УДК 004.78

РАЗРАБОТКА МОБИЛЬНОГО КЛИЕНТА ДЛЯ СИСТЕМЫ БЕЗОПАСНОСТИ

Шведов Павел Андреевич

Разработчик ПО;
ООО «Нордавинд-Дубна»;
141980, Московская обл., г. Дубна, ул. Программистов, 4;
e-mail: p.shvedov@nordavind.ru.

В последнее время все больше и больше завоевывают популярность мобильные технологии. Устройства становятся компактнее и производительнее, и область их применения расширяется с каждым годом. Сейчас уже практически каждый имеет при себе портативное устройство, вполне способное снимать на встроенную камеру видео высокого качества, получать данные с GPS спутников. В статье рассматривается пример разработки мобильного приложения, позволяющего автоматизировать передачу видеоинформации с мобильного приложения на сервер, используя Wi-Fi.

Ключевые слова: разработка, мобильные технологии, системы безопасности.

DEVELOPMENT OF THE MOBILE CLIENT FOR SECURITY SYSTEM

Shvedov Pavel

Software-developer;
Nordavind-Dubna, LLC;
141980, Dubna, Moscow reg., Programmistov str., 4;
e-mail: p.shvedov@nordavind.ru.

Mobile technologies become more and more popular. Devices become more powerful and compact, so they become more useful from year to year. Most of people has mobile phone where high-resolution video-camera and GPS module are installed. In this article will be observed an example of developing process of mobile application, which can automate transporting videodata from device to server over Wi-Fi network.

Keywords: development, mobile technologies, security systems.

Введение

Основное направление деятельности компании «Нордавинд» – разработка систем видеонаблюдения. В 2012 году была начата разработка комплексной автоматизированной информационной системы «Безопасный город», отличительной особенностью которой являлось наличие в ней мобильных приложений. Эти приложения направлены на осуществление связи гражданского населения с органами внутренних дел.

Основных задач, которые должны выполняться мобильным приложением, поставлено несколько. Во-первых, пользователь, попав в чрезвычайную ситуацию, должен иметь возможность активировать режим «тревоги», нажатием на, так называемую, «тревожную кнопку». По нажатию на эту кнопку приложение устанавливает соединение с сервером, отправляет координаты местоположения клиента, определяя их с помощью встроенного в устройство GPS модуля, начинает запись видео со встроенной камеры и, в реальном времени, транслирует его на сервер. Координаты клиента отправляются на сервер с некоторой периодичностью, с целью актуализации данных в случае, когда местоположение клиента изменяется. Передача видеоинформации на сервер обеспечивает сохранность данных. Если вдруг устройство будет повреждено или выключено, на сервере сохранится все, вплоть до этого момента. Во-вторых, приложение имеет возможность получать оповещения о чрезвычайном положении, какой-либо другой опасности или просто оповещения, содержащие полезную информацию. В-третьих, в приложении имеется функция, позволяющая сделать фото с камеры устройства и отправить его на сервер вместе с координатами места съемки, с целью обратить внимание соответствующих служб на потенциально опасные объекты или места.

Прежде чем приступать к разработке приложения, было проведено исследование, в ходе которого были изучены основные современные мобильные операционные системы, и их популярности среди пользователей. По результатам исследования был сделан выбор платформы, под которую предстояло разработать приложение. Выбор пал на мобильную операционную систему Android от компании Google.

Android является одной из самых распространенных мобильных операционных систем на рынке подобного рода ПО. Популярность операционной системы от компании Google явно видна по следующей статистике (рис.1):

Топ-5 операционных систем для смартфонов

ОС	Поставки в I кв. 2013, млн шт.	Доля рынка в I кв. 2013	Поставки в I кв. 2012, млн шт.	Доля рынка в I кв. 2012	Рост год к году
Android	162,1	75,0%	90,3	59,1%	79,5%
iOS	37,4	17,3%	35,1	23,0%	6,6%
Windows Phone	7,0	3,2%	3,0	2,0%	133,3%
BlackBerry OS	6,3	2,9%	9,7	6,4%	-35,1%
Linux	2,1	1,0%	3,6	2,4%	-41,7%
Symbian	1,2	0,6%	10,4	6,8%	-88,5%
Прочие	0,1	0,0%	0,6	0,4%	-83,3%
Всего	216,2	100,0%	152,7	100,0%	41,6%

Источник: IDC

Рис. 1. Топ-5 операционных систем для смартфонов

Более того, *Android* является операционной системой с открытым исходным кодом, что способствует постоянному развитию и нестандартному применению данной ОС (например, проект android-x86.org [1] предлагает портированную версию *Android* OS под архитектуру *x*86, при желании любой пользователь может установить данную систему на компьютер или ноутбук в качестве настольной системы, наряду с *Linux* и *Windows*).

Несмотря на глобальное преимущество, ОС *Android* имеет ряд проблем:

1. Высокая фрагментация устройств. Ввиду открытости платформы, любой производитель портативных устройств может кастомизировать и дорабатывать имеющуюся систему для полноценной работы на конкретной аппаратной платформе. Отсюда вытекает множество устройств с разными диагоналями экрана, разными процессорами, видеоускорителями и с разным количеством оперативной памяти.
2. Высокое энергопотребление. Система организована так, что в фоне, с момента включения устройства, начинают работать некоторые приложения без ведома пользователя. Имеются ввиду различные системные службы, или, как их принято называть на узкоспециализированном языке, «демоны». Часть этих «демонов» работают с мобильной сетью, другие периодически лезут в интернет и так далее. Они активны на протяжении всего времени работы устройства. Так же высокий уровень энергопотребления зависит от наличия все более новых и производительных процессоров и видеоадаптеров с не всегда оптимально написанными драйверами.

Упомянутые выше проблемы выделены как главные, поскольку именно эти два аспекта напрямую влияют на процесс разработки приложений и генерируют некоторые сложности, решение которых не всегда тривиально.

Тем не менее, под платформу *Android* существует достаточное количество приложений (свыше 700 000) различного жанра и тематики. Вот самые популярные жанры приложений в *Google Play* [2]:

1. Personalization (персонализация, различные лаунчеры и виджеты).

2. Entertainment (развлечение).
3. Books & Reference (книги и документация).
4. Lifestyle (образ жизни, различные органайзеры).
5. Tools (различные системные утилиты).

Исследовав площадку *Google Play* на предмет приложений, имеющих схожий с запланированным функционалом, был сделан вывод, что подобных приложений катастрофически мало, а качественных среди них – еще меньше. Ни одно из имеющихся приложений не вписывалось в архитектуру системы «Безопасный город», так что было решено реализовывать приложение под данную платформу своими силами.

Разработка приложения

Основной сущностью большинства приложений на *Android* является *Activity*. Реализуя наследников этого класса, мы описываем отдельно каждую «страницу» приложения, если провести аналогию, например, с веб-сайтом. Помимо компоновки и описания графического интерфейса, *Activity* несет в себе основной функционал приложения: обработка событий элементов управление, навигация к другим *Activity*, запуск/остановка служб и так далее.

Для выполнения каких-либо задач в фоне в *Android SDK* имеется класс *Service*. Наследуясь от этого класса мы можем реализовать собственный класс, выполнение которого происходит в отдельном процессе, не мешая работе основного потока. Правда отсюда вытекают некоторые ограничения с точки зрения возможностей данного класса. Из отдельного потока нет возможности работать с GUI приложения. Ниже приведена общая диаграмма классов приложения (рис. 2).

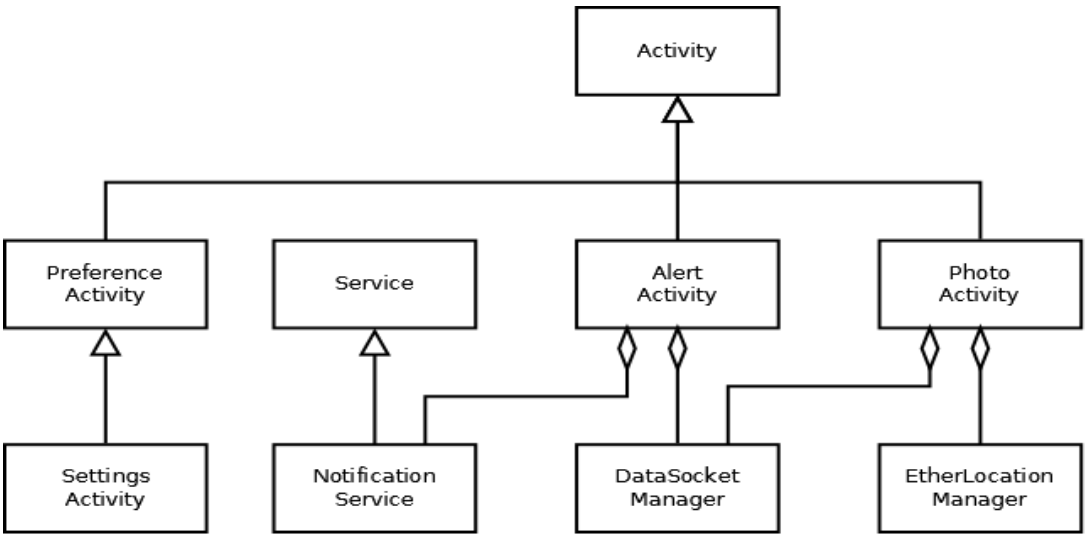

Рис. 2. Диаграмма классов

Preference Activity: класс, входящий в состав *Android SDK*, наследованный от базового класса *Activity*, предназначен для работы с настройками приложений. Его ключевая особенность – генерация *UI* для настроек приложения, основываясь на данных, описанных с помощью *xml*-подобного языка. Для того, чтоб использовать *Preference Activity* в приложении, необходимо реализовать класс, унаследованный от *Preference Activity*, и переопределить класс создания объекта, указав в нем путь к файлу, содержащим определение структуры настроек приложения.

Settings Activity: класс, унаследованный от *Preference Activity*. Содержит переопределенный метод создания объекта данного класса, в котором указан файл со структурой настроек приложения. Для примера, код описания структуры настроек:

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
  <!-- Notifications -->
  <PreferenceCategory
```

```
    android:title="@string/pref_header_notifications">
        <CheckBoxPreference
      android:defaultValue="true"
        android:key="notifications_state"
        android:title="@string/pref_title_notifications" />
    <EditTextPreference
        android:capitalize="words"
        android:defaultValue="@string/pref_default_notifications_server"
        android:inputType="textCapWords"
        android:key="notifications_server"
        android:maxLines="1"
        android:selectAllOnFocus="true"
        android:singleLine="true"
        android:title="@string/pref_title_notifications_server" />
  </PreferenceCategory>
  <!-- Data server -->
  <PreferenceCategory
        android:title="@string/pref_header_data">
   <EditTextPreference
        android:capitalize="words"
        android:defaultValue="@string/pref_default_server_name"
        android:inputType="textCapWords"
        android:key="data_server"
        android:maxLines="1"
        android:selectAllOnFocus="true"
        android:singleLine="true"
        android:title="@string/pref_title_server_name" />
  </PreferenceCategory>
  <!-- Other settings -->
  <PreferenceCategory
        android:title="@string/pref_header_other">
   <CheckBoxPreference
        android:defaultValue="false"
        android:key="autoalert_state"
        android:title="@string/pref_title_autoalert" />
    </PreferenceCategory>
</PreferenceScreen>
```

После обработки приведенного кода, получится следующий интерфейс настроек приложения (рис. 3):

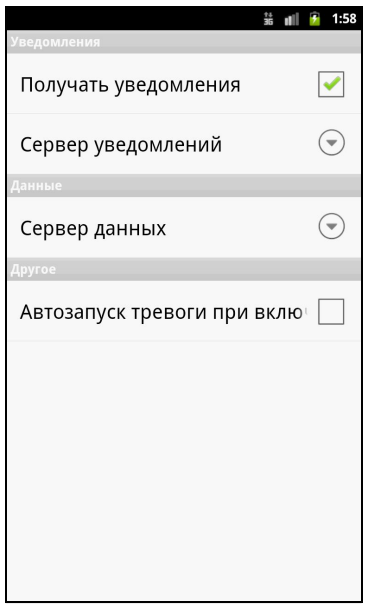

Рис. 3. Интерфейс настроек приложения

***Notification Service*:** класс, наследник базового класса *Service*. Его задача – работая в отдельном потоке, принимать оповещения от сервера, и уведомлять об этом пользователя. Ввиду того, что работа приложения предполагается в сети не имеющей доступа к интернету, использовать стандартный сервис оповещений от *Google* не представлялось возможным. Поэтому было решено реализовать свой сервис оповещений, к которому приложение с неким интервалом будет обращаться за актуальными уведомлениями, каждый раз после запроса сохраняя идентификационный номер последнего полученного оповещения. Данное решение позволяло не зависеть от возможностей доступной сети и предоставляло полный контроль над содержанием и получением оповещений.

Структура данных, которой обмениваются сервер с клиентом была выбрана сразу – *JSON*, так как, во-первых – в состав *Android SDK* входят классы для работы с этим форматом, а во-вторых – это легкий и не нагруженный формат передачи данных.

***DataSocketManager*:** класс-менеджер, в который инкапсулирована вся работа по поддержке соединения с сервером, и передачи данных по сети. Объекты этого класса предоставляют другим сущностям приложения возможность отправлять и принимать необходимую информацию, будь то *GPS* координаты или видеокадры.

***EtherLocationManager*:** вспомогательный класс, задача которого – получение координат от встроенного, в устройство, *GPS*-модуля. Из любой точки приложения можно получить актуальную информацию об актуальном местоположении, создав экземпляр этого класса и вызвав единственный публичный метод *getActualLocation()*, возвращаемый тип которого – *Location*. Это стандартный класс, входящий в состав *Android SDK*, описывающий местоположение клиента.

***Photo Activity*:** класс, наследник от базового класса *Activity*. Объект этого класса содержит описание пользовательского интерфейса приложения в режиме «снять фото с камеры». Он отвечает за создание элементов управления, добавления их на видимую область дисплея приложения, назначение стилей и шаблонов и обработку различных действий пользователя.

Стили оформления и разметку шаблонов принято описывать вне программного кода, используя *xml*-подобный язык. Данное решение весьма популярно в современных технологиях (*WPF* с их *XAML*, *Flex* с *MXML*, *JavaScript* и *HTML*/*CSS*). Такое разделение функционала от описания позволяет программистам и дизайнерам работать над одним и тем же объектом, не мешая друг другу. Пример описания шаблона для *Photo Activity*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">
    <SurfaceView
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:id="@+id/photo_surfaceView" android:layout_gravity="center_horizontal|left" android:layout_weight="30"/>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent" android:layout_weight="10" android:layout_gravity="center">
      <ImageButton
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Capture"
          android:background="@drawable/side_button"
          android:src="@drawable/capture_icon_normal"
          android:id="@+id/capture_button" android:layout_weight="1" android:scaleType="centerInside"/>
      <View android:layout_height="1px"
          android:layout_width="match_parent"
          android:background="#42444b"
           />
      <ImageButton
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:background="@drawable/side_button"
          android:src="@drawable/back_icon_normal"
          android:id="@+id/back_button" android:layout_weight="1" android:scaleType="centerInside"/>
    </LinearLayout>
</LinearLayout>
```

В итоге получаем пользовательский интерфейс вида (рис. 4):

Рис. 4. Интерфейс Photo Activity

***AlertActivity*:** пожалуй, ключевой объект приложения. Эта *Activity* содержит описание ключевых элементов управления, например «тревожной кнопки». Только с помощью этого объекта возможно перейти на другие *Activity*. При создании этой сущности запускается служба оповещений – *NotificationService*. На этот объект также возложена задача захвата видео с камеры устройства. К сожалению,

API Android OS не предоставляет средств для организации реал-тайм видеопотока. Система подразумевает запись конечного файла, записывая нужные заголовки с метаданными в конец, что делает невозможным без каких-либо изменений воспроизводить видео на стороне получателя данных. Преобразовывать видео на стороне клиента влечет за собой использование сторонних библиотек, стабильность которых не гарантирована на разных устройствах, также более бюджетные аппараты будут делать это крайне медленно. Перекладывать работу на сервер – при большом количестве клиентов возможна нехватка ресурсов на сервере. На одном из форумов в интернете была почерпнута идея получать кадры, зажатые в *JPEG* с определенной частотой, система позволяет достичь *FPS* порядка тридцати кадров в секунду, что более чем приемлемо для данной ситуации. На сервере был реализован компонент, собирающий из отдельных кадров полноценный видеопоток, с нужным *FPS*. Интерфейс *Alert Activity* представлен на рис. 5:

Рис. 5. Интерфейс Alert Activity

Заключение

Как результат рассмотренного примера процесса разработки было получено мобильное приложение для платформы *Android*, решающее поставленные ему задачи по экстренной передаче видеопотока с мобильного устройства. На данном этапе разработка всей системы еще продолжается, и данное приложение планируется портировать на другие, чуть менее популярные, но не менее востребованные, мобильные ОС.

Список литературы

1. Android-x86 – Porting Android to x86. – [Электронный ресурс]. URL: http://www.android-x86.org.
2. Google Play. – [Электронный ресурс]. URL: http://play.google.com.
3. Статистика популярных мобильных ОС. – [Электронный ресурс]. URL: http://info.sibnet.ru/?id=351470.
4. Документация Android SDK. – [Электронный ресурс]. URL: http://developer.android.com.