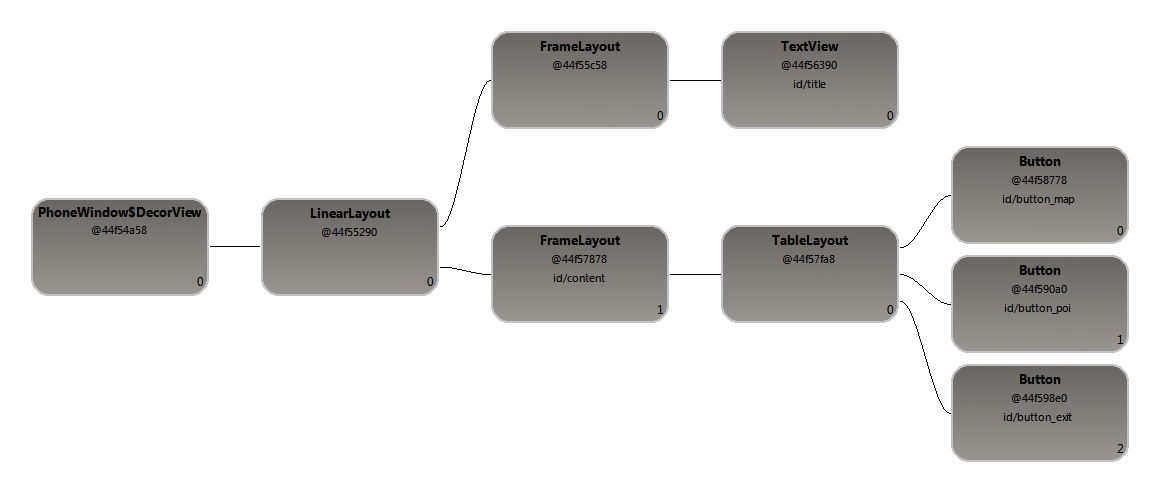
Главная деятельность (Main Menu)

Рассмотрим структуру главной деятельности. Внешний вид окна представлен на рисунке 5.



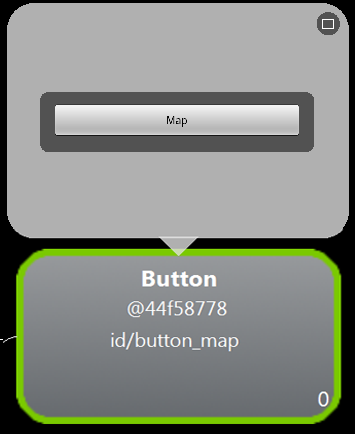
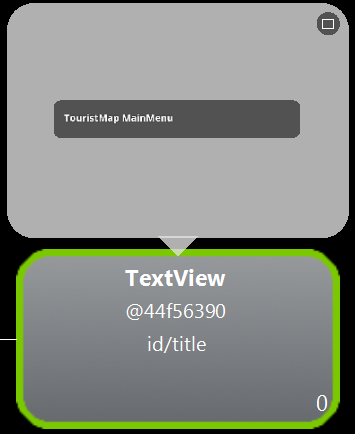
**Рис. 5** Главное окно программы

Рисунок 6 отображает иерархию разметки главного окна программы и её свойства.



**Рис. 6** Иерархия разметки главного окна.

Как видно из рисунка конечными элементами разметки, которые видит пользователь, являются следующие элементы: текстовое поле (title), 3 кнопки (button\_map, button\_poi, button\_exit) (рисунок 7).



**Рис.7** Элементы разметки

При нажатии на кнопку <map> открывается активити содержащую карту Google map.

При нажатии на кнопку <poi> открывается активити содержащее список групп туристических объектов города Екатеринбурга.

При нажатии на кнопку <exit> программа закрывается.

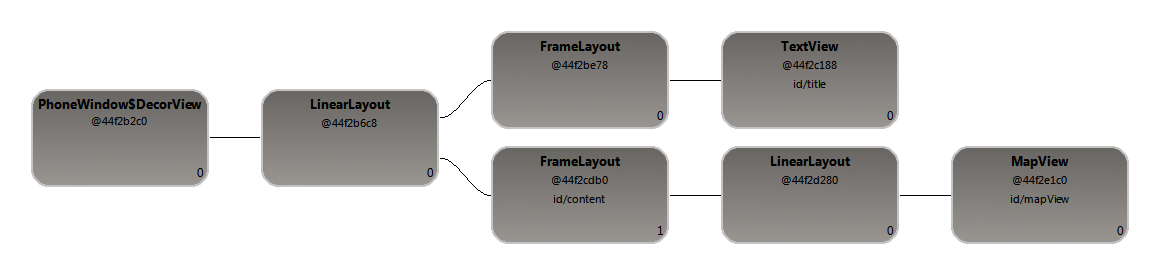
Деятельность map

Рассмотрим структуру деятельности map. Внешний вид окна представлен на рисунке 8.



**Рис.8** Окно map

Рисунок 9 отображает иерархию разметки окна map.



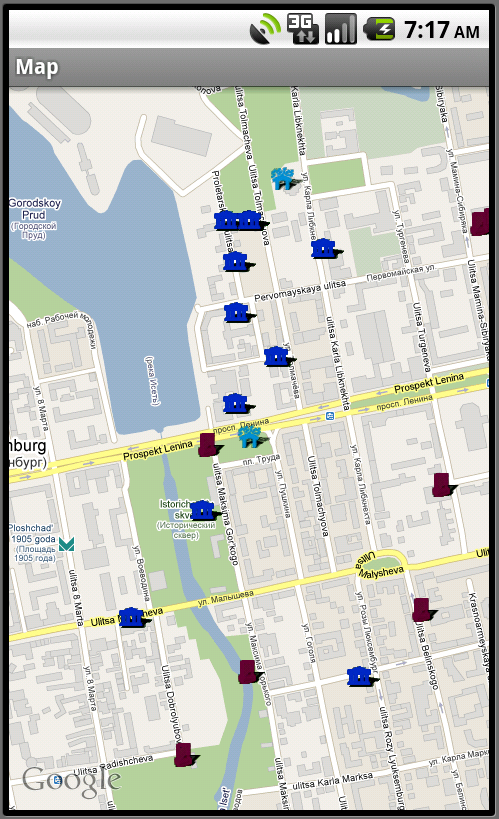
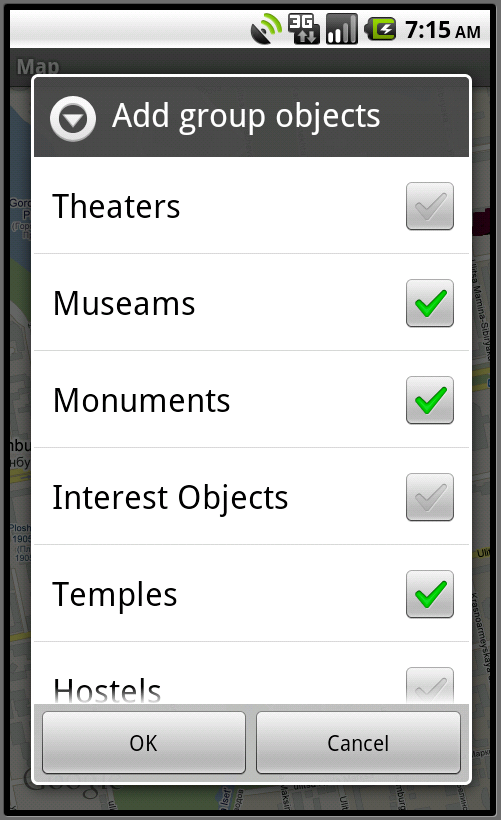
**Рис.9** Иерархия разметки окна map

При нажатии на кнопку <menu> мобильного устройства, на экране появляется меню из четырех пунктов poi, add objects, search, exit.

При выборе пункта poi открывается активити, содержащее список групп туристических объектов города Екатеринбурга.

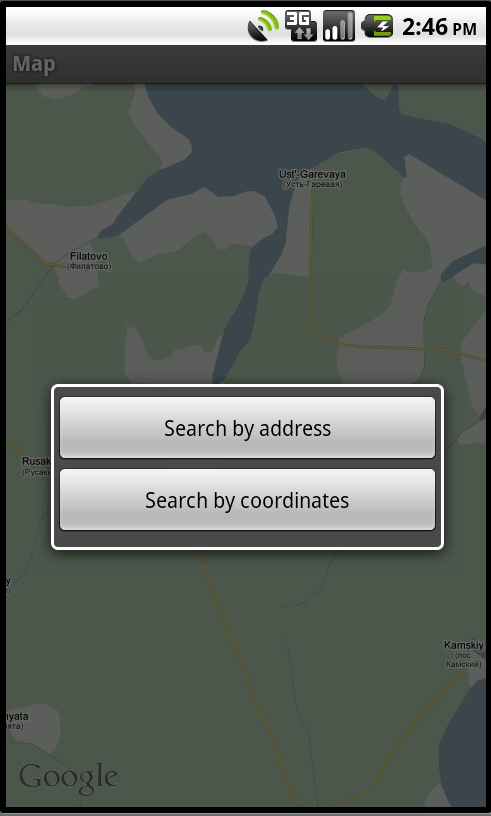
При выборе пункта add objects открывается всплывающее окно содержащее список групп объектов, которые можно отобразить на карте при активации чекбокса (рисунок 10).

При нажатии на маркер объекта появляется всплывающее окно с краткой информацией об объекте.



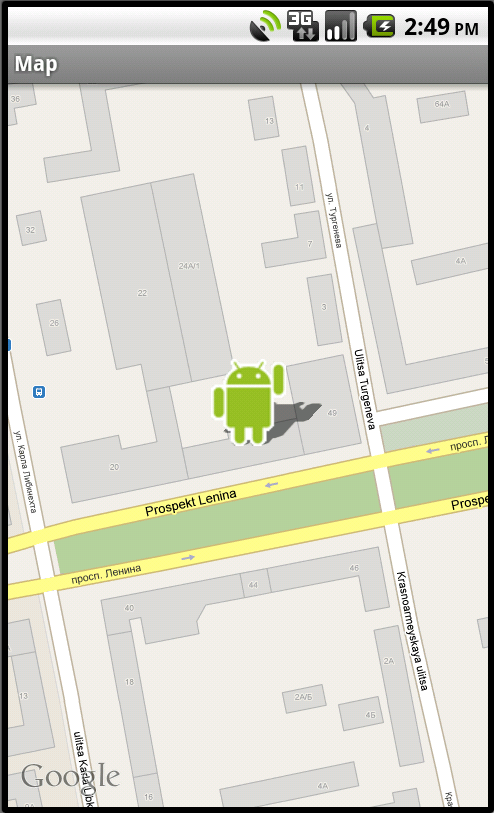
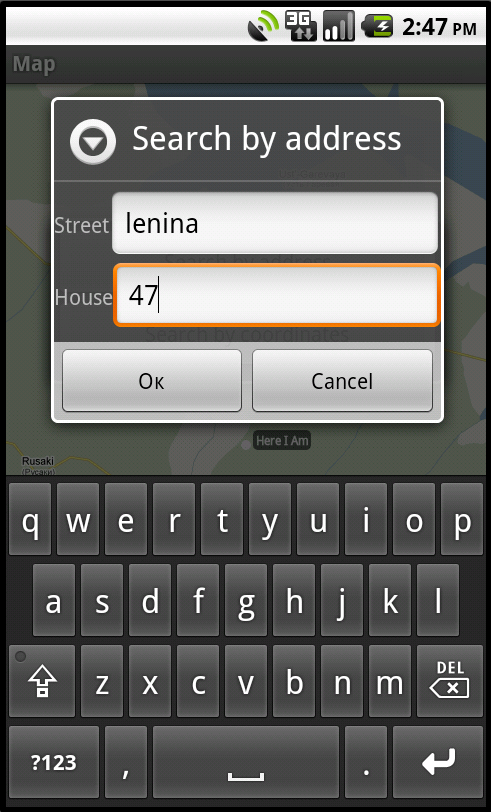
**Рис. 10** Выбор и отображение групп объектов на карте.

При выборе пункта search появляется всплывающее меню с двумя кнопками (search by address, search by coordinates) (рисунок 11).



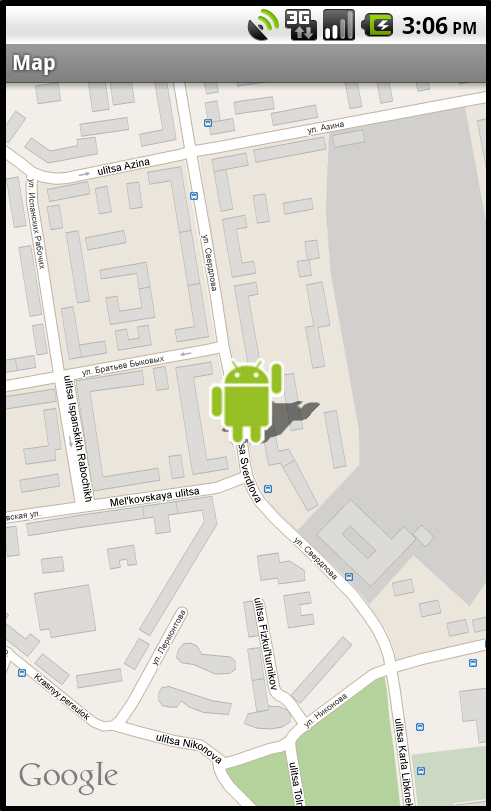
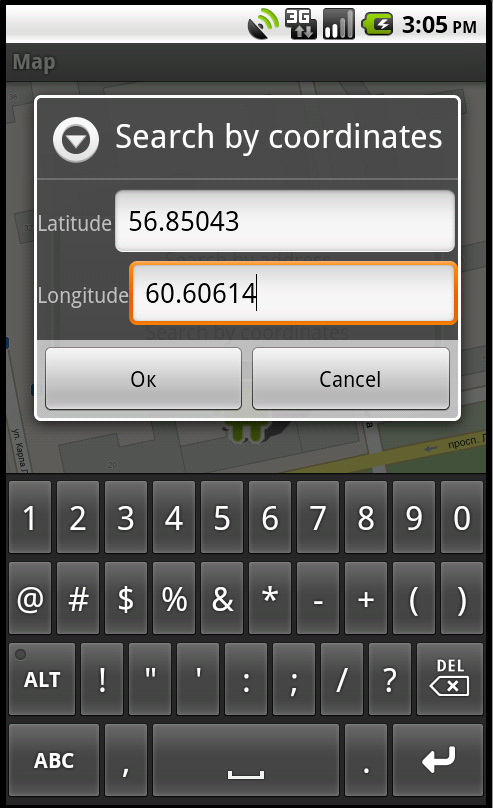
**Рис. 11** Всплывающее меню search

При нажатии на кнопку <Search by address> открывается форма, содержащая два поля (street и house) и две кнопки (Ok и Cancel) . При нажатии на кнопку Ok на карте отображается маркер, соответствующий введенному адресу (рисунок 12).



**Рис. 12** Поиск по адресу

При нажатии на кнопку <Search by coordinates> открывается форма, содержащая два поля (Latitude и Longitude) и две кнопки (Ok и Cancel) . При нажатии на кнопку Ok на карте отображается маркер, соответствующий введенным координатам (рисунок 13).

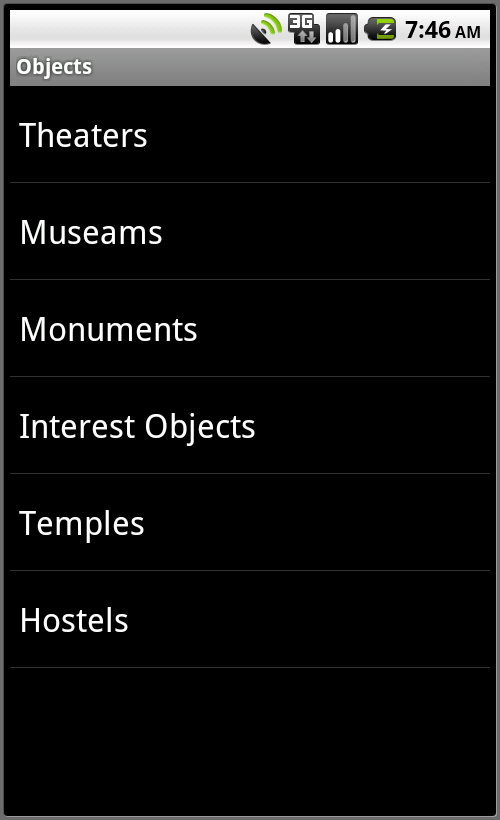


**Рис. 13** Поиск по координатам

При выборе пункта меню exit программа закрывается.

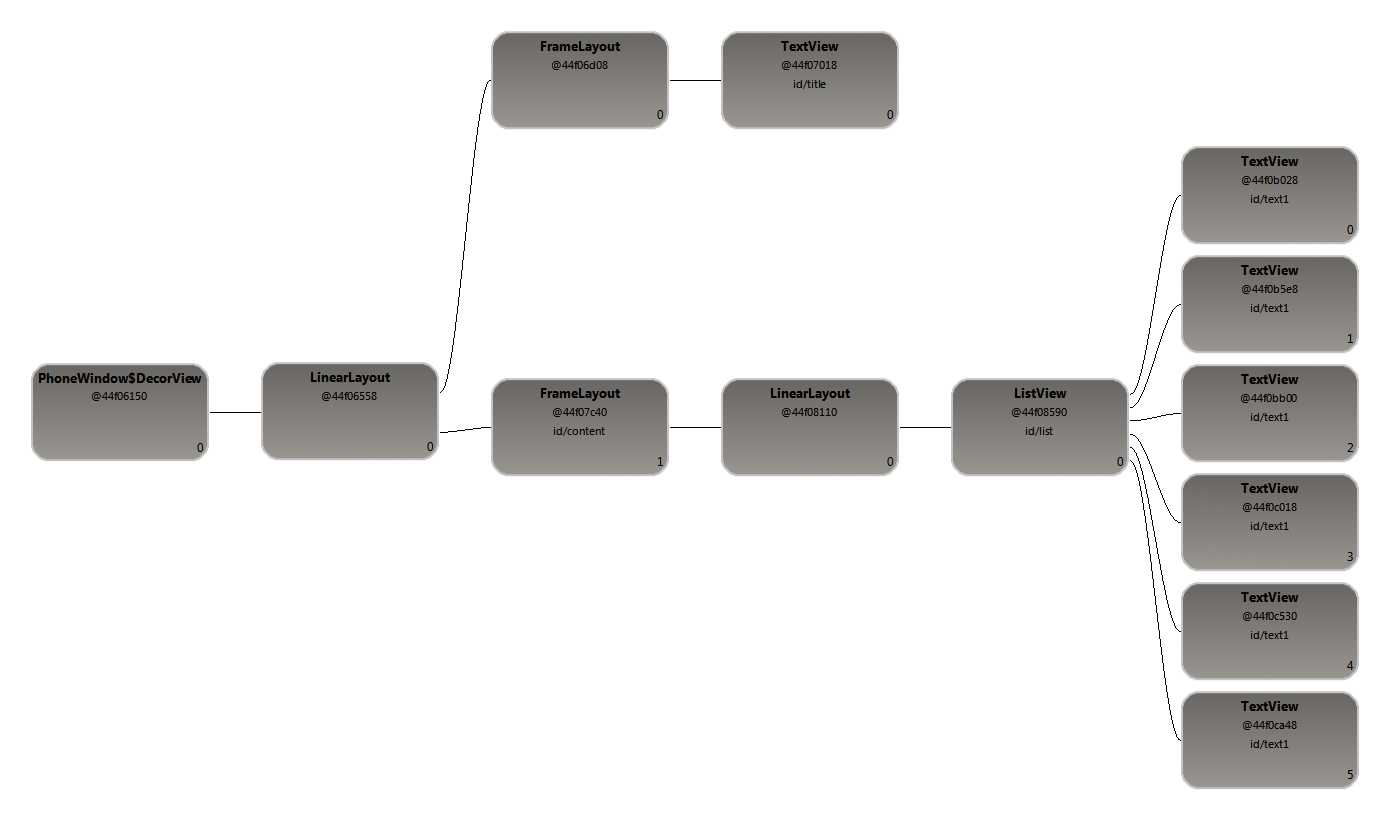
Деятельность poi

Рассмотрим структуру деятельности poi. Внешний вид окна представлен на рисунке 14.



**Рис. 14** Окно poi

Рисунок 15 отображает иерархию разметки окна poi.



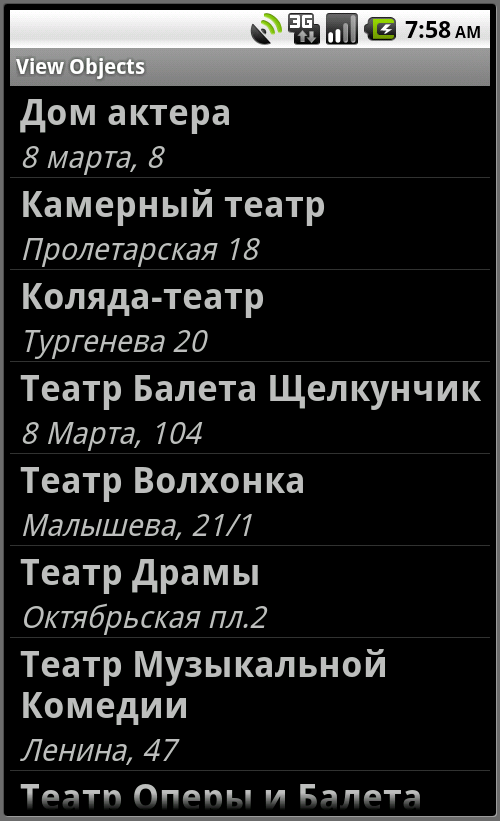
**Рис. 15** Иерархия разметки окна poi

Как видно из рисунка конечными элементами разметки, которые видит пользователь, является список listview содержащий 6 текстовых полей с названиями групп объектов.

При нажатии на какой-либо элемент списка открывается активити, содержащее список всех объектов выбранной группы. Например, при выборе элемента theaters открывается активити theatersviewactivity.

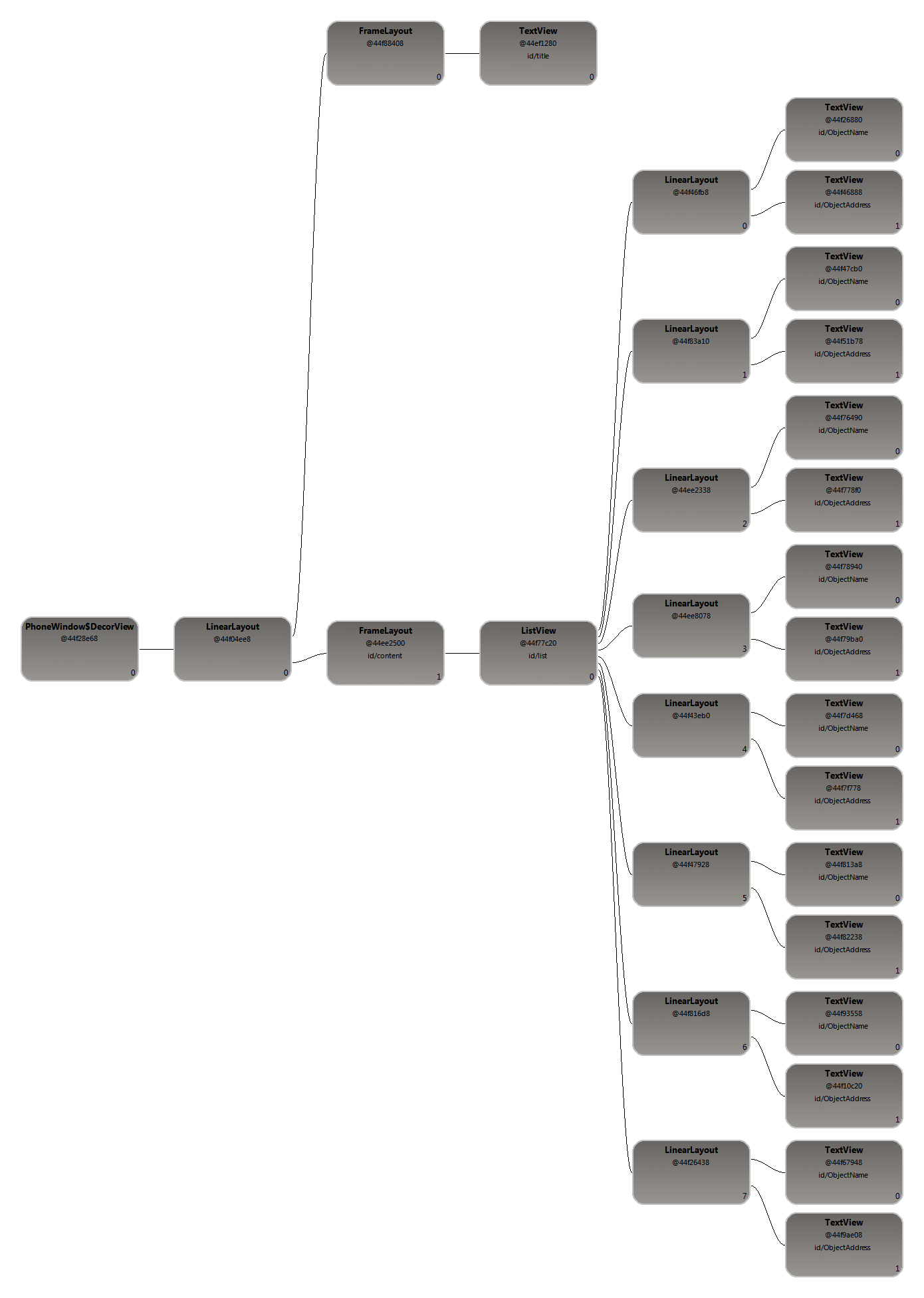
Деятельность TheatersViewActivity

Рассмотрим структуру деятельности TheatersViewActivity. Внешний вид окна представлен на рисунке 16.



**Рис. 16** Окно TheatersViewActivity

Рисунок 17 отображает иерархию разметки окна TheatersViewActivity.



**Рис. 17** Иерархия разметки окна TheatersViewActivity.

Как видно из рисунка конечными элементами разметки, которые видит пользователь, является список listview содержащий текстовые поля с названиями и адресами объектов.

При нажатии на какой-либо элемент списка открывается всплывающее окно с двумя кнопками (add on map, information) (рисунок 18).

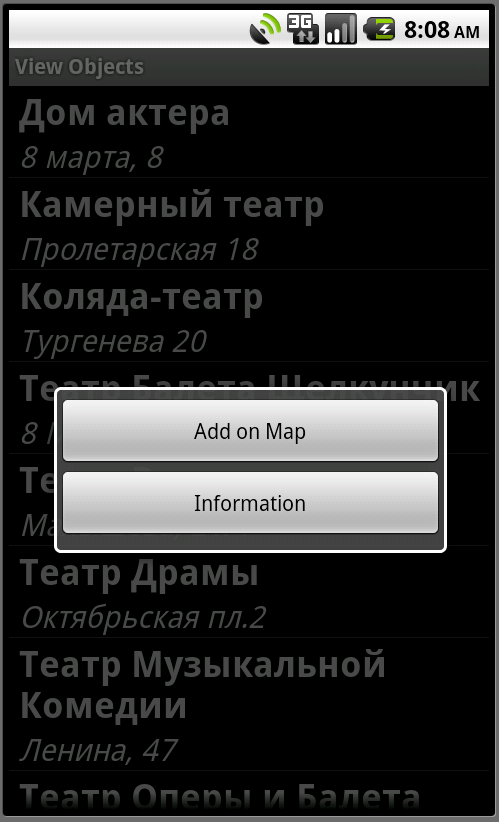
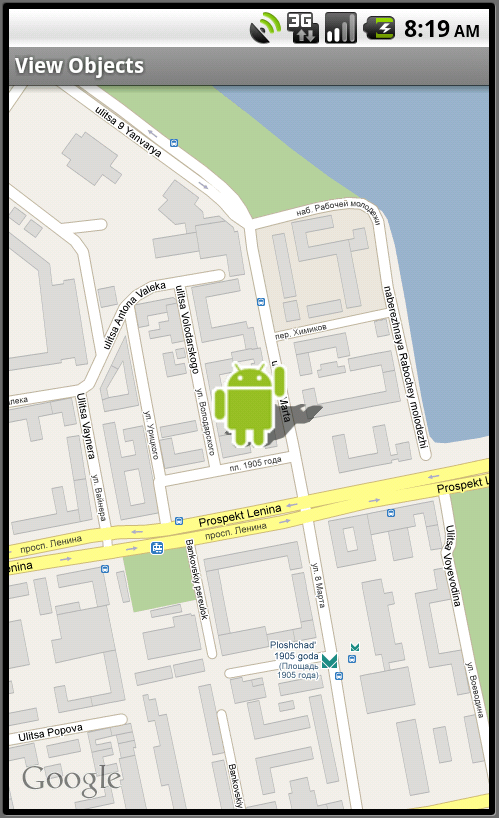
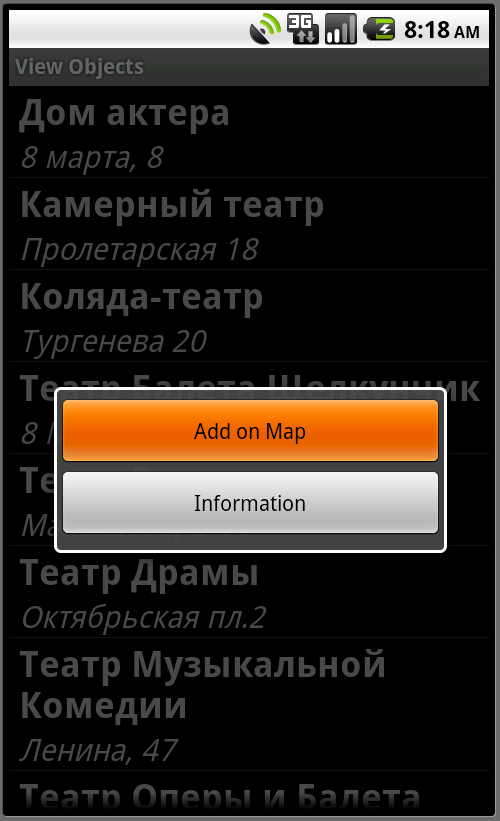
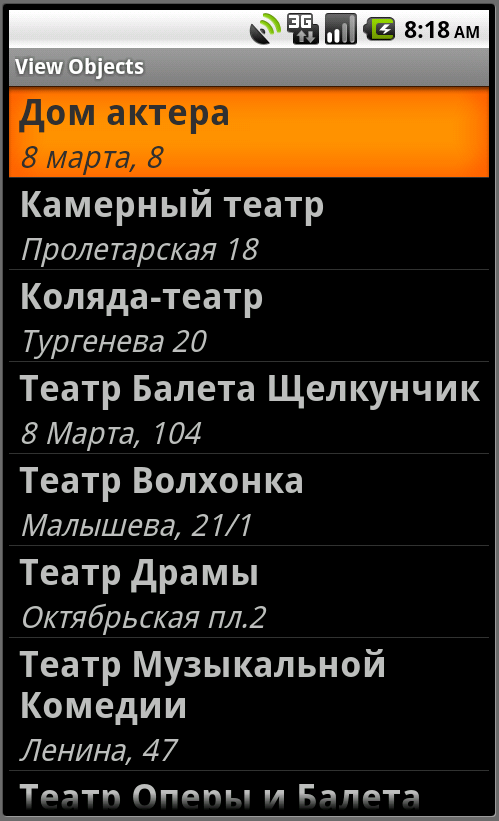


Рис. 18 Всплывающее окно TheatersViewActivity

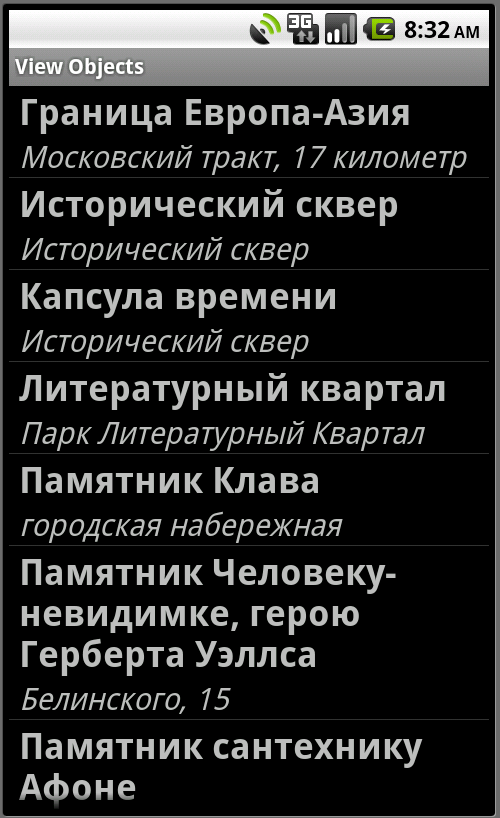
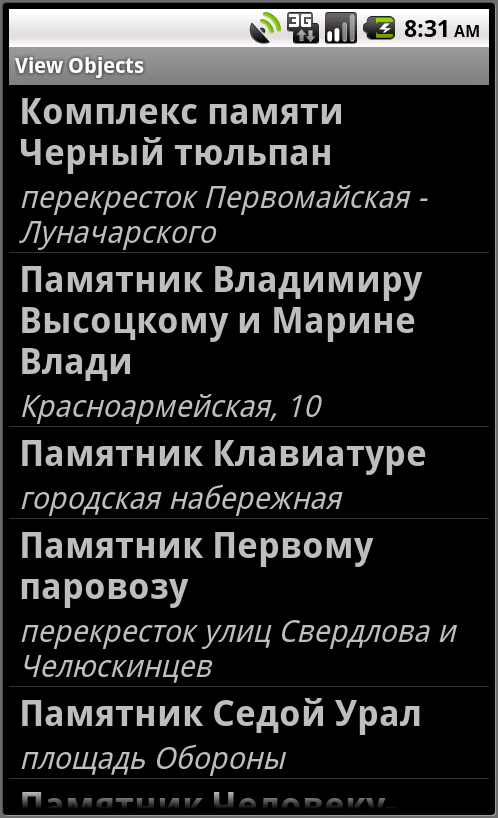
При нажатии на кнопку < add on map > объект отображается на карте (рисунок 19).

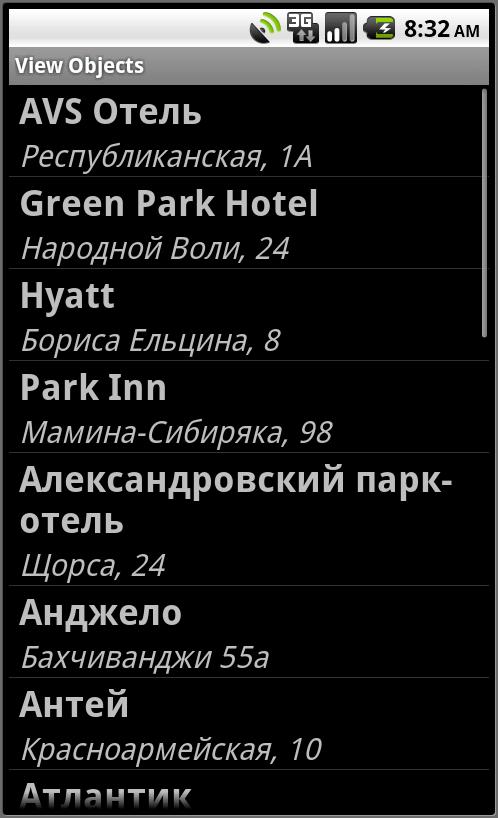
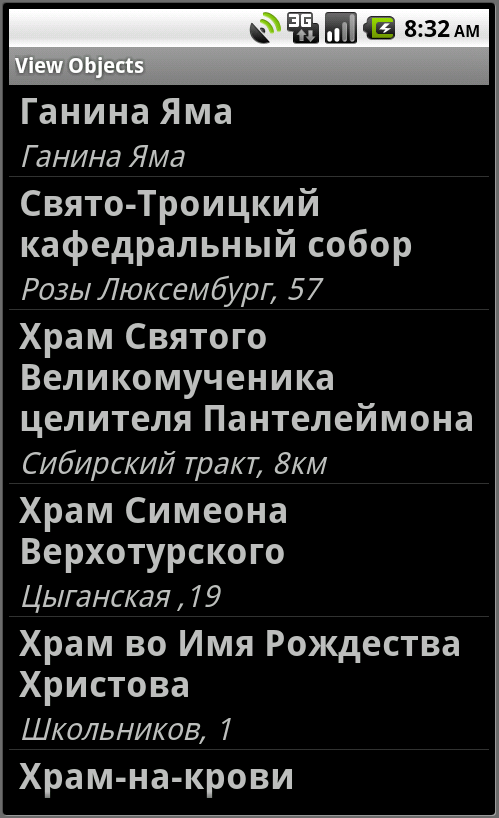


**Рис. 19** Отображение объекта на карте

При нажатии на кнопку <information> открывается всплывающее окно с краткой информацией (название, адрес, время работы, описание объекта)

Деятельности MuseamViewActivity, ArhitectureViewActivity, InterestViewActivity, TemplesViewActivity, HostelViewActivity устроены аналогичным образом (рисунок 20)





**Рис. 20** Деятельности MuseamViewActivity, ArhitectureViewActivity, InterestViewActivity, TemplesViewActivity, HostelViewActivity

## 5.3 Программная реализация.

Компоненты приложения имеют жизненный цикл – начало, когда Android инициализирует их, активный период работы, неактивный период, когда они бездействуют, и конец, когда компоненты уничтожаются и освобождают ресурсы для запуска других компонентов.

Прежде чем Android запустит компонент приложения, он должен узнать, что этот компонент существует. Поэтому приложения объявляют свои компоненты в файле манифеста AndroidManifest.xml, который представляет основную информацию системе.

Структура файла AndroidManifest.xml

Элемент <manifest> является корневым элементом файла.

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.touristmap"* android:versioncode=*"1"* android:versionname=*"1.0."*>

Элемент <uses-sdk> позволяет объявлять совместимость приложения с указанной версией (или более новыми версиями API) платформы Android. Уровень api, объявленный приложением, сравнивается с уровнем API системы мобильного устройства, на который инсталлируется данное приложение.

<uses-sdk android:minsdkversion=*"8"* />

8 версия sdk означает, что приложение будет откомпилировано с использованием библиотек android 2.2.

Элемент <uses-permission> запрашивает разрешения, которые приложению должны быть предоставлены системой для его нормального функционирования.

Использование доступа к интернету:

<uses-permission android:name=*"android.permission.internet"* />

Разрешение приложению доступа к данным о местоположении, предоставляемым сетью wi-fi или сотовой сетью, которые получаются через класс geolocation:

<uses-permission android:name=*"android.permission.access\_coarse\_location"* />

Разрешает приложению доступ к данным gps через класс geolocation.

<uses-permission android:name=*"android.permission.access\_fine\_location"* />

Элемент <application> - это элемент манифеста, содержащий описание компонентов приложения, доступных в пакете. Этот элемент содержит дочерние элементы, которые объявляют каждый из компонентов, входящих в состав приложения.

Элемент <activity> объявляет деятельность. Все деятельности должны быть явно представлены отдельными элементами <activity> в файле манифеста.

Пример задания деятельности:

<activity android:name=*".activities.mapsactivity"*

android:label=*"map"* />

<activity android:name=*".activities.objectseditoractivity"*

android:label=*"objects"*>

</activity>

элемент <provider> объявляет контент-провайдера.

Пример задания контент-провайдера:

<provider android:name=*".providers.objectviewprovider"*

Android:authorities=*"com.touristmap.objectviewprovider"*>

</provider>

Элемент <uses-library> определяет общедоступную библиотеку, с которой должно быть связано приложение.

<uses-library android:name=*"com.google.android.maps"* />

Каталог ресурсов

В данном каталоге хранятся используемые в приложении статические файлы ресурсов: изображения, строки, файлы разметки и т.д.

**Подкаталог res/layout/**

В эту папку помещаются файлы разметки.

Файлы разметки.

Разметка – это архитектура расположения элементов интерфейса пользователя для конкретного окна, представляющего деятельность. Самый общий способ определять разметку и создавать иерархию представления – в xml-файле разметки. Преимущество объявления пользовательского интерфейса в xml-файле состоит в том, что это дает возможность определить представление приложения от программного кода, который управляет поведением приложения. Общая структура xml-файла – это дерево xml элементов, где каждый узел представляет имя класса view.

Файл main.xml

Файл main.xml является файлом разметки для главного активити MainActivity. Корневым элементом данного файла является <tablelayout>

<tablelayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

У данного элемента существует 3 дочерних элемента <button>

<button

android:id=*"@+id/button\_map"*

android:text=*"map"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content*/>

Каждый из этих элементов имеет атрибуты, определяющие свойства данного элемента. Например:

Android:id=*"@+id/button\_map"* – определяет идентификатор объекта, по которому связывается объект из файла разметки с объектом view в коде программы. Символ *+* (*@+id*) означает, что это новое имя ресурса, которое должно быть создано и добавлено к ресурсам в файл r.java, который среда android автоматически генерирует для проекта.

Файлы map.xml, object\_info.xml, object\_menu.xml, object\_view.xml, objects.xml, search\_address.xml, search\_coord.xml, search\_menu.xml, search.xml устроены аналогичным образом.

При запуске деятельности система должна получить ссылку на корневой узел дерева разметки, который будет использоваться для прорисовки графического интерфейса на экране мобильного устройства. Для этого разметка для активити загружается следующим образом

**Public** **void** onCreate(Bundle savedinstancestate) {

**super**.onCreate(savedinstancestate);

SetContentView(R.layout.*main*);

**Подкаталог res/drawable/**

В этом каталоге размещаются все графические файлы, используемые в приложении: иконки, маркеры объектов, кнопки и т.д.

**Подкаталог res/values/**

В данной папке хранятся общие константы для всего приложения: текст используемый элементами управления. Например:

<string name=*"app\_name"*>TouristMap</string>

Файлы деятельностей

Каждый файл деятельности содержит класс, в который добавляется код, реализующий логику работы приложения в данной деятельности. В каждом из таких классов обязательно определяется метод обратного вызова onCreate(), который вызывается системой для прорисовывания окна деятельности на экране устройства.

В файле MainActivity.java задается функциональность главного окна программы: определяются действия, происходящие по нажатию кнопок, а именно запуск активити, соответствующих кнопкам.

**final** Button button\_map = (Button) findViewById (R.id.*button\_map*);

button\_map.setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

Intent mapsActivity = **new** Intent(MainActivity.**this**,MapsActivity.**class**);

startActivity(mapsActivity);

}

});

Также проверяется, включен ли gps приемник на устройстве, если gps выключен, то программа предлагает его включить, т.к. это необходимое условие для работы приложения.

Файл MapsActivity.java описывает функционал для работы с картой:

* Отображение карты (получение контроля над картой, определение вида отображения карты, добавление кнопок масштабирования, загрузка слоев)

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*map*);

myMapView = (MapView) findViewById(R.id.*mapView*);

mapController = myMapView.getController();

myMapView.setStreetView(**true**);

myMapView.setBuiltInZoomControls(**true**);

mapOverlays = myMapView.getOverlays();

mapController.setCenter(*START\_POINT*);

mapController.setZoom(12);

MapOverlay mapOverlay = **new** MapOverlay();

List<Overlay> listOfOverlays = myMapView.getOverlays();

listOfOverlays.clear();

listOfOverlays.add(mapOverlay);

* Создание всплывающего меню в процедуре onCreateOptionsMenu(Menu menu). Создание обработчика событий выбора пункта меню onOptionsItemSelected(MenuItem Item).
* Создание всплывающих диалогов поиска и добавления групп объектов на карту.
* Процедура добавления группы объектов на карту AddGroup(Uri uri, Drawable drawable)
* Процедура добавления точки на карту isAddLocationPoint(**double** netlat, **double** netlng, String name)
* Процедуры вызова диалогов поиска CallSearchAddressDialog(), CallSearchCoordDialog()

В файле ObjectsEditorActivity.java описывается функциональность окна poi:

* Создается список групп объектов
* Создание обработчика событий выбора элементов списка (групп объектов), а именно переходы на новые активити содержащие списки объектов данной группы.

В файле TheatersViewActivity.java описывается функциональность окна theaters:

* Создается список объектов, данные о которых берутся из соответствующей базы данных
* Создание обработчика событий выбора элементов списка
* Процедура добавления объекта на карту CallViewOnMap()
* Процедура отображения диалога с информацией об объекте CallObjectInfoDialog().

файлы MuseamViewActivity.java, ArhitectureViewActivity.java, InterestViewActivity.java, TemplesViewActivity.java, HostelViewActivity.java аналогичным образом определяют структуру соответствующих окон.

Вспомогательные файлы для работы с базой данных

Рассмотрим структуру данных файлов на примере ObjectDbHelper.java. Класс ObjectDbHelper, расширяющий класс SQLiteOpenHelper представляет таблицу object базы данных. В нем объявлена структура таблицы и в методе onCreate() производится создание таблицы и заполнение её текстовыми данными при первом запуске приложения на устройстве.

**public** **void** onCreate(SQLiteDatabase db){

db.execSQL("CREATE TABLE " + *TABLE\_NAME* + " (\_id INTEGER PRIMARY KEY AUTOINCREMENT, "+ *NAME* + " TEXT, " + *ADDRESS* + " TEXT, " + *LAT* + " TEXT, " + *LON* + " TEXT);");

ContentValues values = **new** ContentValues();

values.put(*NAME*, "Театр Эстрады");

values.put(*ADDRESS*, "8 Марта 15");

values.put(*LAT*, "56.835662");

values.put(*LON*, "60.600392");

db.insert(*TABLE\_NAME*, *NAME*, values);

Файлы MuseamDbHelper.java, ArhitectureDbHelper.java, InterestDbHelper.java, TemplesDbHelper.java, HostelDbHelper.java устроены аналогичным образом.

Провайдеры

Рассмотрим структуру провайдера на примере файла класса провайдера ObjectViewProvider.java.

Класс ObjectViewProvider данного файла расширяющий базовый класс ContentProvider представляет логику доступа к содержимому базы данных object.

В методе обратного вызова onCreate() который вызывается системой при создании экземпляра ContentProvider, инициализируется объект SQLiteDatabase.

**private** SQLiteDatabase db;

…

@Override

**public** **boolean** onCreate() {

db = (**new** ObjectDbHelper(getContext())).getWritableDatabase();

**return** (db == **null**) ? **false** : **true**;

}

Метод query(Uri url, String[] projection,

String selection, String[] selectionArgs, String sort)реализует запрос к бд.

Метод insert(Uri url, ContentValues inValues) используется для вставки данных и возвращает клиенту контент-провайдера Uri вставляемой строки.

Метод update(Uri url, ContentValues values,

String where, String[] whereArgs) реализует модификацию данных, а метод delete(Uri url, String where, string[] whereArgs) – удаление данных.

Файлы MuseamProvider.java, InterestProvider.java, ArhitectureProvider.java, TemplesProvider.java, HostelProvider.java устроены аналогичным образом.

# ЗАКЛЮЧЕНИЕ

Подведем итоги нашей работы. Была поставлена задача создания туристической карты для мобильных устройств, работающих на платформе Android. Основной идеей было создание специализированной активной карты с послойной организацией географических и семантических данных на данной карте. Разработан прототип программы, реализующий нашу идею. Программа представляет собой «электронный помощник» туриста, используя который он может познакомиться с достопримечательностями города. На следующем этапе необходимы консультации со специалистами в области туризма и аккуратное опробование на реальных пользователях.

Следует отметить, что данный прототип нуждается в развитии. Перечислим некоторые идеи для дальнейшей доработки нашей системы:

* Создание локальной карты города, с которой можно работать без подключения к интернету.
* Добавление возможности редактирования и сохранения информации о туристических объектах в базе данных.
* Расширение базы данных туристических объектов.
* Расширение файла связи адресов и географических координат.
* Локализация программы на различные языки

# ЛИТЕРАТУРА

1. Ципилева Т.А. - Геоинформационные системы.
2. <http://www.dataplus.ru/> - ООО "ДАТА+" – распространение в России и других странах СНГ передовых геоинформационных технологий ESRI.
3. <http://www.trisoftrus.com/> - Научно-техническая фирма Трисофт
4. <http://www.objectland.ru/> - Геоинформационная система ObjectLand
5. Голощапов А.Л. - Google Android. Программирование для мобильных устройств
6. Reto Meier – Professional Android™ 2 Application development
7. Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike - Android Application Development
8. Chris Haseman – Android Essentials

# ПРИЛОЖЕНИЕ

**Файл AndroidManifest.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.touristmap"* android:versionCode=*"1"* android:versionName=*"1.0."*>

<uses-sdk android:minSdkVersion=*"8"* />

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"*>

<provider android:name=*".providers.ObjectViewProvider"*

android:authorities=*"com.touristmap.ObjectViewProvider"*>

</provider>

<provider android:name=*".providers.MuseamProvider"*

android:authorities=*"com.touristmap.MuseamProvider"*>

</provider>

<provider android:name=*".providers.ArhitectureProvider"*

android:authorities=*"com.touristmap.ArhitectureProvider"*>

</provider>

<provider android:name=*".providers.InterestProvider"*

android:authorities=*"com.touristmap.InterestProvider"*>

</provider>

<provider android:name=*".providers.TemplesProvider"*

android:authorities=*"com.touristmap.TemplesProvider"*>

</provider>

<provider android:name=*".providers.HostelProvider"*

android:authorities=*"com.touristmap.HostelProvider"*>

</provider>

<uses-library android:name=*"com.google.android.maps"* />

<activity android:name=*".activities.MainActivity"*

android:label=*"TouristMap MainMenu"*

android:launchMode=*"singleTask"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*".activities.MapsActivity"*

android:label=*"Map"* />

<activity android:name=*".activities.ObjectsEditorActivity"*

android:label=*"Objects"*>

</activity>

<activity android:name=*".activities.TheatersViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.MuseamViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.ArhitectureViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.InterestViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.TemplesViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.HostelViewActivity"*

android:label=*"@string/title\_view"*>

</activity>

<activity android:name=*".activities.ViewObjectOnMapActivity"*

android:label=*"@string/title\_view"*>

</activity>

</application>

<uses-permission android:name=*"android.permission.INTERNET"* />

<uses-permission android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"* />

<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"* />

</manifest>

**Файл MainActivity.java**

package com.touristmap.activities;

import com.touristmap.R;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.location.LocationManager;

import android.os.Bundle;

import android.provider.Settings;

import android.view.View;

import android.widget.Button;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

LocationManager locationManager = (LocationManager)getSystemService(Context.LOCATION\_SERVICE);

if (!locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER)) {

final AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("GPS выключен")

.setCancelable(false)

.setPositiveButton("Настройки GPS", new DialogInterface.OnClickListener() {

public void onClick(final DialogInterface dialog, final int id) {

Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

startActivityForResult(intent, 1);

}

})

.setNegativeButton("Отмена", new DialogInterface.OnClickListener() {

public void onClick(final DialogInterface dialog, final int id) {

finish();

}

});

final AlertDialog alert = builder.create();

alert.show();

}

final Button button\_map = (Button) findViewById (R.id.button\_map);

button\_map.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent mapsActivity = new Intent(MainActivity.this,MapsActivity.class);

startActivity(mapsActivity);

}

});

final Button button\_settings = (Button) findViewById (R.id.button\_poi);

button\_settings.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent objectsEditorActivity = new Intent(MainActivity.this,ObjectsEditorActivity.class);

startActivity(objectsEditorActivity);

}

});

final Button button\_exit = (Button) findViewById (R.id.button\_exit);

button\_exit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

finish();

}

});

}

}

**Файл map.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<com.google.android.maps.MapView

android:id=*"@+id/mapView"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:enabled=*"true"*

android:clickable=*"true"*

android:apiKey=*"0GJi6BBZP0bHZdweHL03Lq1vTlpaVxMqpoZrqMQ"*

/>

</LinearLayout>

**Файл MapsActivity.java**

package com.touristmap.activities;

import android.net.Uri;

import android.os.Bundle;

import com.touristmap.DirectionPathOverlay;

import com.touristmap.HelloItemizedOverlay;

import com.touristmap.MyPositionOverlay;

import com.touristmap.ObjectDbHelper;

import com.touristmap.R;

import com.google.android.maps.GeoPoint;

import com.google.android.maps.MapActivity;

import com.google.android.maps.MapController;

import com.google.android.maps.MapView;

import com.google.android.maps.Overlay;

import com.google.android.maps.OverlayItem;

import com.touristmap.activities.MapsActivity;

import com.touristmap.providers.ArhitectureProvider;

import com.touristmap.providers.HostelProvider;

import com.touristmap.providers.InterestProvider;

import com.touristmap.providers.MuseamProvider;

import com.touristmap.providers.ObjectViewProvider;

import com.touristmap.providers.TemplesProvider;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.database.Cursor;

import android.graphics.drawable.Drawable;

import android.location.Address;

import android.location.Criteria;

import android.location.Geocoder;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.MotionEvent;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.util.\*;

public class MapsActivity extends MapActivity {

MapController mapController;

MyPositionOverlay positionOverlay;

private String[] street;

private int[] house;

private double[] lath;

private double[] lonh;

private int sizeSTR=-1;

private String streetIsc="ленина";

private int houseIsc = 47;

private double latIsc=0;

private double lonIsc=0;

public static final int IDM\_SETTINGS = 101;

public static final int IDM\_ADD\_OBJ = 102;

public static final int IDM\_ADD\_ROUTE = 103;

public static final int IDM\_PATH\_TEST = 104;

public static final int IDM\_SEARCH = 105;

public static final int IDM\_EXIT = 106;

private final static int IDD\_OBJECTS = 0;

private final static int IDD\_ADD\_TOURIST\_ROUTE = 1;

private final static int IDD\_COLLECT\_ROUTE=2;

final CharSequence[] mObjects = { "Theaters", "Museams", "Monuments", "Interest Objects", "Temples", "Hostels" };

final boolean[] mCheckedObjItems = { false, false, false, false, false, false };

final boolean[] mCheckedPathItems = { false, false, false, false, false, false };

final CharSequence[] mRouteObj = { "Памятник первому паравозу", "гостиница Свердловск", "Усадьба Расторгуева-Харитонова", "Храм-на-крови", "Отель Вознесенский", "Галерея Эгида", "Коляда-театр", "Площадь Советской Армии", "Театр Музыкальной Комедии", "Гостиница Исеть", "Памятник “Первая конная”" };

final boolean[] mCheckedRouteObjItems = { false, false, false, false, false, false, false, false, false, false, false};

private static final String[] mCoord = new String[] { ObjectDbHelper.\_ID,

ObjectDbHelper.NAME, ObjectDbHelper.LAT, ObjectDbHelper.LON };

public static String NLAT = "nlat";

public static String NLNG = "nlng";

List<Overlay> mapOverlays;

HelloItemizedOverlay itemizedOverlay;

MapView myMapView;

double lat;

double lng;

private double[] latiC;

private double[] longC;

private String[] nameC;

Drawable theatres;

Drawable museams;

Drawable monuments;

Drawable interestObjects;

Drawable temples;

Drawable hostels;

String path = "/mnt/sdcard/OutCoord.txt";

String coord = "/mnt/sdcard/coord.txt";

String in = "/mnt/sdcard/in.txt";

String out = "/mnt/sdcard/out.txt";

String out\_coord = "/mnt/sdcard/OutCoord.txt";

private int[][] A;

private int s, t;

private int size = -1;

int[] d;

private Stack<Integer> st;

private Stack<Integer> st1;

private int stackSize = 0;

public LinkedList<Integer> l = new LinkedList<Integer>();

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.map);

myMapView = (MapView) findViewById(R.id.mapView);

mapController = myMapView.getController();

myMapView.setStreetView(true);

myMapView.setBuiltInZoomControls(true);

mapOverlays = myMapView.getOverlays();

mapController.setCenter(START\_POINT);

mapController.setZoom(12);

// Добавляем маркер своего местоположения

positionOverlay = new MyPositionOverlay();

List<Overlay> overlays = myMapView.getOverlays();

overlays.add(positionOverlay);

theatres = getResources().getDrawable(R.drawable.h\_theatre);

museams = getResources().getDrawable(R.drawable.h\_museum);

monuments = getResources().getDrawable(R.drawable.h\_memorial);

interestObjects = getResources().getDrawable(R.drawable.h\_view\_point);

temples = getResources().getDrawable(R.drawable.h\_shelter);

hostels = getResources().getDrawable(R.drawable.h\_hotel);

LocationManager locationManager;

String context = Context.LOCATION\_SERVICE;

locationManager = (LocationManager) getSystemService(context);

Criteria criteria = new Criteria();

criteria.setAccuracy(Criteria.ACCURACY\_FINE);

criteria.setAltitudeRequired(false);

criteria.setBearingRequired(false);

criteria.setCostAllowed(true);

criteria.setPowerRequirement(Criteria.POWER\_LOW);

String provider = locationManager.getBestProvider(criteria, true);

Location location = locationManager.getLastKnownLocation(provider);

updateWithNewLocation(location);

locationManager.requestLocationUpdates(provider, 1500, 3,

locationListener);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

menu.add(Menu.NONE, IDM\_SETTINGS, Menu.NONE, "POI")

.setAlphabeticShortcut('p');

menu.add(Menu.NONE, IDM\_ADD\_OBJ, Menu.NONE, "Add objects")

.setAlphabeticShortcut('a');

menu.add(Menu.NONE, IDM\_ADD\_ROUTE, Menu.NONE, "Route")

.setAlphabeticShortcut('r');

menu.add(Menu.NONE, IDM\_SEARCH, Menu.NONE, "Search")

.setAlphabeticShortcut('s');

menu.add(Menu.NONE, IDM\_EXIT, Menu.NONE, "Exit").setAlphabeticShortcut(

'x');

return (super.onCreateOptionsMenu(menu));

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case IDM\_SETTINGS:

Intent objectsEditorActivity = new Intent(MapsActivity.this, ObjectsEditorActivity.class);

startActivity(objectsEditorActivity);

item.setChecked(true);

break;

case IDM\_ADD\_OBJ:

showDialog(IDD\_OBJECTS);

item.setChecked(true);

break;

case IDM\_SEARCH:

LayoutInflater inflater = LayoutInflater.from(this);

View root = inflater.inflate(R.layout.search\_menu2, null);

final Button button\_addr = (Button) root.findViewById(R.id.button\_addr);

final Button button\_coord = (Button) root.findViewById(R.id.button\_coord);

button\_addr.setText("Search by address");

button\_coord.setText("Search by coordinates");

AlertDialog.Builder menu = new AlertDialog.Builder(this);

menu.setView(root);

menu.show();

button\_addr.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

CallSearchAddressDialog();

};

});

button\_coord.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

CallSearchCoordDialog();

};

});

item.setChecked(true);

break;

case IDM\_EXIT:

System.exit(0);

break;

default:

return false;

}

return true;

}

// Меню с чекбоксами для добавления групп объектов на карту

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_OBJECTS:

AlertDialog.Builder builderObj = new AlertDialog.Builder(this);

builderObj.setTitle("Add group objects");

builderObj.setMultiChoiceItems(mObjects, mCheckedObjItems, new DialogInterface.OnMultiChoiceClickListener() {

@Override

public void onClick(DialogInterface dialog, int which, boolean isChecked) {

mCheckedObjItems[which] = isChecked;

}

});

builderObj.setPositiveButton("OK", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

if (!mapOverlays.isEmpty()) {

mapOverlays.clear();

} else {

if (mCheckedObjItems[0]) {

addGroup(ObjectViewProvider.CONTENT\_URI, theatres);

}

if (mCheckedObjItems[1]) {

addGroup(MuseamProvider.CONTENT\_URI, museams);

}

if (mCheckedObjItems[2]) {

addGroup(ArhitectureProvider.CONTENT\_URI, monuments);

}

if (mCheckedObjItems[3]) {

addGroup(InterestProvider.CONTENT\_URI, interestObjects);

}

if (mCheckedObjItems[4]) {

addGroup(TemplesProvider.CONTENT\_URI, temples);

}

if (mCheckedObjItems[5]) {

addGroup(HostelProvider.CONTENT\_URI, hostels);

}

}

}

});

builderObj.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

Toast.makeText(getApplicationContext(), "Dialog cancel", Toast.LENGTH\_SHORT).show();

}

});

builderObj.setCancelable(false);

return builderObj.create();

// Процедура Добавления группы объектов на карту

private void addGroup(Uri uri, Drawable drawable) {

Cursor cursor = managedQuery(uri, mCoord, "\_ID", null, null);

int count = cursor.getCount();

latiC = new double[count];

longC = new double[count];

nameC = new String[count];

for (int i = 0; i < count; i++) {

latiC[i] = -1;

longC[i] = -1;

nameC[i] = "";

}

for (int i = 0; i < count; i++) {

cursor.moveToPosition(i);

lat = Double.parseDouble(cursor.getString(2));

lng = Double.parseDouble(cursor.getString(3));

latiC[i] = lat;

longC[i] = lng;

nameC[i] = cursor.getString(1);

}

double nl, nlg;

itemizedOverlay = new HelloItemizedOverlay(this, drawable);

try {

for (int i = 0; i < count; i++) {

nl = latiC[i];

nlg = longC[i];

String name = nameC[i];

isAddLocationPoint(nl, nlg, name);

}

} catch (NullPointerException e) {

e.printStackTrace();

}

}

// Процедура добавления точки на карту

public void isAddLocationPoint(double netlat, double netlng, String name) {

GeoPoint point = new GeoPoint((int) (netlat \* 1E6),

(int) (netlng \* 1E6));

String coord = "Широта: " + netlat + "\nДолгота: " + netlng;

// Создаем точку и текст появляющийся при нажатии

OverlayItem overlayitem = new OverlayItem(point, name, coord);

// Положение карты при запуске в точке point

mapController.animateTo(point);

mapController.setZoom(13);

// Добавляем точку на карту

itemizedOverlay.addOverlay(overlayitem);

mapOverlays.add(itemizedOverlay);

}

//Процедура поиска по адресу

private void CallSearchAddressDialog() {

LayoutInflater inflater = LayoutInflater.from(this);

View root = inflater.inflate(R.layout.search\_address, null);

final EditText textStreet = (EditText)root.findViewById(R.id.street);

final EditText textHouse = (EditText)root.findViewById(R.id.house);

AlertDialog.Builder b = new AlertDialog.Builder(this);

b.setView(root);

b.setTitle("Search by address");

b.setPositiveButton(

"Ок", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

String street = textStreet.getText().toString();

int house = Integer.parseInt(textHouse.getText().toString());

ReadInTxtSearch();

searchadress(street, house);

if (latIsc!=0 & lonIsc !=0 ){

String text = "Широта: " + latIsc + "\nДолгота: "

+ lonIsc;

Toast.makeText(getBaseContext(), text, Toast.LENGTH\_SHORT).show();

itemizedOverlay = new HelloItemizedOverlay(null, getResources().getDrawable(R.drawable.androidmarker));

isAddLocationPoint(latIsc, lonIsc, "Искомая точка");

}

else {

String ErrMsg = "Адрес не найден!";

Toast.makeText(getBaseContext(), ErrMsg, Toast.LENGTH\_SHORT).show();

}

}

});

b.setNegativeButton(

"Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {}

});

b.show();

}

//Процедура поиска по координатам

private void CallSearchCoordDialog() {

LayoutInflater inflater = LayoutInflater.from(this);

View root = inflater.inflate(R.layout.search\_coord, null);

final EditText textLatitude = (EditText)root.findViewById(R.id.latitude);

final EditText textLongitude = (EditText)root.findViewById(R.id.longitude);

AlertDialog.Builder b = new AlertDialog.Builder(this);

b.setView(root);

b.setTitle("Search by coordinates");

b.setPositiveButton(

"Ок", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

String sslat = textLatitude.getText().toString();

String sslon = textLongitude.getText().toString();

double slat = Double.parseDouble(sslat);

double slon = Double.parseDouble(sslon);

Drawable SearchCoord = getResources().getDrawable(R.drawable.androidmarker);

itemizedOverlay = new HelloItemizedOverlay( null, SearchCoord);

isAddLocationPoint (slat,slon,"искомая точка");

}

});

b.setNegativeButton(

"Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {}

});

b.show();

}

//Процедура считывания данных для поиска

private void ReadInTxtSearch()

{

try

{

BufferedReader ibr = new BufferedReader(

new InputStreamReader(

new FileInputStream("/mnt/sdcard/insearch.txt"),"windows-1251"));

String str;

Scanner scanner;

String[] vr;

sizeSTR=Integer.parseInt(ibr.readLine());

street = new String [sizeSTR];

house = new int [sizeSTR];

lath = new double [sizeSTR];

lonh = new double [sizeSTR];

vr = new String [4];

int a;

for(int i = 0; i<sizeSTR; i++)

{

lath[i]=-1000;

lonh[i]=-1000;

street[i]=" ";

house[i]=-1;

}

for(int i = 0; i<sizeSTR; i++)

{

for (int j=0;j<4;j++)

{

vr[j]="";

}

a=0;

str=ibr.readLine();

scanner = new Scanner(str);

while (scanner.hasNext())

{

vr[a]=scanner.next();

a=a+1;

}

street[i]=vr[0];

house[i]=Integer.parseInt(vr[1]);

lath[i]=Double.parseDouble(vr[2]);

lonh[i]=Double.parseDouble(vr[3]);

}

ibr.close();

}

catch(IOException ae){}

};

private void searchadress(String streetIsc, int houseIsc)

{

streetIsc = streetIsc.toLowerCase();

int fl=0, ch=0;

while ((fl==0) & (ch<sizeSTR))

{

if ((streetIsc.equals(street[ch])) & (houseIsc == house[ch]))

{

latIsc = lath[ch];

lonIsc = lonh[ch];

fl=1;

}

ch=ch+1;

}

}}

**Файл ObjectsEditorActivity.java**

package com.touristmap.activities;

import com.touristmap.R;

import android.os.Bundle;

import android.app.ListActivity;

import android.content.Intent;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

public class ObjectsEditorActivity extends ListActivity {

/\*\* Called when the activity is first created. \*/

private int mId = -1;

String[] mObjectsClass = { "Theaters", "Museams", "Monuments",

"Interest Objects", "Temples", "Hostels" };

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.objects);

setListAdapter(new ArrayAdapter<String>(this,

android.R.layout.simple\_list\_item\_1, mObjectsClass));

}

public void onListItemClick(ListView parent, View v, int position, long id) {

mId = position;

switch (mId) {

case 0: {

Intent intent = new Intent();

intent.setClass(this, TheatersViewActivity.class);

startActivity(intent);

}

break;

case 1: {

Intent intent = new Intent();

intent.setClass(this, MuseamViewActivity.class);

startActivity(intent);

}

break;

case 2: {

Intent intent = new Intent();

intent.setClass(this, ArhitectureViewActivity.class);

startActivity(intent);

}

break;

case 3: {

Intent intent = new Intent();

intent.setClass(this, InterestViewActivity.class);

startActivity(intent);

}

break;

case 4: {

Intent intent = new Intent();

intent.setClass(this, TemplesViewActivity.class);

startActivity(intent);

}

break;

case 5: {

Intent intent = new Intent();

intent.setClass(this, HostelViewActivity.class);

startActivity(intent);

}

break;

}

}

}

**Файл TheatersViewActivity.java**

package com.touristmap.activities;

import com.touristmap.ObjectDbHelper;

import com.touristmap.R;

import com.touristmap.providers.ObjectViewProvider;

import android.os.Bundle;

import android.app.AlertDialog;

import android.app.ListActivity;

import android.database.Cursor;

import android.view.LayoutInflater;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.ListAdapter;

import android.widget.SimpleCursorAdapter;

import android.widget.Toast;

import android.content.Intent;

public class TheatersViewActivity extends ListActivity {

private Cursor mCursor;

private ListAdapter mAdapter;

private String netlat;

private String netlng;

double lat;

double lng;

private static final String[] *mContent* = new String[] { ObjectDbHelper.*\_ID*,

ObjectDbHelper.*NAME*, ObjectDbHelper.*ADDRESS* };

private static final String[] *displayContent* = new String[] {

ObjectDbHelper.*NAME*, ObjectDbHelper.*ADDRESS* };

int[] displayViews = new int[] { R.id.*ObjectName*, R.id.*ObjectAddress* };

private int infId;

private static final String[] *mCoord* = new String[] { ObjectDbHelper.*\_ID*,

ObjectDbHelper.*LAT*, ObjectDbHelper.*LON* };

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mCursor = managedQuery(ObjectViewProvider.*CONTENT\_URI*, *mContent*, null,

null, null);

mAdapter = new SimpleCursorAdapter(this, R.layout.*object\_view*, mCursor,

*displayContent*, displayViews);

setListAdapter(mAdapter);

}

public void onListItemClick(ListView parent, View v, int position, long id) {

infId = position;

Cursor cursor = managedQuery(ObjectViewProvider.*CONTENT\_URI*, *mCoord*,

"\_ID", null, null);

cursor.moveToPosition(position);

String text = "Широта: " + cursor.getString(1) + "\nДолгота: "

+ cursor.getString(2);

Toast.*makeText*(this, text, Toast.*LENGTH\_SHORT*).show();

netlat = cursor.getString(1);

netlng = cursor.getString(2);

LayoutInflater inflater = LayoutInflater.*from*(this);

View root = inflater.inflate(R.layout.*object\_menu*, null);

final Button button\_add = (Button) root.findViewById(R.id.*button\_add*);

final Button button\_info = (Button) root.findViewById(R.id.*button\_info*);

final Button button\_route = (Button) root.findViewById(R.id.*button\_route*);

button\_add.setText("Add on Map");

button\_info.setText("Information");

button\_route.setText("Construct a route");

AlertDialog.Builder menu = new AlertDialog.Builder(this);

menu.setView(root);

menu.show();

button\_info.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

CallObjectInfoDialog();

};

});

button\_add.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

CallViewOnMap();

};

});

button\_route.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

};

});

}

private void CallViewOnMap(){

Intent intent = new Intent();

intent.setClass(this, ViewObjectOnMapActivity.class);

intent.putExtra(MapsActivity.*NLAT*, netlat);

intent.putExtra(MapsActivity.*NLNG*, netlng);

startActivity(intent);

finish();

}

private void CallObjectInfoDialog() {

LayoutInflater inflater = LayoutInflater.*from*(this);

View root = inflater.inflate(R.layout.*object\_info*, null);

final TextView textName = (TextView) root.findViewById(R.id.*name*);

final TextView textPhone = (TextView) root.findViewById(R.id.*address*);

mCursor.moveToPosition(infId);

textName.setText(mCursor.getString(1));

textPhone.setText(mCursor.getString(2));

AlertDialog.Builder b = new AlertDialog.Builder(this);

b.setView(root);

b.show();

}

}

**Файл ObjectViewProvider.java**

package com.touristmap.providers;

import java.util.HashMap;

import com.touristmap.ObjectDbHelper;

import android.content.ContentProvider;

import android.content.ContentUris;

import android.content.ContentValues;

import android.content.UriMatcher;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.net.Uri;

import android.text.TextUtils;

public class ObjectViewProvider extends ContentProvider {

public static final String DB\_OBJECTS = "objects.db";

public static final Uri CONTENT\_URI = Uri.parse("content://com.touristmap.ObjectViewProvider/object");

public static final int URI\_CODE = 1;

public static final int URI\_CODE\_ID = 2;

private static final UriMatcher mUriMatcher;

private static HashMap<String, String> mObjectMap;

private SQLiteDatabase db;

static {

mUriMatcher = new UriMatcher(UriMatcher.NO\_MATCH);

mUriMatcher.addURI("com.touristmap.ObjectViewProvider",

ObjectDbHelper.TABLE\_NAME, URI\_CODE);

mUriMatcher.addURI("com.touristmap.ObjectViewProvider",

ObjectDbHelper.TABLE\_NAME + "/#", URI\_CODE\_ID);

mObjectMap = new HashMap<String, String>();

mObjectMap.put(ObjectDbHelper.\_ID, ObjectDbHelper.\_ID);

mObjectMap.put(ObjectDbHelper.NAME, ObjectDbHelper.NAME);

mObjectMap.put(ObjectDbHelper.ADDRESS, ObjectDbHelper.ADDRESS);

mObjectMap.put(ObjectDbHelper.LAT, ObjectDbHelper.LAT);

mObjectMap.put(ObjectDbHelper.LON, ObjectDbHelper.LON);

}

public String getDbName() {

return(DB\_OBJECTS);

}

@Override

public boolean onCreate() {

db = (new ObjectDbHelper(getContext())).getWritableDatabase();

return (db == null) ? false : true;

}

@Override

public Cursor query(Uri url, String[] projection, String selection, String[] selectionArgs, String sort) {

String orderBy;

if (TextUtils.isEmpty(sort)) {

orderBy = ObjectDbHelper.NAME;

}

else {

orderBy = sort;

}

Cursor c = db.query(ObjectDbHelper.TABLE\_NAME, projection, selection, selectionArgs, null, null, orderBy);

c.setNotificationUri(getContext().getContentResolver(), url);

return c;

}

@Override

public Uri insert(Uri url, ContentValues inValues) {

ContentValues values = new ContentValues(inValues);

long rowId = db.insert(ObjectDbHelper.TABLE\_NAME, ObjectDbHelper.NAME, values);

if (rowId > 0) {

Uri uri = ContentUris.withAppendedId(CONTENT\_URI, rowId);

getContext().getContentResolver().notifyChange(uri, null);

return uri;

}

else {

throw new SQLException("Failed to insert row into " + url);

}

}

@Override

public int delete(Uri url, String where, String[] whereArgs) {

int retVal = db.delete(ObjectDbHelper.TABLE\_NAME, where, whereArgs);

getContext().getContentResolver().notifyChange(url, null);

return retVal;

}

@Override

public int update(Uri url, ContentValues values,

String where, String[] whereArgs) {

int retVal = db.update(ObjectDbHelper.TABLE\_NAME, values, where, whereArgs);

getContext().getContentResolver().notifyChange(url, null);

return retVal;

}

@Override

public String getType(Uri uri) {

return null;

}

}

**Файл ObjectDbHelper.java**

package com.touristmap;

//музеи

import com.touristmap.providers.ObjectViewProvider;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.provider.BaseColumns;

public class ObjectDbHelper extends SQLiteOpenHelper implements BaseColumns{

public static final String TABLE\_NAME="object";

public static final

LAT="latitude";

public static final String NAME="name";

public static final String ADDRESS="address";

public static final String LON="longitude";

private SQLiteDatabase db;

private final Context mCtx;

private ObjectDbHelper mDbHelper;

@Override

public void onCreate(SQLiteDatabase db){

db.execSQL("CREATE TABLE " + TABLE\_NAME + " (\_id INTEGER PRIMARY KEY AUTOINCREMENT, "+ NAME + " TEXT, " + ADDRESS + " TEXT, " + LAT + " TEXT, " + LON + " TEXT);");

ContentValues values = new ContentValues();

values.put(NAME, "Театр Эстрады");

values.put(ADDRESS, "8 Марта 15");

values.put(LAT, "56.835662");

values.put(LON, "60.600392");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Коляда-театр");

values.put(ADDRESS, "Тургенева 20");

values.put(LAT, "56.843166");

values.put(LON, "60.6141");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театрон");

values.put(ADDRESS, " Ленина, 52/4");

values.put(LAT, "56.840243");

values.put(LON, "60.625806");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Шарманка");

values.put(ADDRESS, "Старых Большевиков, 22 (ДК Эльмаш)");

values.put(LAT, " 56.892666");

values.put(LON, "60.630371");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Драмы");

values.put(ADDRESS, "Октябрьская пл.2");

values.put(LAT, "56.84356");

values.put(LON, "60.593352");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Балета Щелкунчик");

values.put(ADDRESS, "8 Марта, 104");

values.put(LAT, "56.81438");

values.put(LON, "60.607221");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Оперы и Балета");

values.put(ADDRESS, "Ленина, 46а ");

values.put(LAT, "56.838897");

values.put(LON, "60.616672");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Волхонка");

values.put(ADDRESS, "Малышева, 21/1");

values.put(LAT, "56.83337");

values.put(LON, "60.591834");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Камерный театр");

values.put(ADDRESS, "Пролетарская 18");

values.put(LAT, "56.843585");

values.put(LON, "60.607145");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Театр Музыкальной Комедии");

values.put(ADDRESS, "Ленина, 47");

values.put(LAT, "56.839809");

values.put(LON, "60.61336");

db.insert(TABLE\_NAME, NAME, values);

values.put(NAME, "Дом актера");

values.put(ADDRESS, "8 марта, 8");

values.put(LAT, "56.838589");

values.put(LON, "60.59797");

db.insert(TABLE\_NAME, NAME, values);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

public ObjectDbHelper (Context ctx){

super(ctx, ObjectViewProvider.DB\_OBJECTS, null, 1);

this.mCtx = ctx;

}

public ObjectDbHelper open() throws SQLException {

mDbHelper = new ObjectDbHelper(mCtx);

db = mDbHelper.getWritableDatabase();

return this;

}

public void close() {

mDbHelper.close();

}

//Поиск запросом LIKE

public Cursor fetchRecordsByQuery(String query) {

return db.query(true, TABLE\_NAME, new String[] { \_ID,

NAME }, NAME + " LIKE" + "'%" + query + "%'", null,

null, null, null, null);

}

}