Департамент образования города Москвы

Государственное бюджетное общеобразовательное учреждение

города Москвы

«Гимназия № 1505 «Московская городская педагогическая гимназия-лаборатория»»

**ДИПЛОМНОЕ ИССЛЕДОВАНИЕ**

на тему:

**Друзья всегда рядом**

Выполнил:

Мухортов Виктор, 10В

Руководитель

Мокин Евгений Анатольевич

Москва

2016/2017 уч.г.

**Оглавление**

1. Титульный лист……………………………………………………1

2. Введение……………………………………………………………3

3. Глава 1. Структура приложения Android…………………………4

4. Глава 2. Мое приложение..………………………………………..12

5. Заключение………………………………………………………...13

6. Приложение………………………………………………………..14

7. Список литературы………………………………………………..17

**Введение**

Приложения для операционной системы Android всегда были популярны, в последнее время выходит достаточно много интересных приложений, к примеру, определяющих кто находится рядом или для определения названия трека, по звучанию.

В данной работе я собираюсь изучить принципы работы приложения, определяющего местоположение пользователя через GPS и рассчитывающее примерное расстояние до гимназии 1505. Для создания такого приложения мне нужно изучить и определить, на какой базе будет создано приложение, научиться: работать с гео-датчиком, рассчитывать расстояние между точками в пространстве, создавать приложения для Android.

Структура работы. Диплом состоит из введения, двух глав, заключения, и списка литературы. В первой главе описывается, что такое приложение Android и его структура. Во второй главе описывается работа и функциональность моего приложения. В заключении, что в моем приложении получилось и не получилось.

**Глава 1.Структура приложения Android**

Приложения для Android пишутся на языке программирования Java. Инструменты Android SDK (Software Development Kit – комплект разработки программного обеспечения) компилируют написанный код приложения — и все требуемые файлы данных и ресурсов — в файл APK – *программный пакет Android* - файл архива с расширением .apk. В файле APK находится все, что нужно для работы Android-приложения, и позволяет установить приложение на любое Android-устройство.

Каждое приложение Android, установленное на устройстве, работает в собственной изолированной программной среде:

* операционная система Android является многопользовательской системой Linux, где каждое приложение является отдельным пользователем;
* каждому приложению система дает уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению); система устанавливает полномочия для всех файлов в приложении, чтобы доступ к ним был разрешен только приложению с нужным идентификатором;
* у каждого процесса имеется собственная виртуальная машина, так что код приложения выполняется изолированно от других приложений;
* по умолчанию каждое приложение выполняется в собственном процессе Linux. Android запускает процесс, когда требуется выполнить какой-либо компонент приложения, а затем завершает процесс, когда он больше не нужен либо когда системе требуется освободить память для других приложений.

Каждое приложение по умолчанию имеет доступ только к тем компонентам, которые ему необходимы для работы, и ни к каким другим. Благодаря этому формируется исключительно безопасная среда, в которой приложение не имеет доступа к недозволенным областям системы.

Так же, у приложения есть варианты предоставления своих данных другим приложениям и доступа к системным службам:

* Можно назначить один идентификатор двум приложениям. Тогда каждый из них сможет обращаться к файлам другого приложения. Для экономии ресурсов системы также можно сделать так, чтобы приложения с одинаковым идентификатором пользователя выполнялись в одном процессе Linux и использовали одну виртуальную машину (приложения также должны быть подписаны одним сертификатом);
* приложение может запрашивать разрешения на доступ к данным устройства, например к контактам пользователя, SMS-сообщениям, подключаемой карте памяти (SD-карте), камере, Bluetooth и др. Все разрешения должны предоставляться приложению при его установке.

**Компоненты приложения**

Приложение для Android состоит из компонентов. Каждый компонент представляет собой отдельную точку, через которую система может взаимодействовать с приложением. Некоторые компоненты являются точками входа для пользователя, а некоторые из них зависят друг от друга. Каждый компонент это самостоятельная структурная единицей и представляет из себя уникальный элемент структуры, который в целом определяет работу приложения.

Существует четыре типа компонентов. Компоненты каждого типа предназначены для определенной цели, обладают собственным жизненнм циклом, который определяет вариант создания и прекращения существования компонента.

* **Операции**

Операция относится к подклассу класса Activity.

Activity представляет собой один экран с пользовательским интерфейсом. Несмотря на то, что операции совместно формируют связное взаимодействие пользователя с приложением, каждая из операций не зависит от других. Любые операции могут быть запущены другим приложением. Например, приложение для камеры может запустить операцию в приложении по работе с электронной почтой, которая составляет новое сообщение, чтобы пользователь мог отослать фотографию.

* **Службы**

Служба относится к подклассу класса Service.

Служба (Service) *представляет* собой компонент, который работает в фоновом режиме и выполняет длительные операции, связанные с работой удаленных процессов. Служба не имеет пользовательского интерфейса. Например, она может воспроизводить музыку в фоновом режиме, пока пользователь работает в другом приложении, или же она может получать данные по сети, не блокируя взаимодействие пользователя с операцией. Служба может быть запущена другим компонентом, который затем будут взаимодействовать с ней, – например операцией.

* **Поставщики контента**

Поставщик контента *(Content provider)* управляет общим набором данных приложения. Данные можно хранить в любом постоянном месте хранения, к которому у данного приложения есть доступ. Посредством поставщика контента другие приложения могут запрашивать или даже изменять данные (если поставщик контента позволяет делать это). Любое приложение, получившее разрешения на управление информацией контактов пользователя, может запросить часть этого поставщика контента.

Также поставщик контента используется для чтения и записи данных, доступ к которым у внешних компонентов не имеет. Например, с помощью поставщика контента в образце приложения Note Pad  выполняется сохранение заметок.

* **Приемники широковещательных сообщений**

Приемник широковещательных сообщений (Broadcast receiver) представляет собой компонент, реагирующий на объявления распространяемые по всей системе. Большую часть этих объявлений рассылает сама система. Приложениями также могут рассылать объявления. Несмотря на отсутствие пользовательского интерфейса, приемники широковещательных сообщений, могут создавать уведомления в строке состояния, чтобы предупредить пользователя о событии "рассылка объявления". Хотя чаще всего они являются просто "шлюзом" для других компонентов и предназначены для выполнения минимального объема работы. К примеру, инициировать выполнение службой определенных действий при возникновении события.

Приемник широковещательных сообщений относится к подклассу класса BroadcastReceiver , а каждое такое сообщение предоставляется как объект Intent.

Уникальной особенностью системы Android является возможность любого приложения запустить компонент другого приложения. Для пользователя же это выглядит как одно приложение.

Когда система запускает компонент, она запускает для этого приложения процесс (если он еще не был запущен) и создает экземпляры классов, нужных этому компоненту. Поэтому, в отличие от приложений для большинства других систем, в приложениях для Android отсутствует единая точка входа (например, в них нет функции main()).

Из-за того что система выполняет каждое приложение в отдельном процессе с такими правами доступа к файлам, которые ограничивают доступ в другие приложения, ваше приложение не может вызвать компонент из другого приложения напрямую. Это может сделать сама система Android. Поэтому, необходимо сообщить системе о своем намерении (*Intent*) запустить определенный компонент, чтобы вызвать компонент в другом приложении. После этого системой активируется этот компонент.

**Активация компонентов**

Компоненты трех из четырех возможных типов — операции, службы и приемники широковещательных сообщений — активируются асинхронным сообщением, которое называется *Intent* (намерение). Объекты Intent связывают друг с другом отдельные компоненты во время выполнения.

Объект Intent создается с помощью объекта Intent, который описывает запрос на активацию либо конкретного компонента, либо компонента конкретного *типа* — соответственно, намерение Intent может быть явным или неявным.

Для операций и служб Объект Intent определяет действие, которое требуется выполнить, а также может указывать URI (Uniform Resource Identifier – унифицированный идентификатор ресурса) данных, с которыми это действие нужно выполнить. Например, объект Intent может передавать запрос на выполнение операции "показать изображение" или "открыть веб-страницу". В некоторых ситуациях операцию можно запустить, чтобы получить результат. В этом случае операция возвращает результат также в виде объекта Intent.

Для приемников широковещательных сообщений Intent просто определяет передаваемое объявление (например, широковещательное сообщение о низком уровне заряда аккумулятора содержит только строку "аккумулятор разряжен").

Компоненты четвертого типа – поставщики контента – сообщениями Intent не активируются. Они активируются по запросу от ContentResolver. Процедура определения контента (content resolver) обрабатывает все прямые транзакции с поставщиком контента, с тем, чтобы этого не пришлось делать компоненту, который выполняет транзакции с поставщиком. Вместо этого он вызывает методы для объекта ContentResolver. Это формирует слой, абстрагирующий (в целях безопасности) поставщика контента от компонента, запрашивающего информацию.

Для активации компонентов каждого типа имеются отдельные методы:

* Можно запустить операцию (или определить для нее какое-то новое действие), передав объект Intent методу startActivity() или startActivityForResult() (если требуется, чтобы операция вернула результат).
* Можно запустить службу (либо выдать работающей службе новые инструкции), передав объект Intent методу startService(). Либо можно установить привязку к службе, передав объектIntent методу bindService().
* Можно инициировать рассылку сообщений, передав объект Intent таким методам, какsendBroadcast(), sendOrderedBroadcast() и sendStickyBroadcast().
* Можно выполнить запрос к поставщику контента, вызвав метод query() для объекта ContentResolver.

**Глава 2.Мое приложение**

Изначально планировалось создание одного приложения, с экраном регистрации, базой данных пользователей, содержащей логины и пароли пользователей, для входа в приложение. Но из-за количества разных элементов в одном приложении, время ожидания увеличивалось параллельно увеличению количества разных элементов, что привело к идее разделить экран регистрации \ входа и экран самого приложения на два разных приложения. Итак, рассмотрим Экран регистрации \ входа (Рис.1)

На главном экране мы видим две кнопки Вход и Регистрация, так же два поля для ввода логина и пароля, две иконки социальных сетей ВКонтакте и Facebook, а так же еще одна кнопка. При нажатии на кнопку Регистрация, открывается другое окно (рис.2), в котором пользователь вводит свое имя, и дважды свой пароль, во избежание ошибок в написании. По завершении регистрации, перед нами опять открывается главный экран (рис.1). По кнопке Вход, происходит проверка, если введенный логин и пароль существуют в базе данных, то пользователь входит в приложение и появляется экран, сигнализирующий, что пользователь вошел в приложение, а так же кнопка, позволяющая вернуться назад. Третья кнопка, на главном экране, служебная и не будет присутствовать в самом приложении, это же лишь прототип и здесь можно посмотреть все логины и пароли нашей базы данных. База данных – локальная, поэтому все данные хранятся на телефоне. Небольшие иконки соц. сетей, в будущем, могут быть использованы для входа в приложение, но сейчас данной функцией не обладают.

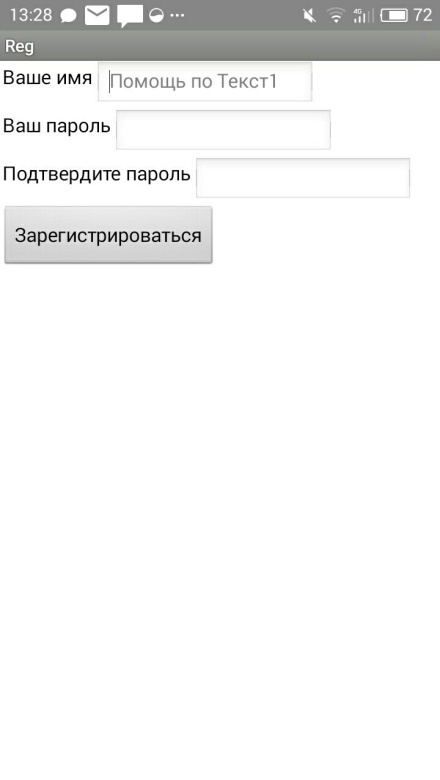
Далее рассмотрим вторую часть моего приложения, приложение определяющее расстояние до гимназии (рис.3), на экране расположены 3 кнопки: проверка доступности, открыть карту и определение расстояния до школы. По кнопке проверка доступности проверяется возможность использования гео-датчика для определения нашего местоположения. По кнопке открыть карту, открывается карта Google, на которой, если нет доступа к gps, показывает расположение гимназии 1505, а если работает гео-датчик, то определяет наше местоположение и возвращает координаты в приложение. По кнопке расстояние до школы, соответственно рассчитывается расстояние между нынешним местоположением и расположением гимназии 1505, при помощи сферической теоремы косинусов: C:\Users\1\Desktop\Снимок.PNG, где C:\Users\1\Desktop\Снимок1.PNG иC:\Users\1\Desktop\Снимок2.PNG являются широтами, а C:\Users\1\Desktop\Снимок1.PNG и C:\Users\1\Desktop\Снимок1.PNG являются долготами, R – радиус Земли в метрах(6371000 в среднем). Для вычислений нужно перевести широты и долготы в радианы, при вычислении возможны небольшие отклонения, за счет формы Земли, которая не является сферой.

Так же создано третье приложение (рис.4), которое является прототипом приложения, определяющего расстояния между пользователями, на примере шариков, которые можно двигать по экран. Координаты двигаемого шара отображаются в строке, номер которой совпадает с номером шара. Если расстояние от двигаемого шара до любого из других шариков меньше либо равно 100 пикселям, то это расстояние записывается в строку и отображается на экране, если же расстояние больше 100 пикселей, то мы видим надпись «вне зоны», что означает, что шарики находятся на расстоянии больше 100 пикселей. Расстояние между шариками вычисляется по средствам теоремы Пифагора, так как шарики находятся в одной плоскости.

**Заключение**

В рамках данной работы было создано 2 приложения, являющихся частями одного, получилось создать приложение, использующее геопозиционирование и рассчитывающее расстояние между координатами гимназии и нынешним местоположением пользователя. Так же получилось создать форму входа и регистрации для приложения, но, только с локальной базой данных. Так же создан макет приложения, определяющего расстояние до ближайших пользователей, на примере передвигаемых по экрану шариков, между которыми определяется расстояние в пикселях. Не удалось создать приложение, которое являлось бы многопользовательским и обладало не локальной базой данных из-за трудностей с созданием такой базы данных, а так же имело бы возможность определять расстояние до других пользователей.

**Приложение**



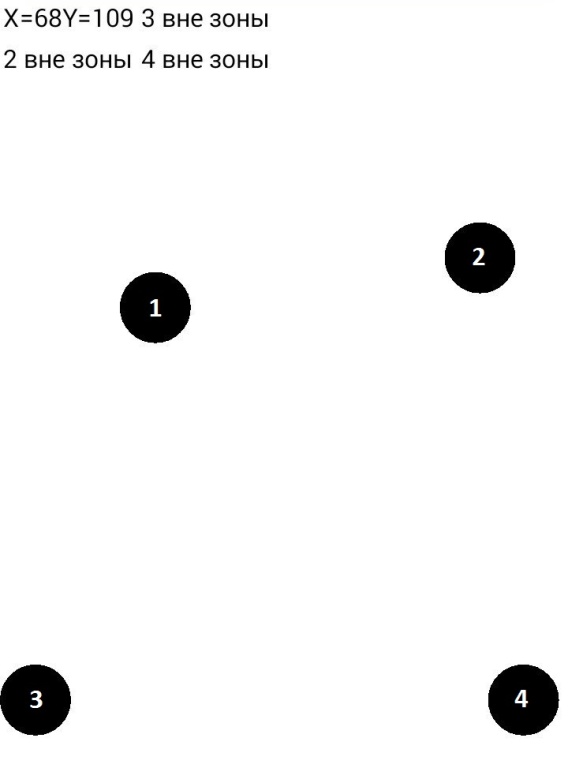


Рисунок 4

Рисунок 2

Рисунок 3

Рисунок 1

**Приложение входа и регистрации – главный экран (работа кнопок)**

Вход

Уведомление об ошибке

Логин и пароль есть в базе данных

Открытие нового экрана

Нет

Да

Регистрация

Открытие экрана регистрации

База пользователй

Открывает список всех логинов и паролей в базе данных

Clear TDB

Очистка базы данных пользователей

**Приложение входа и регистрации - экран регистрации (работа кнопок)**

Зарегистрироваться

Поле логина не пустое и поле пароля не пустое и пароли совпадают

Нет

Да

Уведомление «Проверьте все поля на предмет ошибок»

Записать значение ячеек логин и пароль в базу данных

**Приложение, определяющее расстояние до гимназии (описание кнопок)**

Список литературы:

Выводится сообщение «false»

Выводится сообщение «true»

Нет

Да

Обладает нынешними координатами(доступен ли gps?)

Проверка доступности

Расстояние до школы

Выполняются вычисления по сферической теореме косинусов

На экран выводится сообщение, в котором указано расстояние до гимназии

Открыть карту

Открывает Google-карты и показывает текущее местоположение, если приложение имеет доступ к gps, иначе показывает на карте гимназию 1505

* <https://developer.android.com/guide/components/fundamentals.html>