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

2.1. Краткое описание темы………………………………………….3

2.2. Актуальность……………………………………………………..3

2.3. Задачи……………………………………………………………..3

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

4. Практическая часть………………………………………………..12

5. Заключение

6. Список литературы

**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 зарегистрированных в системе приложения пользователей через GPS и рассчитывающее примерное расстояние до ближайших пользователей. Для создания такого приложения мне нужно изучить и определить на какой базе будет создано приложение и сервер, который будет отправлять информацию о постоянном местоположении пользователей.

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.

**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 вами код — и все требуемые файлы данных и ресурсов — в файл APK – *программный пакет Android*, который представляет собой файл архива с расширением .apk. В файле APK находится все, что требуется для работы Android-приложения, и он позволяет установить приложение на любом Android

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

* операционная система Android представляет собой многопользовательскую систему Linux, в которой 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, с тем, 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.

Но у приложения есть варианты предоставления своих данных другим приложениям и доступа к системным службам:

* Так же 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 запросить разрешение 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. При этом каждый компонент является самостоятельной структурной единицей и представляет собой уникальный элемент структуры, который определяет работу приложения в целом.

Компоненты приложения можно отнести к одному из четырех типов. Компоненты каждого типа предназначены для определенной цели, они имеют собственный жизненный цикл, который определяет способ создания и прекращения существования компонента.

* **Операции**

Операция относится к подклассу класса Activity.

Операция (Activity) *представляет* собой один экран с пользовательским интерфейсом. Например, в приложении для работы с электронной почтой одна операция может служить для отображения списка новых сообщений, другая – для составления сообщения и третья операция – для чтения сообщений. Несмотря на то, что операции совместно формируют связное взаимодействие пользователя с приложением по работе с электронной почтой, каждая из них не зависит от других операций. Любые из этих операций могут быть запущены другим приложением (если это позволяет приложение по работе с электронной почтой). 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 файловой системе, базе данных, в Интернете или любом другом постоянном месте хранения, к которому у вашего приложения имеется доступ. Посредством поставщика контента другие приложения могут запрашивать или даже изменять данные (если поставщик контента позволяет делать это). Например, в системе Android есть поставщик контента, который управляет информацией контактов пользователя. Любое приложение, получившее соответствующие разрешения, может запросить часть этого поставщика контента (например ContactsContract.Data), для чтения и записи сведений об определенном человеке.

Поставщики контента также используются для чтения и записи данных, доступ к которым внешним компонентам приложение не предоставляет. Например, в образце приложения Note Pad с помощью поставщика контента выполняется сохранение заметок.

Поставщик контента относится к подклассу класса ContentProvider. Он должен реализовывать стандартный набор API-интерфейсов, с помощью которых другие приложения будут выполнять транзакции.

* **Приемники широковещательных сообщений**

Приемник широковещательных сообщений (Broadcast receiver) *представляет* собой компонент, который реагирует на объявления распространяемые по всей системе. Многие из этих объявлений рассылает система — например объявление о том, что экран выключился, аккумулятор разряжен или был сделан фотоснимок. Объявления также могут рассылаться приложениями, — например, чтобы сообщить другим приложениям о том, что какие-то данные были загружены на устройство и теперь готовы для использования. Несмотря на то что приемники широковещательных сообщений не имеют пользовательского интерфейса, они могутсоздавать уведомления в строке состояния, чтобы предупредить пользователя о событии "рассылка объявления". Однако чаще всего они являются просто "шлюзом" для других компонентов и предназначены для выполнения минимального объема работы. Например , они могут инициировать выполнение службой определенных действий при возникновении события.

Приемник широковещательных сообщений относится к подклассу класса BroadcastReceiver , а каждое такое сообщение предоставляется как объект Intent. Подробные сведения изложены в руководстве, посвященном классу BroadcastReceiver.

Уникальной особенностью системы Android является то, что любое приложение может запустить компонент другого приложения. Например, если вы хотите дать пользователю возможность фотографировать, используя камеру устройства, то, поскольку наверняка имеется другое приложение, которое может выполнить это действие, вместо того чтобы разработать операцию фотографирования в своем приложении, вы можете вызвать такое приложение. Вам не нужно внедрять код из приложения для камеры или даже устанавливать на него ссылку. Вместо этого вы можете просто запустить операцию фотографирования из приложения для камеры. По завершении этой операции фотография будет возвращена в ваше приложение, и ее можно будет использовать. Для пользователя это будет выглядеть как одно приложение.

Когда система запускает компонент, она запускает процесс для этого приложения (если он еще не был запущен) и создает экземпляры классов, которые требуются этому компоненту. Например, если ваше приложение запустит операцию фотографирования в приложении для камеры, эта операция будет выполняться в процессе, который относится к этому стороннему приложению, а не в процессе вашего приложения. Поэтому, в отличие от приложений для большинства других систем, в приложениях для Android отсутствует единая точка входа (например, в них нет функции main()).

Поскольку система выполняет каждое приложение в отдельном процессе с такими правами доступа к файлам, которые ограничивают доступ в другие приложения, ваше приложение не может напрямую вызвать компонент из другого приложения. Это может сделать сама система Android. Поэтому, чтобы вызвать компонент в другом приложении, необходимо сообщить системе о своем намерении (*Intent*) запустить определенный компонент. После этого система активирует для вас 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, будь то это компоненты вашего или стороннего приложения (эти объекты Intent можно представить себе в виде мессенджеров, которые посылают другим компонентам запрос на выполнение действий).

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 (например, просмотреть (view) или отправить (send) что-то), а также может указывать URI (Uniform Resource Identifier – унифицированный идентификатор ресурса) данных, с которыми это действие нужно выполнить (помимо прочих сведений, которые нужно знать запускаемому компоненту). 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 (например, можно отправить сообщение Intent, чтобы дать пользователю возможность выбрать контакт и вернуть его вам — в ответном сообщении Intent будет содержаться URI, указывающий на выбранный контакт).

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 данных из базы данных, 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 лишь прототип и здесь можно посмотреть все логины и пароли нашей базы данных.База данных – локальная, все данные хранятся на телефоне. Небольшие иконки соц.сетей, в будущем, могут быть использованы для входа в приложение, но сейчас данной функцией не обладают. Код листов Вход \ Регистрация(рис.3-4).

Далее рассмотрим вторую часть моего приложения, связанную с гео-датчиком и определением расстояния от точки до точки, по уравнению

<https://wikimedia.org/api/rest_v1/media/math/render/svg/349227e7ebb7664d6354a0b970956f994a71f4b6>

**Приложение**



****

Рисунок 4

Рисунок 3

Рисунок 2

Рисунок 1

Список литературы:

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