УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
«10» 10 2018 г.

БАКАЛАВРИАТ

Кафедра «Компьютерные информационные системы и управление»

Учебно-методический комплекс дисциплины

Средства визуальной разработки приложений

Направление: **710100 «Информатика и вычислительная техника»**

Профиль: **Компьютерные информационные системы в бизнесе**

Академическая степень **- бакалавр**

Форма обучения **(очная)**

График проведения модулей8-семестр

Нед.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Лек.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Лаб.	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.

Составитель к.т.н., и.о., доцента Нежинских С.С.

Директор Научной библиотеки Асанова Ж.Ш.

БИШКЕК 2018

ОГЛАВЛЕНИЕ

АННОТАЦИЯ 3
УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ) 4
1. Пояснительная записка 4
1.1. Миссия и стратегия 4
1.2. Цель и задачи дисциплины (модулей) 4
1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю) 4
1.4. Место дисциплины (модулей) в структуре ООП ВПО 5
2. Структура и содержание дисциплины (модулей) 5
3. Конспект лекций 6
4. Информационные и образовательные технологии 6
5. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей) 7
5.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины 7
5.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности 7
5.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания 9
6. Учебно-методическое и информационное обеспечение дисциплины 10
6.1. Список источников и литературы 10
6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей) 10
7. Материально-техническое обеспечение дисциплины 10
8. Приложения 11

АННОТАЦИЯ

Цель дисциплины (модулей) заключается в подготовке выпускника, владеющего специализированными знаниями в области применения современной техники для решения прикладных задач обработки данных на мобильных устройствах, получение высшего профессионального образования, позволяющего выпускнику успешно работать в избранной сфере деятельности с применением современных мобильных информационных технологий.

На изучение дисциплины отводится 120 часов в восьмом семестре. Рубежный контроль успеваемости проводится на 4, 8, 11, 14 неделях. Формы текущего контроля: опрос, проверка заданий, посещаемость. Форма рубежного контроля — модульная работа. Форма итогового контроля — экзамен.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)

1. Пояснительная записка

1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества.
Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.
Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины (модулей)

Целью дисциплины является подготовка выпускника, владеющего специализированными знаниями в области применения современной техники для решения прикладных задач обработки данных на мобильных устройствах, получение высшего профессионального образования, позволяющего выпускнику успешно работать в избранной сфере деятельности с применением современных мобильных информационных технологий.

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю)

Дисциплина (модуль) направлена на формирование следующих компетенций:

- профессиональными (ПК):
  - (ПК-2);
  - разрабатывать интерфейсы «человек - электронно-вычислительная машина» (ПК-3);
  - (ПК-10).

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. Знать:
   - основные компоненты архитектуры мобильных платформ;
   - основные элементы пользовательского интерфейса мобильных приложений и их структуру;
   - работу с файлами, базами данных, пользовательскими настройками в мобильных приложениях;
   - возможности взаимодействия с геолокационными, картографическими сервисами.
2. Уметь:
   - настраивать программные интерфейсы, обеспечивающие функции телефонии, отправки/получения SMS;
   - разрабатывать приложения для мобильных операционных систем.
3. Владеть:
   - инструментами для программирования и основ проектирования мобильных приложений;

- навыками практического применения инструментальных средств и методов разработки мобильных приложений.

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина (модуль) является частью общенаучного цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника». Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: программирование, базы данных.

2. Структура и содержание дисциплины (модулей)

Структура дисциплины (модулей) для очной формы обучения
Общая трудоемкость дисциплины составляет 4 кредита, 120 ч., в том числе аудиторная работа обучающихся с преподавателем 60 ч., самостоятельная работа обучающихся 60 ч.

№ п/п	*Раздел, Темы Дисциплины*	*Семестр*	*Неделя семестра*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
				Лекции	*Сем /лаб*	*СРС*	*СРСП*	
Раздел 1.								
1	Основы работы со средой разработки приложений для мобильных устройств.	8	1	1	3	3	1	опрос, проверка задания, посещаемость
2	Создание каркаса работоспособного приложения.	8	2	1	3	3	1	опрос, проверка задания, посещаемость
3	Формирование интерфейса пользователя.	8	3	1	3	3	1	опрос, проверка задания, посещаемость
4	Передача программы пользователю, подписывание программ.	8	4	1	3	3	1	Модуль 1
Раздел 2.								
5	Обращение с данными и их долговременное хранение.	8	5	1	3	3	1	опрос, проверка задания, посещаемость
6	Использование поставщиков содержимого.	8	6	1	3	3	1	опрос, проверка задания, посещаемость

7	Организация поиска.	8	7	1	3	3	1	опрос, проверка задания, посещаемость
8	Поисковый интерфейс.	8	8	1	3	3	1	Модуль 2
Раздел 3.								
9	Работа с картами и геолокационными системами.	8	9	1	3	3	1	опрос, проверка задания, посещаемость
10	Использование мультимедийных возможностей мобильных устройств.	8	10	1	3	3	1	опрос, проверка задания, посещаемость
11	Коммуникация, личные данные, синхронизация и социальные сети.	8	11	1	3	3	1	Модуль 3
Раздел 4.								
12	Эффективное использование памяти.	8	12	1	3	4	1	опрос, проверка задания, посещаемость
13	Энергоэффективность приложений.	8	13	1	3	4	1	опрос, проверка задания, посещаемость
14	Оптимизация мобильных приложений.	8	14	1	3	4	1	Модуль 4
15	Консультация	8	15	1	3	0	1	посещаемость

3. Конспект лекций

Конспект лекций можно просмотреть в приложении 1.

4. Информационные и образовательные технологии

№ п/п	**Наименование раздела**	**Виды учебной работы**	**Формируемые компетенции (указывается код компетенции)**	**Информационные и образовательные технологии**
1	Разделы: 1, 2, 3, 4	Лекция	ПК-2, ПК-3, ПК-10	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций

		Лабораторная работа	ПК-2, ПК-3, ПК-10	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ПК-2, ПК-3, ПК-10	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

5. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

5.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

№ п/п	Контролируемые разделы дисциплины (модулей)	Код контролируемой компетенции (компетенций)	Наименование оценочного средства
1	Разделы №1, №2, №3, №4	ПК-2	опрос, выполнение лабораторных работ
2	Разделы №1, №2, №3, №4	ПК-3	опрос, выполнение лабораторных работ
3	Разделы №1, №2, №3, №4	ПК-10	опрос, выполнение лабораторных работ

5.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль:			
- опрос	1, 2, 3, 4 недели	8 баллов	До 32 баллов
- выполнение лабораторных работ	1, 2, 3, 4 недели	10 баллов	До 40 баллов
- посещаемость	1, 2, 3, 4 недели	2 балла	8 баллов

Рубежный контроль: (сдача модуля)	4 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	5, 6, 7, 8 недели	8 баллов	До 32 баллов
- выполнение лабораторных работ	5, 6, 7, 8 недели	10 баллов	До 40 баллов
- посещаемость	5, 6, 7, 8 недели	2 балла	8 баллов
Рубежный контроль: (сдача модуля)	8 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	9, 10, 11 недели	7 баллов	До 21 баллов
- выполнение лабораторных работ	9, 10, 11 недели	15 баллов	До 45 баллов
- посещаемость	9, 10, 11 недели	4 балла	12 баллов
Рубежный контроль: (сдача модуля)	11 неделя	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	12, 13, 14 недели	7 баллов	До 21 баллов
- выполнение лабораторных работ	12, 13, 14 недели	15 баллов	До 45 баллов
- посещаемость	12, 13, 14 недели	4 балла	12 баллов
Рубежный контроль: (сдача модуля)	14 неделя	100%×0,2=20 баллов	
Итого за IV модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
от 0 до 54	неудовлетворительно
от 55 до 69	удовлетворительно
от 70 до 84	хорошо
от 85 до 100	отлично

5.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0-80 баллов)

При оценивании посещаемости, опроса и выполнении лабораторных работ учитываются:

- посещаемость (10 баллов)
- степень раскрытия содержания материала (25 баллов);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (20 баллов);
- знание теории изученных вопросов, сформированность и устойчивость используемых при ответе умений и навыков (25 баллов).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 10 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 5 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность – 5 баллов.

Итоговый контроль (экзаменационная сессия) – ИК = Бср × 0,8 + Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (0 баллов);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (10 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (20 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (30 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (0-9 баллов);
- ответ содержит 21-89 % правильного решения (10-39 баллов);
- ответ содержит 90% и более правильного решения (40 баллов).

6. Учебно-методическое и информационное обеспечение дисциплины

6.1. Список источников и литературы

Литература:

- Основная:
  1. Java 7, Хабибуллин, Ильдар Шаукатович, 2012г.
  2. Java, Васильев, Алексей Николаевич, 2013г. Технология Java, Хабибуллин, Ильдар Шаукатович, 2010г.
  3. Голощапов А. Л. Google Android: программирование для мобильных устройств. - СПб.: БХВ-Петербург, 2010. - 448 с.
- Дополнительная:
  1. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java, Медведев, Владислав Иосифович, 2013г.
  2. Мобильные сообщения: службы и технологии SMS, EMS и MMS, Ле-Бодик, Гвинель, 2005г.
  3. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java, Медведев, Владислав Иосифович, 2011г.
  4. Практикум по программированию на языке JAVA, Пинягина, Ольга Владиславовна;Кашина, Ольга Андреевна;Андрианова, Анастасия Александровна, 2007г.
  5. Справочник по высшей математике, Выгодский, Марк Яковлевич, 2004г.
  6. Современные датчики. Справочник, Фрайден, Дж.;Свинцов, Е. Л.;Заболотная , Ю. А., 2006г.

6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

- //www.intuit.ru
- //developer.android.com/docs
- http://www.iprbookshop.ru/
- http://kyrlibnet.kg/ru/
- http://biblioteka.kg/

7. Материально-техническое обеспечение дисциплины

Для изучения дисциплины, необходимо следующее оборудование: ЭВМ, проектор.

Требования к аудитории: компьютерный класс, имеющий ЭВМ в количестве идентичном количеству обучающихся, ЭВМ для преподавателя с подключенным проектором, наличие доски и средств для отображения/удаления информации на доске (мел/ветошь, маркер/губка).

8. Приложения

Приложение 1

Android - операционная система для мобильных устройств: смартфонов, планшетных компьютеров, КПК. В настоящее время именно Android является самой широко используемой операционной системой для мобильных устройств. Подтверждение этого факта можно найти в таблице, составленной по данным аналитической компании Gartner.

Таблица 1.1. Мировые продажи смартфонов конечным пользователям, распределение по ОС

Операционная система	Продано (тыс.ед.) III кв. 2013	Доля рынка (%) III кв. 2013	Продано (тыс.ед.) III кв. 2012	Доля рынка (%) III кв. 2012
Android	205022,7	81,9	124552,3	72,6
iOS	30330,0	12,1	24620,3	14,3
Microsoft	8912,3	3,6	3993,6	2,3
BlackBerry	4400,7	1,8	8946,8	5,2
Bada	633,3	0,3	4454,7	2,6
Symbian	457,5	0,2	4401,3	2,6
другие	475,2	0,2	683,7	0,4
Общее кол-во:	250231,7	100,0	171652,7	100,0

Источник: Gartner (ноябрь 2013)

Внимательное изучение таблицы позволяет увидеть подавляющую популярность смартфонов под управлением ОС Android в мире, доля таких устройств не первый год превышает половину от общего числа купленных смартфонов. Кроме всего прочего, эта популярность продолжает расти. Очевидно, что армия пользователей смартфонов под управлением Android будет искать дополнительные приложения для своих устройств, в связи с этим умение разрабатывать эти самые приложения может принести много пользы своему владельцу. Например, можно разрабатывать для себя полезные, интересные, занимательные (нужное подчеркнуть) приложения, а можно, разведав обстановку и осмотревшись, сделать разработку мобильных приложений своей профессиональной деятельностью, основной или дополнительной.

Курс "Разработка приложений для смартфонов на ОС Android" предоставляет возможность приобрести начальные навыки разработки мобильных приложений, если остановиться только на первой его части. Изучение полной версии курса позволит сделать серьезный шаг к тому, чтобы профессионально разрабатывать мобильные приложения и получать от этой деятельности не только моральное, но и материальное удовлетворение.

Данная лекция является первой для всего курса, призвана ввести читателя в курс дела. В первую очередь в ней рассматриваются вопросы становления и развития ОС Android. Для успешного программирования под Android необходимо понимать внутреннюю организацию и архитектуру этой платформы, а также полезно знать, какие инструменты и среды разработки можно использовать. Этим вопросам посвящена основная часть лекции. Кроме того, в лекции рассматриваются особенности запуска и отладки мобильных приложений.

Немного истории

Рассмотрим, как все начиналось. В 2003 году в Пало Альто, штат Калифорния Энди Рубин с единомышленниками (Рич Майнер, Ник Сирс и Крис Уайт) основали компанию Android Inc. Поначалу в компании занимались проектированием мобильных гаджетов, которые на основе геолокационных данных автоматически подстраивались под нужды пользователей.

В августе 2005 года Android Inc. стала дочерней компанией Google. Энди Рубин, Рич Майнер и Крис Уайт остались в Android Inc. и начали работать над операционной системой,

базирующейся на ядре Linux. В Google задумали реализовать мощнейшую платформу, пригодную к использованию на тысячах различных моделей телефонов. В связи с этим был создан Open Handset Alliance (OHA) - консорциум, состоящий из более 80 компаний, направляющий свои усилия на разработку открытых стандартов для мобильных устройств. В состав OHA входят такие гиганты, как Google (организатор и идейный вдохновитель), HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung Electronics, LG Electronics, T-Mobile, Sprint Corporation, NVIDIA и многие другие.

Первая версия Android была представлена 23 сентября 2008 года, версии было дано название Apple Pie (можно заметить созвучие с прямым конкурентом). Далее так повелось, что название каждой очередной версии представляет какой-либо десерт, при этом первые буквы наименований в порядке версий соответствуют буквам латинского алфавита по порядку. С развитием обновлений Android можно познакомиться в таблице 1.2.
Таблица 1.2. История обновлений ОС Android Версия, логотип, дата выхода Основные возможности

Android 1.0

Apple Pie

Первый стабильный релиз, основан на ядре Linux 2.6.25.
Поддерживается:

- файловая система FAT32, стек интернет-протоколов TCP/IP;
- протоколы передачи данных: 802.11 b/g Wi-Fi,Bluetooth 2.0 EDR, GPRS, EDGE, UMTS, HSDPA;
- фото и видео съемка, однако недостаточно опций для настройки разрешения камеры, баланса белого и др.;
- сенсорные дисплеи и landscape режим отображения данных на экране, максимальная цветность дисплея - 16 бит (тип HVGA);
- виджеты и ярлыки на рабочем столе (Home Screen), сменные обои;
- регулярные телефонные функции, контроль вызова, конференц-связь, легкая интеграция с контактами;
- полноценный web-браузер на движке WebKit, HTML, XHTML;
- e-mail клиент, протоколы POP3, IMAP4, SMTP;
- медиа проигрыватель, позволяющий управлять, импортировать, проигрывать медиа контент в различных форматах.

Базовые приложения:
будильник; калькулятор; календарь; камера; контакты; сообщения (в том числе MMS); настройки; голосовой набор.
Минимальные системные требования для запуска и работы:
архитектура ARM, 128 MB RAM, 256 MB ROM.
Видео презентация:
http://www.youtube.com/watch?feature=player_embedded&v=1FJHYqE0RDg

Android 1.1

Banana Bread
февраль 2009

(API level: 2)

Нововведения:
Исправлены проблемы: с будильником; со спящим режимом; с вызовом дисплея набора номера; в IMAP ошибки запроса пароля и др.
Изменения API.
Добавлены подробности и отзывы к картам.
Добавлена поддержка вложений из MMS.
Локализации:
Английская US (en_US)
Немецкая (de)
Подробности:
http://developer.android.com/about/versions/android-1.1.html
Android 1.5
Cupcake

апрель 2009
(API level: 3)

Нововведения:
Поддержка экранной клавиатуры (портретный и книжный режимы); акселерометра; видеозапись и воспроизведение видео; приложение для работы с YouTube; стерео Bluetooth; функция копирования и вставки между приложениями (copy&paste).
Локализации:
добавились очень многие, в том числе и русская (ru_RU).
Система:
новое Linux ядро (версия 2.6.27); автоматическая проверка и восстановление файловой системы на SD card; новое приложение для просмотра СТК меню оператора (SIM Application Toolkit 1.0).
Изменения в пользовательском интерфейсе (UI):
изменено большинство UI-элементов, добавлены новые виджеты; определение режима (книжный или портретный) работы программы; анимированное переключение между окнами.
Подробности:
http://developer.android.com/about/versions/android-1.5.html

Android 1.6

Donut
сентябрь 2009
(API level: 4)

Нововведения:
Система:
новое ядро Linux (версия 2.6.29); поддержка сотового стандарта CDMA; поддержка разрешений дисплеев: QVGA и WVGA; обновленный медиа-движок OpenCore 2; движок синтеза речи (многоязыковой); Gesture Builder поддержка возможности (для разработчиков) создавать, сохранять, загружать и распознавать жесты, прикреплять к определенным действиям.
Пользовательские возможности:
строка быстрого поиска (прямо с рабочего стола): история и закладки в браузере, контакты и

поиск в интернете; возможность подключаться к видам VPN: L2TP/IPSEC pre-shared key based VPN, L2TP/IPSEC certificate based VPN, L2TP only VPN, PPTP only VPN; ускорение работы камеры; индикатор работы батареи позволяет увидеть сколько энергии потребляют работающие программы и сервисы.
Обновленный Android Market.
Подробности:
http://developer.android.com/about/versions/android-1.6.html
Android 2.0,
2.0.1, 2.1

Eclair
октябрь 2009
(API level: 5)
(API level: 6)
(API level: 7)

Нововведения в 2.0:
поддержка работы нескольких почтовых аккаунтов одновременно, возможность использования совместных папок (входящие, исходящие) для всех аккаунтов; быстрый способ работы с контактами Quick Contact;
поиск по всем сохраненным SMS и MMS сообщениям, удаление старых после заданного срока;
возможности камеры: вспышка, цифровой зум, сценические режимы, баланс белого, цветовые эффекты, макрофокусировка;
улучшенное расположение виртуальных клавиш клавиатуры, поддержка комбинированных нажатий клавиш (технология мультитач), усовершенствованная функция автодополнения;
поддержка HTML5, версии Bluetooth 2.1, новых профилей OPP и PBAP.
Подробности:
http://developer.android.com/about/versions/android-2.0.html
Нововведения 2.0.1:
подрелиз версии 2.0, включающий в себя незначительные изменения в функционале и по большей части bugfix-ом версии 2.0.
Подробности:
http://developer.android.com/about/versions/android-2.0.1.html
Нововведения 2.1:
основным новшеством, представляющим интерес для конечного пользователя, стало добавление анимированных (живых) обоев, остальные изменения в Framework API, представляют интерес для разработчиков.
Подробности:
http://developer.android.com/about/versions/android-2.1.html

Android 2.2

Froyo
май 2010
(API level: 8)

Нововведения:
рост производительности примерно в 3-5 раз за счет использования Dalvik Virtual Machine Just-in-Time компилятора (Dalvik устарел, вместо него используется ART);

возможности установки приложений на SD-карту, переноса приложений из внутренней памяти на карту и обратно;
возможность использовать смартфон в качестве точки доступа к интернету, в качестве модема для других устройств;
поддержка Adobe Flash;
V8 javascript существенно повысил скорость работы штатного браузера.
Подробности:
http://developer.android.com/about/versions/android-2.2.html
Android 2.3,
2.3.3

Gingerbread
декабрь 2010
(API level: 9)
(API level: 10)

До весны 2013 года самая массовая версия на рынке.
Нововведения:
новое ядро Linux 2.6.35; поддержка открытых мультимедийных стандартов (VP8 и WebM), форматов ACC/AMR, звуковых эффектов и эквалайзера, фронтальной камеры (интеграция с VOIP(SIP));
обновленный GUI: уменьшение времени доступа к функциям, повышение общей энергоэффективности системы;
улучшение стандартной клавиатуры системы: поддержка словарей, технологии мультитач, упрощенное выделение и копирование текста;
поддержка технологии NFC; расширение возможностей работы с датчиками положения телефона.
Подробности:
http://developer.android.com/about/versions/android-2.3.html

Android 3.0-3.2

Honeycomb
февраль 2011
(API level: 11)
(API level: 12)
(API level: 13)

Специальная версия для работы на планшетах (MID, tablets).
Нововведения 3.0:
новое ядро Linux 2.6.36; поддержка файловой системы ext4, файловой системы FUSE для MTP устройств; поддержка режима USB-хост для работы с клавиатурой, мышью и USB-хабами; поддержка MTP/PTP;
виртуальная машина Dalvik: поддержка и оптимизация SMP, множество улучшений JIT, улучшенный сборщик мусора;
совершенно новый интерфейс с полноценной оптимизацией под устройства с большими экранами; поддержка виртуальных рабочих столов, каждый из которых может иметь свой набор виджетов и ярлыков;
улучшенные и переработанные базовые приложения: Browser, e-mail и др.
Подробности:

http://developer.android.com/about/versions/android-3.0.html
Нововведения 3.1:
поддержка работы кардридера;
усовершенствован GUI: доработан менеджер задач, позволяющий переключаться между множеством различных приложений (в 3.0 только 5 программ одновременно); возможность менять размер виджетов, как по горизонтали, так и по вертикали.
Подробности:
http://developer.android.com/about/versions/android-3.1.html
Нововведения 3.2:
расширен спектр поддерживаемых планшетов; возможность автоматического масштабирования приложений для отображения на более крупных экранах.
Подробности:
http://developer.android.com/about/versions/android-3.2.html
Android 4.0, 4.0.3
Ice Cream Sandwich

ноябрь 2011
(API level: 14)
(API level: 15)

Нововведения:
поддержка и смартфонов, и планшетов; поддержка новых процессорных архитектур, помимо ARM поддержка Intel x86 и MIPS;
возможность разблокировки экрана: при помощи функции определения лица; жестами: перетащить замочек из центра экрана на иконку приложения и оно запустится;
многозадачность: кнопка Recent Apps позволяет мгновенно переходить от одной задачи к другой с помощью списка в системной панели;
новые элементы управления передачей данных через сеть: в приложении Настройки можно увидеть общее использование данных по каждому типу сети, объем данных, используемых каждым работающим приложением;
доступность Android 4.0 для слепых и слабовидящих пользователей, браузер поддерживает экранного чтеца, который воспроизводит все видимое активное содержимое на экране;
AndroidBeam - удобное средство обмена между двумя NFC-устройствами;
Wi-Fi Direct и Bluetooth HDP, HFP: возможность прямого подключения к соответствующим устройствам.
Подробности:
http://developer.android.com/about/versions/android-4.0.html

Android 4.1-4.3

Jelly Bean
июль 2012
(API level: 16)
(API level: 17)
(API level: 18)

Нововведения 4.1:
увеличена скорость прорисовки интерфейса, улучшен поиск, добавлено несколько полезных сервисов;
улучшена работа со словарями, возможно использовать голосовой ввод без подключения к

интернету;
специальные возможности: возможность управления смартфоном с помощью жестов и голосовых подсказок, подключения устройств ввода, поддерживающих шрифт Брайля;
существенно доработана функция передачи данных Beam; переработан поиск (вместо ссылок ответ на запрос); голосовой поиск; Google Now: нужная информация в нужное время.
Подробности:
http://developer.android.com/about/versions/android-4.1.html
Нововведения 4.2:
реализована поддержка нескольких пользователей (планшеты); поддержка wireless display: возможность трансляции видео и изображений на внешний экран; возможность отображения полезной информации в режиме сна, при подключении к док-станции или на зарядке; улучшена панель уведомлений.
Подробности:
http://android.com.ua/android_42.html
Нововведения 4.3:
ускорение работы системы; более точный набор на клавиатуре; скрытая возможность управления процессами программ (необходима активация); поддержка OpenGL/ES 3.0 (не на всех устройствах).
Подробности:
http://developer.android.com/about/versions/android-4.3.html

Android 4.4

Kit Kat
октябрь 2013
(API level: 19)

Нововведения:
многозадачность, оптимизация распределения ресурсов между приложениями; определитель номера работает не только с адресной книгой (например, Google maps); серьезная интеграция приложения Hangouts (отправка SMS, MMS, голосовые и видеовызовы); в состав вошел Quickoffice, интегрированный с Google Drive; поддержка принтеров, подключение через приложения поддерживающие печать (например, Google Cloud Print, HP ePrint); поддержка стандарта Wi-Fi Miracast, позволяющий вещать изображение на телевизор; возможность захвата экрана для записи видео.
Подробности:
http://developer.android.com/about/versions/android-4.4.html

1.2 Устройство платформы Android

Платформа Android объединяет операционную систему, построенную на основе ядра ОС Linux, промежуточное программное обеспечение и встроенные мобильные приложения. Разработка и развитие мобильной платформы Android выполняется в рамках проекта AOSP (Android Open Source Project) под управлением OHA (Open Handset Alliance), руководит всем процессом поисковый гигант Google.

Android поддерживает фоновое выполнение задач; предоставляет богатую библиотеку элементов пользовательского интерфейса; поддерживает 2D и 3D графику, используя OpenGL стандарт; поддерживает доступ к файловой системе и встроенной базе данных SQLite.

С точки зрения архитектуры, система Android представляет собой полный программный стек, в котором можно выделить следующие уровни:

Базовый уровень (Linux Kernel) - уровень абстракции между аппаратным уровнем и программным стеком;

Набор библиотек и среда исполнения (Libraries & Android Runtime) обеспечивает важнейший базовый функционал для приложений, содержит виртуальную машину Dalvik и базовые библиотеки Java необходимые для запуска Android приложений;

Уровень каркаса приложений (Application Framework) обеспечивает разработчикам доступ к API, предоставляемым компонентами системы уровня библиотек;

Уровень приложений (Applications) - набор предустановленных базовых приложений.

Наглядное изображение архитектуры на рисунке 1.1.
Архитектура Android

увеличить изображение
Рис. 1.1. Архитектура Android

Рассмотрим компоненты платформы более подробно.

В основании компонентной иерархии лежит ядро ОС Linux 2.6 (несколько урезанное), служит промежуточным уровнем между аппаратным и программным обеспечением, обеспечивает функционирование системы, предоставляет системные службы ядра: управление памятью, энергосистемой и процессами, обеспечение безопасности, работа с сетью и драйверами.

Уровнем выше располагается набор библиотек и среда исполнения. Библиотеки реализуют следующие функции:

предоставляют реализованные алгоритмы для вышележащих уровней;

обеспечивает поддержку файловых форматов;

осуществляет кодирование и декодирование информации (например, мультимедийные кодеки);

выполняет отрисовку графики и т.д.

Библиотеки реализованы на C/C++ и скомпилированы под конкретное аппаратное обеспечение устройства, вместе с которым они и поставляются производителем в предустановленном виде.

Рассмотрим некоторые библиотеки:

Surface Manager - композитный менеджер окон. Поступающие команды отрисовки собираются в закадровый буфер, где они накапливаются, составляя некую композицию, а потом выводятся на экран. Это позволяет системе создавать интересные бесшовные эффекты, прозрачность окон и плавные переходы.

Media Framework - библиотеки, реализованные на базе PacketVideo OpenCORE. Используются для записи и воспроизведения аудио и видео контента, а также для вывода статических изображений. Поддерживаются форматы: MPEG4, H.264, MP3, AAC, AMR, JPG и PNG.

SQLite - легковесная и производительная реляционная СУБД, используется в Android в качестве основного движка для работы с базами данных.

3D библиотеки - используются для высокооптимизированной отрисовки 3D-графики, при

возможности используют аппаратное ускорение. Библиотеки реализованы на основе API OpenGL|ES. OpenGL|ES (OpenGL for Embedded Systems) - подмножество графического программного интерфейса OpenGL, адаптированное для работы на встраиваемых системах.
FreeType - библиотека для работы с битовыми картами, для растеризации шрифтов и осуществления операций над ними.
LibWebCore - библиотеки браузерного движка WebKit, используемого также в известных браузерах Google Chrome и Apple Safari.
SGL (Skia Graphics Engine) - открытый движок для работы с 2D-графикой. Графическая библиотека является продуктом Google и часто используется в других программах.
SSL - библиотеки для поддержки одноименного криптографического протокола.
Libc - стандартная библиотека языка C, а именно ее BSD реализация, настроенная для работы на устройствах на базе Linux.

Среда исполнения включает в себя библиотеки ядра, обеспечивающие большую часть низкоуровневой функциональности, доступной библиотекам ядра языка Java, и виртуальную машину Dalvik, позволяющую запускать приложения. Каждое приложение запускается в своем экземпляре виртуальной машины, тем самым обеспечивается изоляция работающих приложений от ОС и друг от друга. Для исполнения на виртуальной машине Dalvik Java-классы компилируются в исполняемые файлы с расширением .dex с помощью инструмента dx, входящего в состав Android SDK. DEX (Dalvik EXecutable) - формат исполняемых файлов для виртуальной машины Dalvik, оптимизированный для использования минимального объема памяти. При использовании IDE Eclipse и плагина ADT (Android Development Tools) компиляция классов Java в формат .dex происходит автоматически.

Архитектура Android Runtime такова, что работа программ осуществляется строго в рамках окружения виртуальной машины, что позволяет защитить ядро ОС от возможного вреда со стороны других ее составляющих. Поэтому код с ошибками или вредоносное ПО не смогут испортить Android и устройство на его базе, когда сработают.

На еще более высоком уровне располагается каркас приложений (Application Framework), архитектура которого позволяет любому приложению использовать уже реализованные возможности других приложений, к которым разрешен доступ. В состав каркаса входят следующие компоненты:

- богатый и расширяемый набор представлений (Views), который может быть использован для создания визуальных компонентов приложений, например, списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера;
- контент-провайдеры (Content Providers), управляющие данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы;
- менеджер ресурсов (Resource Manager), обеспечивающий доступ к ресурсам без функциональности (не несущим кода), например, к строковым данным, графике, файлам и другим;
- менеджер оповещений (Notification Manager), позволяющий приложениям отображать собственные уведомления для пользователя в строке состояния;
- менеджер действий (Activity Manager), управляющий жизненными циклами приложений, сохраняющий историю работы с действиями, предоставляющий систему навигации по действиям;
- менеджер местоположения (Location Manager), позволяющий приложениям периодически получать обновленные данные о текущем географическом положении устройства.

Application Framework предоставляет в распоряжение приложений в ОС Android вспомогательный функционал, благодаря чему реализуется принцип многократного использования компонентов приложений и ОС. Естественно, в рамках политики безопасности.

И, наконец, самый высокий, самый близкий к пользователю уровень приложений. Именно на этом уровне пользователь взаимодействует со своим устройством, управляемым ОС Android. Здесь представлен набор базовых приложений, который предустановлен на ОС Android. Например, браузер, почтовый клиент, программа для отправки SMS, карты, календарь, менеджер контактов и др. Список интегрированных приложений может меняться в зависимости от модели устройства и версии Android. К этому уровню также относятся все пользовательские приложения.

Разработчик обычно взаимодействует с двумя верхними уровнями архитектуры Android для создания новых приложений. Библиотеки, система исполнения и ядро Linux скрыты за каркасом приложений.

Повторное использование компонентов других приложений приводит к идее задач в Android. Приложение может использовать компоненты другого Android приложения для решения задачи, например, если разрабатываемое приложение предполагает использование фотографий, оно может вызвать приложение, управляющее фотографиями и зарегистрированное в системе Android, выбрать с его помощью фотографию и работать с ней.

Для пополнения коллекции приложений своего мобильного устройства пользователь может воспользоваться приложением Google Play, которое позволяет покупать и устанавливать приложения с сервиса Google Play. Разработчики, в свою очередь, могут выкладывать свои приложения в этот сервис, Google Play отслеживает появление обновлений приложения, сообщает пользователям этого приложения об обновлении и предлагает установить его. Также Google Play предоставляет разработчикам доступ к услугам и библиотекам, например, доступ к использованию и отображению Google Maps.

Для установки приложения на устройствах с ОС Android создается файл с расширением *.apk (Android package), который содержит исполняемые файлы, а также вспомогательные компоненты, например, файлы с данными и файлы ресурсов. После установки на устройство каждое приложение "живет" в своем собственном изолированном экземпляре виртуальной машины Dalvik.

1.3 Обзор сред программирования

Прежде чем начать разрабатывать приложения под Android, рассмотрим существующие инструменты, подходящие для этих целей. Можно выделить необходимые инструменты, без которых разработка мобильных приложений под Android просто невозможна. С другой стороны, существует большое количество вспомогательных систем, в какой-то мере упрощающих процесс разработки.

К обязательным инструментам относится Android SDK - набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения.

Рассмотрим кратко наиболее важные инструменты, входящие в состав Android SDK:

SDK Manager - инструмент, позволяющий загрузить компоненты Android SDK. Показывает пакеты Android SDK и их статус: установлен (Installed), не установлен (Not Installed), доступны обновления (Update available).

Android SDK Manager

увеличить изображение

Рис. 1.2. Android SDK Manager

Debug Monitor - самостоятельный инструмент, предоставляющий графический интерфейс к нескольким инструментам, предназначенным для анализа и отладки Android приложений:

DDMS (Dalvik Debug Monitor Server) предоставляет услуги переброса портов, захват экрана устройства, информацию о потоках и динамической памяти устройства, вывод информации о действиях Android в реальном времени (logcat) и многое другое.

Hierarchy Viewer позволяет отлаживать и оптимизировать пользовательский интерфейс Android приложения.

Tracer for OpenGL ES - инструмент для анализа OpenGL|ES кода, используемого в мобильном приложении, позволяет захватывать команды OpenGL|ES и демонстрировать их по отдельным кадрам, что помогает понять как исполняются графические команды.

Окно инструмента Monitor

увеличить изображение

Рис. 1.3. Окно инструмента Monitor

Android Emulator (emulator) - виртуальное мобильное устройство, которое создается и работает на компьютере разработчика, используется для разработки и тестирования мобильных приложений без привлечения реальных устройств.

AVD Manager - предоставляет графический интерфейс для создания виртуальных Android устройств (AVDs), предусмотренных Android Emulator, и управления ими.

(В ЛР№1 подробно рассматривается создание и использование виртуального устройства).

Android Debug Bridge (adb) - гибкий инструмент, позволяющий управлять состоянием эмулятора или реального Android устройства, подключенного к компьютеру. Также может использоваться для установки Android приложения (.apk файл) на реальное устройство.

Мы рассмотрели основные инструменты, входящие в состав Android SDK, разумеется, не все и недостаточно подробно. Для более серьезного изучения инструментов имеет смысл обратиться к сайту разработчиков (http://developer.android.com/tools/help/index.html). Для разработки мобильных приложений под Android уверенного владения инструментами из SDK вполне достаточно. Если же возникают какие-то вопросы, дополнительные инструкции по созданию проектов, компиляции, запуску из командной строки содержатся в руководстве от Google (http://developer.android.com/training/basics/firstapp/index.html).

В современных условиях разработка ПО в большинстве случаев ведется с использованием интегрированных сред разработки (IDE). IDE имеют несомненные достоинства: процесс компиляции, сборки и запуска приложения обычно автоматизирован, в связи с чем для начинающего разработчика создать свое первое приложение труда не составляет. Но чтобы заниматься разработкой всерьез, необходимо потратить силы и время на изучение возможностей самой среды. Рассмотрим IDE, пригодные для разработки под Android1.

Для начала поговорим о двух средах разработки, которые рекомендует Google: Android IDE

(ADT) и Android Studio.

Android IDE - среда разработки под Android, основанная на Eclipse. Предоставляет интегрированные инструменты для разработки, сборки и отладки мобильных приложений. В данном курсе Android IDE выбрана в качестве основной среды разработки. Возможности этой среды более подробно рассмотрены в первой лабораторной работе. Также там даны рекомендации по установке и настройке среды, созданию и запуску первого приложения как на эмуляторе, так и на реальном устройстве.

Android Studio - среда разработки под Android, основанная на IntelliJ IDEA. Подобно Android IDE, она предоставляет интегрированные инструменты для разработки и отладки. Дополнительно ко всем возможностям, ожидаемым от IntelliJ, в Android Studio реализованы:

поддержка сборки приложения, основанной на Gradle;
специфичный для Android рефакторинг и быстрое исправление дефектов;
lint инструменты для поиска проблем с производительностью, с юзабилити, с совместимостью версий и других;
возможности ProGuard (утилита для сокращения, оптимизации и обфускации кода) и подписи приложений;
основанные на шаблонах мастера для создания общих Android конструкций и компонентов;
WYSIWYG редактор, работающий на многих размерах экранов и разрешений, окно предварительного просмотра, показывающее запущенное приложение сразу на нескольких устройствах и в реальном времени;
встроенная поддержка облачной платформы Google.

Загрузить последнюю версию Android Studio, а также получить рекомендации по установке, настройке и началу работы можно тут: http://developer.android.com/sdk/installing/studio.html.
Среда разработки Android Studio

увеличить изображение
Рис. 1.4. Среда разработки Android Studio

Перейдем к рассмотрению других инструментов, пригодных для разработки мобильных приложений под Android. Начнем с инструментов от Intel - Intel XDK и Intel Beacon Mountain.

Intel XDK позволяет легко разрабатывать кроссплатформенные мобильные приложения; включает в себя инструменты для создания, отладки и сборки ПО, а также эмулятор устройств; поддерживает разработку для Android, Apple iOS, Microsoft Windows 8, Tizen; поддерживает языки разработки: HTML5 и JavaScript.

Последняя тема данного курса полностью посвящена изучению нового поколения инструментальных средств разработки мобильных HTML5-приложений и Intel XDK, предполагается разработка мобильного приложения с использованием этих инструментов.

Intel Beacon Mountain - среда разработки, позволяющая создавать приложения для устройств, работающих под управлением ОС Android. Предоставляет инструменты необходимые для проектирования, разработки, отладки и оптимизации приложений под Android. Освобождает разработчика от необходимости поддерживать систему разработки в актуальном состоянии, следит за обновлениями и добавляет их в среду разработки по мере появления. Поддерживает разработку для целевых платформ на основе процессоров Intel Atom и ARM.

Beacon Mountain построена на основе Android IDE (Eclipse, Android ADT, Android SDK), для более серьезной разработки и оптимизации добавлены следующие инструменты Intel:

Intel* Hardware Accelerated Execution Manager (Intel* HAXM) - аппаратно поддерживаемый процессор виртуализации, использующий технологию виртуализации Intel* (Intel* VT) для ускорения работы эмулятора в среде разработки.

Intel* Graphics Performance Analyzers (Intel* GPA) System Analyzer поддерживает мобильные устройства с процессором Intel Atom под управлением ОС Android. Позволяет разработчикам оптимизировать загруженность системы при использовании процедур OpenGL, предоставляя возможность получать множество системных метрик в реальном времени, отображающих загруженность CPU, GPU и OpenGL ES API. Разработчик может запустить несколько графических экспериментов для выявления узких мест в обработке графики.

Intel* Integrated Performance Primitives (Intel* IPP) Preview - библиотека оптимизированной обработки данных и изображений, поддерживающая мобильные устройства с платформой Intel под управлением ОС Android. Preview версия является частью полной версии Intel IPP, которая тоже поддерживает ОС Android.

Intel* Threading Building Blocks (Intel* TBB) - широко используемая, признанная библиотека шаблонов C++ для создания масштабируемых приложений и увеличения производительности. Поддерживает мобильные устройства с платформой Intel под управлением Android. Проверенные алгоритмы позволяют разработчикам эффективно распараллелить C++ мобильные приложения, что повышает производительность при снижении энергетических затрат.

Загрузить Intel Beacon Mountain можно по ссылке http://software.intel.com/ru-ru/vcsource/tools/beaconmountain
Страница поддержки Intel* Beacon Mountain

увеличить изображение
Рис. 1.5. Страница поддержки Intel* Beacon Mountain

Нельзя обойти вниманием инструментарий Marmalade SDK.

Marmalade SDK - кроссплатформенное SDK от Ideaworks3D Limited. Представляет собой набор библиотек, образцов, инструментов и документации, необходимых для разработки, тестирования и развертывания приложений для мобильных устройств. Используется, в основном, для разработки игр. Многие получившие признание игры, такие как Cut the Rope и Plants vs. Zombies, были разработаны с использованием этого программного средства. К сожалению, Marmalade SDK представляет собой проприетарное программное обеспечение (самая дешевая лицензия $15 в месяц) и не может быть рекомендована в данном учебном курсе, но читатель может самостоятельно попробовать бесплатную 30-дневную версию, доступную по ссылке https://www.madewithmarmalade.com/free-trial.

Нельзя не сказать об отечественных разработках. Например, компания 1С идет в ногу со временем, версия платформы 1С 8.3 позволяет разрабатывать мобильные приложения. Программный продукт "1С:Предприятие 8. Расширение для карманных компьютеров" обеспечивает возможность работы с данными информационных баз 1С:Предприятия 8 на мобильных устройствах (карманных компьютерах, коммуникаторах, терминалах сбора данных), а также на персональных компьютерах (в том числе ноутбуках), не имеющих

прямого доступа к информационным базам 1С:Предприятия 8.

1.4 Эмуляторы

Эмуляция. Стандартный эмулятор Android

Эмуляция (англ. emulation) в вычислительной технике - комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или эмулирования) функций одной вычислительной системы (гостя) на другой, отличной от первой, вычислительной системе (хосте) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (гостя). Целью является максимально точное воспроизведение поведения в отличие от разных форм компьютерного моделирования, в которых имитируется поведение некоторой абстрактной модели (Википедия).

Эмулятор - виртуальное мобильное устройство, которое запускается на компьютере. При помощи эмулятора можно разрабатывать и тестировать приложения без использования реальных устройств. На рисунке 1.7 приведен пример запущенного стандартного эмулятора. Подробно работа с эмуляторами рассмотрена в лабораторной работе.

увеличить изображение

увеличить изображение
Рис. 1.7. Эмулятор Android SDK в процессе запуска и приложение "Hello, world!"

К достоинствам использования эмуляторов можно отнести простоту их использования и нулевую стоимость. Разработчику не нужно покупать огромное количество устройств с различными характеристиками, чтобы проверить работоспособность приложения на различных смартфонах. Достаточно создать несколько эмуляторов с требуемыми характеристиками и запустить на них приложение.

К сожалению, эмуляторы имеют и ряд недостатков:

Требуют много системных ресурсов.

Из-за различий в архитектуре процессоров компьютера и смартфона медленно запускаются. Современные персональные компьютеры построены на архитектурах x86 и x64, а большинство процессоров смартфонов на Android - ARM. Процесс эмуляции одной архитектуры на другой чрезвычайно сложен и происходит довольно медленно.

В некоторых случаях стандартного эмулятора недостаточно. Речь идет о возможностях смартфонов, которыми обычные компьютеры не обладают (например, наличие датчика gps или акселерометра). В таких случаях полноценную отладку можно провести только с использованием реального устройства.

Альтернативные эмуляторы

Стандартный эмулятор, поставляемый вместе с Android SDK, не устраивает многих. Существуют проекты, поддерживающие разработку и развитие альтернативных эмуляторов. В качестве примера можно привести Genymotion (см. рис. 1.8) - быстрый эмулятор Android (по мнению его разработчиков). Он содержит предварительно настроенные образы Android (x86 с аппаратным ускорением OpenGL). Genymotion доступен для Linux, Windows и Mac OS X и требует для своей работы VirtualBox. Иными словами, Genymotion представляет собой

виртуальную машину с установленной ОС Android, которую пользователь запускает так же, как и другие виртуальные машины. Проблема высокого потребления системных ресурсов, конечно, не исчезает, однако скорость запуска существенно увеличивается.

В настоящее время активно развивается.
Альтернативный эмулятор Genymotion

увеличить изображение
Рис. 1.8. Альтернативный эмулятор Genymotion
1.5 Возможности отладки на реальных устройствах

Разработанное приложение можно запустить на реальном устройстве, например, на смартфоне. Для этого необходимо проделать предварительную работу.

Для запуска приложений, разработанных в Android IDE, необходимо:

Настроить устройство (включить режим отладки по USB).
Настроить компьютер (для Windows необходимо установить нужный драйвер вручную, нужны права администратора).
Настроить среду и запустить проект на устройстве.

Подробности отладки на реальных устройствах описаны в лабораторной работе.
1.6 Примеры приложений

Google Play - это магазин приложений от Google, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения. Учётная запись разработчика, которая даёт возможность публиковать приложения, стоит $25. В настоящее время Google Play насчитывает более миллиона различных приложений, каждый месяц пользователями загружается несколько миллиардов. Разумеется, далеко не все из них высокого качества и поддерживаются разработчиками, встречается и вредоносное программное обеспечение.

В настоящий момент доступно более 30 различных категорий приложений. Внутри каждой категории приложения упорядочены на основании рейтинга, отзывов, количества скачиваний, страны распространения и других факторов.

Время от времени редакция Google Play собирает коллекции приложений или игр, основанных на теме или сезонном событии. Коллекции пользуются популярностью у клиентов за счет своевременности и актуальности.

Приведем примеры интересных и удобных приложений, заслуженно удерживающих высокие места в своих категориях уже долгое время.

Популярное игровое приложение Cut the Rope позволяет разобраться в правилах игры прямо в ее процессе и не требует чтения сложных инструкций (см. рисунок 1.9). Идея игры предельно проста - в коробке сидит маленький зелёный монстр Ам Ням, которого надо кормить леденцами. Леденцы болтаются на веревках, и их надо правильно перерезать, чтобы леденец попал точно в рот Ам Няма. По ходу игры сложность уровней возрастает, появляются дополнительные препятствия. Попутно надо собирать звездочки, которые позволяют открывать новые локации.

Первый уровень игры Cut the Rope

Рис. 1.9. Первый уровень игры Cut the Rope

Если вам нужен интерактивный помощник, который способен понимать сделанные устно указания и напоминать о делах в нужное время, приложение "Помнить все" (см. рис. 1.10) - то, что вам нужно. Используя библиотеку для распознавания речи, оно анализирует полученную информацию, отображает ее в виде текста, который при необходимости можно исправить, и устанавливает время для напоминания.

В "Введение в разработку мобильных приложений" были рассмотрены общие вопросы, связанные с операционной системой Android, а также инструменты, используемые для разработки мобильных приложений. Изучены основы разработки интерфейсов мобильных приложений. В данной теме обсуждаются вопросы, связанные, непосредственно, с разработкой мобильных приложений для устройств, работающих под управлением Android.

Для начала предполагается рассмотреть еще несколько общих вопросов: во-первых, какие виды мобильных приложений существуют и каковы особенности каждого вида; во-вторых, как организовано исполнение приложений в ОС Android и каким образом обеспечивается безопасная среда их функционирования. Понимание этих вопросов позволяет вести более осознанную разработку приложений.

Невозможно создать осмысленное приложение, не изучив внутреннюю организацию, свойственную приложениям, работающим на определенной платформе. В данном курсе, очевидно, необходимо изучить структуру и основные компоненты приложений, разрабатываемых для работы на смартфонах под управлением ОС Android. От типа мобильного устройства внутренняя организация приложений не зависит, т. е. Android-приложения, разработанные для смартфонов вполне смогут выполняться и на планшетах. В данной лекции рассматривается архитектура Android приложений, основанная на идее многократного использования компонентов, которые являются основными строительными блоками. Подробно описываются основные компоненты, а также такие важные понятия для мобильных приложений, работающих под управлением Android, как манифест приложения и ресурсы.

3.2 Основные виды Android-приложений

Приступая к разработке мобильных приложений хорошо бы иметь представление о том, какие виды приложений существуют. Дело в том, что если удастся определить к какому типу относится приложение, то становится понятнее на какие моменты в процессе его разработки необходимо обращать основное внимание. Можно выделить следующие виды приложений:

Приложения переднего плана выполняют свои функции только, когда видимы на экране, в противном же случае их выполнение приостанавливается. Такими приложениями являются, например, игры, текстовые редакторы, видеопроигрыватели. При разработке таких приложений необходимо очень внимательно изучить жизненный цикл активности, чтобы переключения в фоновый режим и обратно проходили гладко (бесшовно), т. е. при возвращении приложения на передний план было незаметно, что оно вообще куда-то пропадало. Для достижения этой гладкости необходимо следить за тем, чтобы при входе в фоновый режим приложение сохраняло свое состояние, а при выходе на передний план восстанавливало его. Еще один важный момент, на который обязательно надо обратить внимание при разработке приложений переднего плана, удобный и интуитивно понятный

интерфейс1.

Фоновые приложения после настройки не предполагают взаимодействия с пользователем, большую часть времени находятся и работают в скрытом состоянии. Примерами таких приложений могут служить, службы экранирования звонков, SMS-автоответчики. В большинстве своем фоновые приложения нацелены на отслеживание событий, порождаемых аппаратным обеспечением, системой или другими приложениями, работают незаметно. Можно создавать совершенно невидимые сервисы, но тогда они будут неуправляемыми. Минимум действий, которые необходимо позволить пользователю: санкционирование запуска сервиса, настройка, приостановка и прерывание его работы при необходимости.

Смешанные приложения большую часть времени работают в фоновом режиме, однако допускают взаимодействие с пользователем и после настройки. Обычно взаимодействие с пользователем сводится к уведомлению о каких-либо событиях. Примерами таких приложений могут служить мультимедиа-проигрыватели, программы для обмена текстовыми сообщениями (чаты), почтовые клиенты. Возможность реагировать на пользовательский ввод и при этом не терять работоспособности в фоновом режиме является характерной особенностью смешанных приложений. Такие приложения обычно содержат как видимые активности, так и скрытые (фоновые) сервисы, и при взаимодействии с пользователем должны учитывать свое текущее состояние. Возможно потребуется обновлять графический интерфейс, если приложение находится на переднем плане, или же посылать пользователю уведомления из фонового режима, чтобы держать его в курсе происходящего. И эти особенности необходимо учитывать при разработке подобных приложений.

Виджеты - небольшие приложения, отображаемые в виде графического объекта на рабочем столе. Примерами могут служить, приложения для отображения динамической информации, такой как заряд батареи, прогноз погоды, дата и время. Разумеется, сложные приложения могут содержать элементы каждого из рассмотренных видов. Планируя разработку приложения, необходимо определить способ его использования, только после этого приступать к проектированию и непосредственно разработке.

3.3 Безопасность

Обратим внимание на организацию исполнения приложений в ОС Android. Как уже было отмечено приложения под Android разрабатываются на языке программирования Java, компилируется в файл с расширением .apk, после этот файл используется для установки приложения на устройства, работающие под управлением Android. После установки каждое Android приложение "живет" в своей собственной безопасной "песочнице", рассмотрим, как это выглядит:

операционная система Android является многопользовательской ОС, в которой каждое приложение рассматривается как отдельный пользователь;

по умолчанию, система назначает каждому приложению уникальный пользовательский ID, который используется только системой и неизвестен приложению;

система устанавливает права доступа ко всем файлам приложения следующим образом: доступ к элементам приложения имеет только пользователь с соответствующим ID;

каждому приложению соответствует отдельный Linux процесс, который запускается, как только это необходимо хотя бы одному компоненту приложения, процесс прекращает работу, когда ни один компонент приложения не использует его или же системе требуется освободить память для других (возможно, более важных) приложений;

каждому процессу соответствует отдельный экземпляр виртуальной машины Dalvik, в связи с этим код приложения исполняется изолировано от других приложений.

Перечисленные идеи функционирования приложения в ОС Android реализуют принцип минимальных привилегий, т. е. каждому приложению, по умолчанию, разрешен доступ только к компонентам, необходимым для его работы и никаким больше. Таким образом обеспечивается очень безопасная среда функционирования приложений.

Однако, в случае необходимости приложения могут получить доступ к данным других приложений и системным сервисам (услугам). В случае, когда двум приложениям необходимо иметь доступ к файлам друг друга, им присваивается один и тот же пользовательский ID. Для экономии системных ресурсов такие приложения запускаются в одном Linux процессе и делят между собой один и тот же экземпляр виртуальной машины, в этом случае приложения также должны быть подписаны одним сертификатом. В случае же, когда приложению требуется доступ к системным данным, например, контактам, SMS сообщениям, картам памяти, камере, Bluetooth и т. д., пользователю необходимо дать приложению такие полномочия во время установки его на устройство.

3.4 Архитектура приложения, основные компоненты

Вот и пришла пора поговорить непосредственно о внутренней организации приложений под Android: обсудить их архитектуру и основные компоненты.

Архитектура Android приложений основана на идее многократного использования компонентов, которые являются основными строительными блоками. Каждый компонент является отдельной сущностью и помогает определить общее поведение приложения.

Система Android выстроена таким образом, что любое приложение может запускать необходимый компонент другого приложения. Например, если приложение предполагает использование камеры для создания фотографий, совершенно необязательно создавать в этом приложении активность для работы с камерой. Наверняка на устройстве уже есть приложение для получения фотографий с камеры, достаточно запустить соответствующую активность, сделать фотографию и вернуть ее в приложение, так что пользователь будет считать, что камера часть приложения, с которым он работает.

Когда система запускает компонент, она запускает процесс приложения, которому принадлежит компонент, если он еще не запущен, и создает экземпляры классов, необходимых компоненту. Поэтому в отличие от большинства других систем, в системе Android приложения не имеют единой точки входа (нет метода main(), например). В силу запуска каждого приложения в отдельном процессе и ограничений на доступ к файлам, приложение не может напрямую активировать компонент другого приложения. Таким образом для активации компонента другого приложения необходимо послать системе сообщение о намерении запустить определенный компонент, система активирует его.

Можно выделить четыре различных типа компонентов, каждый тип служит для достижения определенной цели и имеет свой особый жизненный цикл, который определяет способы создания и разрушения соответствующего компонента. Рассмотрим основные компоненты Android-приложений.

Активности (Activities). Активность - это видимая часть приложения (экран, окно, форма), отвечает за отображение графического интерфейса пользователя. При этом приложение может иметь несколько активностей, например, в приложении, предназначенном для работы с электронной почтой, одна активность может использоваться для отображения списка новых

писем, другая активность - для написания, и еще одна - для чтения писем. Несмотря на то, что для пользователя приложение представляется единым целым, все активности приложения не зависят друг от друга. В связи с этим любая из этих активностей может быть запущена из другого приложения, имеющего доступ к активностям данного приложения. Например, приложение камеры может запустить активность, создающую новые письма, чтобы отправить только что сделанную фотографию адресату, указанному пользователем.

Сервисы (Services). Сервис - компонент, который работает в фоновом режиме, выполняет длительные по времени операции или работу для удаленных процессов. Сервис не предоставляет пользовательского интерфейса. Например, сервис может проигрывать музыку в фоновом режиме, пока пользователь использует другое приложение, может загружать данные из сети, не блокируя взаимодействие пользователя с активностью. Сервис может быть запущен другим компонентом и после этого работать самостоятельно, а может остаться связанным с этим компонентом и взаимодействовать с ним.

Контент-провайдеры (Content providers). Контент-провайдер управляет распределенным множеством данных приложения. Данные могут храниться в файловой системе, в базе данных SQLite, в сети, в любом другом доступном для приложения месте. Контент-провайдер позволяет другим приложениям при наличии у них соответствующих прав делать запросы или даже менять данные. Например, в системе Android есть контент-провайдер, который управляет информацией о контактах пользователя. В связи с этим, любое приложение с соответствующими правами может сделать запрос на чтение и запись информации какого-либо контакта. Контент-провайдер может быть также полезен для чтения и записи приватных данных приложения, не предназначенных для доступа извне.

Приемники широковещательных сообщений (Broadcast Receivers). Приемник - компонент, который реагирует на широковещательные извещения. Большинство таких извещений порождаются системой, например, извещение о том, что экран отключился или низкий заряд батареи. Приложения также могут инициировать широковещание, например, разослать другим приложениям сообщение о том, что некоторые данные загружены и доступны для использования. Хотя приемники не отображают пользовательского интерфейса, они могут создавать уведомление на панели состояний, чтобы предупредить пользователя о появлении сообщения. Такой приемник служит проводником к другим компонентам и предназначен для выполнения небольшого объема работ, например, он может запустить соответствующий событию сервис.

Все рассмотренные компоненты являются наследниками классов, определенных в Android SDK.
Иерархия классов Android SDK

Рис. 3.1. Иерархия классов Android SDK

(источник: http://habrahabr.ru/post/141201/)

На рис. 3.1 показана иерархия основных классов Android SDK, с которыми обычно имеет дело разработчик. На самом деле классов намного больше, желтым цветом выделены классы, с которыми разработчик работает непосредственно, наследует от них свои классы. Остальные классы не менее важны, но они реже используются напрямую. Для начала рассмотрим классы Intent и View.

Класс View является основным строительным блоком для компонентов пользовательского интерфейса (UI), он определяет прямоугольную область экрана и отвечает за прорисовку и обработку событий. Является базовым классом для виджетов (GUI widgets), которые используются для создания интерактивных компонентов пользовательского интерфейса: кнопок, текстовых полей и т. д. А также является базовым классом для класса ViewGroup, который является невидимым контейнером для других контейнеров и виджетов, определяет свойства расположения компонентов пользовательского интерфейса. Интерфейс Android-приложения представляет собой иерархию UI компонентов (см. рис. 3.2), можно описать эту иерархию программно, но более простым и эффективным способом задать расположение элементов интерфейса является XML файл, который предоставляет удобную для восприятия структуру компоновки (layout file). Во время исполнения XML файл автоматически превращается в дерево соответствующих объектов. Подробнее о классе View, свойствах и методах: http://developer.android.com/reference/android/view/View.html.
Иерархия компонентов, определяющая компоновку интерфейса пользователя

Рис. 3.2. Иерархия компонентов, определяющая компоновку интерфейса пользователя

Объекты-экземпляры класса Intent используются для передачи сообщений между основными компонентами приложений. Известно, что три из четырех основных компонентов: активности, сервисы и приемники широковещательных сообщений, могут быть активированы с помощью сообщений, которые называются намерениями. Такие сообщения являются инструментом позднего связывания компонентов одного или нескольких приложений. Экземпляр класса Intent представляет собой структуру данных, содержащую описание операции, которая должна быть выполнена, и обычно используется для запуска активности или сервиса. В случае с приемниками широковещательных сообщений объект Intent содержит описание события, которое произошло или было объявлено.

Для каждого типа компонентов существуют свои механизмы передачи намерений.

Чтобы запустить активность или вызвать у работающей активности новое действие, необходимо передать объект-намерение в метод Context.startActivity() или Activity.startActivityForResult().

Чтобы запустить сервис или доставить новые инструкции работающему сервису, необходимо передать объект-намерение в метод Context.startService(). Также объект-намерение может быть передан в метод Context.bindService(), чтобы связать между собой вызывающий компонент и сервис.

Чтобы доставить объект-намерение всем заинтересованным приемникам широковещательных сообщений, необходимо передать его в любой из широковещательных методов: Context.sendOrderedBroadcast(), Context.sendStickyBroadcast(), Context.sendBroadcast().

В каждом случае система Android в ответ на намерение находит соответствующий компонент: активность, сервис или множество широковещательных приемников и запускает его если необходимо. В этой системе сообщений не случается накладок: сообщение-намерение, отправленное определенному компоненту, будет получено именно этим компонентом и никем другим.
Передача намерений (Intent)

Рис. 3.3. Передача намерений (Intent)

На рис. 3.3 можно увидеть как происходит передача намерений (Intent), в данном случае одна активность запускает другую. [6] Активность А создает намерение (Intent) с описанием действия и передает его в метод startActivity(). [7] Система Android проверяет все приложения на совпадение с намерением, когда совпадение найдено, [8] система запускает соответствующую активность, для чего вызывает метод onCreate() и передает в него объект-намерение Intent.

Подробнее о классе Intent:
http://developer.android.com/guide/components/intents-filters.html,
http://developer.android.com/reference/android/content/Intent.html

Пришло время более серьезно рассмотреть основные компоненты: активности, сервисы, контент-провайдеры, приемники широковещательных сообщений. В первую очередь нас будет интересовать жизненный цикл этих компонентов. Что же такое этот жизненный цикл? Жизненный цикл можно рассматривать, как процесс функционирования компонента: начиная с момента создания и запуска, включая активный и неактивный периоды работы, и, заканчивая уничтожением и освобождением ресурсов.

3.4.1 Активности (Activities)

Активность - окно, несущее графический интерфейс пользователя. Окно активности обычно занимает весь экран устройства, однако вполне возможно создавать полупрозрачные или плавающие диалоговые окна. Мобильные приложения обычно являются многооконными, т. е. содержат несколько активностей, по одной на каждое окно. Одна из активностей определяется как "главная", и именно ее пользователь видит при первом запуске приложения.

Каждый экран приложения является наследником класса Activity. Для создания активности необходимо создать класс-наследник класса Activity напрямую или через любого его потомка. В этом классе необходимо реализовать все методы, вызываемые системой для управления жизненным циклом активности. Таких методов семь:
onCreate() - метод, вызываемый системой при создании активности. В реализации метода необходимо инициализировать основные компоненты активности и в большинстве случаев вызвать метод setContentView() для подключения соответствующего XML-файла компоновки (layout file). После метода onCreate() всегда вызывается метод onStart().
onRestart() - метод, вызываемый системой при необходимости запустить приостановленную активность. После этого метода всегда вызывается метод onStart().
onStart() - метод, вызываемый системой непосредственно перед тем, как активность станет видимой для пользователя. После этого метода вызывается onResume().
onResume() - метод, вызываемый системой непосредственно перед тем, как активность начнет взаимодействовать с пользователем. После этого метода всегда вызывается onPause().
onPause() - метод, вызываемый системой при потере активностью фокуса. В этом методе необходимо фиксировать все изменения, которые должны быть сохранены за пределами текущей сессии. После этого метода вызывается onResume(), если активность вернется на передний план, или onStop(), если активность будет скрыта от пользователя.
onStop() - метод, вызываемый системой, когда активность становится невидимой для пользователя. После этого метода вызывается либо onRestart(), если активность возвращается к взаимодействию с пользователем, либо onDestroy(), если активность уничтожается.
onDestroy() - метод, вызываемый системой перед уничтожением активности. Этот метод вызывается либо когда активность завершается, либо когда система уничтожает активность, чтобы освободить ресурсы. Можно различать эти два сценария с помощью метода

isFinishing(). Это последний вызов, который может принять активность.
Жизненный цикл активности

Рис. 3.4. Жизненный цикл активности

(источник: http://developer.android.com/guide/components/activities.html).

При реализации вышеперечисленных методов первым делом всегда необходимо вызывать соответствующий метод предка.

Рассмотренные методы определяют жизненный цикл активности. На рис. 3.4 можно увидеть пути, по которым активность может переходить из одного состояния в другое. В прямоугольниках указаны методы, которые вызываются при смене состояний активности.

Фактически активность может существовать в одном из трех состояний:

Выполняется (running). Активность находится на переднем плане и удерживает фокус ввода. Если внимательно рассмотреть рис. 3.4 можно заметить, что в это состояние активность попадает после вызова метода onResume(). Пока активность находится в этом состоянии ее процесс не может быть уничтожен системой.

Приостановлена. Активность частично видима, однако фокус ввода потерян. В это состояние активность попадает после вызова метода onPause() (рис. 3.4). В этом состоянии активность поддерживается в "боевой готовности", т.е. в любой момент может получить фокус ввода и стать активной. Однако в этом состоянии процесс активности может быть уничтожен системой, в случае экстремальной нехватки памяти.

Остановлена. Активность полностью невидима. В это состояние активность попадает после вызова метода onStop()(рис. 3.4). В этом состоянии активность может быть "вызвана к жизни", она сохраняет все состояния и необходимую для восстановления информацию, однако процесс активности может быть уничтожен, если память понадобится для других целей.

3.4.2 Сервисы (Services)

Сервис (Service) является компонентом приложения, предназначенным для выполнения длительных операций в фоновом режиме. Существует два способа существования сервисов:

первый заключается в том, что сервис запущен (started) и работает самостоятельно в фоновом режиме, так он может работать неопределенно долго, пока не выполнит свою задачу;

второй заключается в том, что сервис привязан (bound) к некоторому компоненту или нескольким компонентам, в этом случае сервис предлагает интерфейс для взаимодействия с компонентом и работает пока привязан хотя бы к одному компоненту, как только связь со всеми компонентами разрывается сервис завершает свою работу.

Для создания сервиса необходимо создать класс-наследник класса Service напрямую или через любого его потомка. При этом в реализации класса необходимо переопределить (т. е. написать свою реализацию) некоторые методы, управляющие ключевыми аспектами жизненного цикла сервиса и обеспечивающие механизм связывания компонентов с сервисом, в соответствующем случае. Рассмотрим наиболее важные методы требующие реализации при создании сервиса.
onStartCommand() - метод, вызываемый системой, когда некоторый компонент, например активность, вызывает метод startService(). В этом случае сервис запускается и может работать

в фоновом режиме неопределенно долго, поэтому необходимо позаботиться об остановке сервиса, когда он выполнит свою работу. Для остановки сервиса используется метод stopSelf() в случае, когда сервис сам прекращает свою работу, или stopService() в случае, когда работу сервиса прекращает некоторый компонент. Нет необходимости писать реализацию метода onStartCommand(), если не предполагается самостоятельной работы сервиса (т. е. он будет работать только в связке с некоторыми компонентами).
onBind() - метод, вызываемый системой, когда некоторый компонент желает привязать к себе сервис и вызывает метод bindService(). Этот метод должен возвращать реализацию интерфейса IBinder, которая может быть использована компонентом-клиентом для взаимодействия с сервисом. Метод onBind() необходимо реализовать в любом случае, но, если не предполагается связывания сервиса с какими-либо компонентами, возвращаемое значение должно быть равным null.

Необходимо отметить, что сервис может быть запущен как самостоятельная единица, а впоследствии может быть привязан к некоторым компонентам. В этом случае в сервисе должны быть обязательно реализованы оба метода onStartCommand() и onBind().
onCreate() - метод, вызываемый системой, при первом обращении к сервису для выполнения первоначальных настроек. Этот метод вызывается до вызова методов onStartCommand() и/или onBind().
onDestroy() - метод, вызываемый системой, когда сервис либо выполнил все действия, для которых создавался, либо больше не связан ни с одним компонентом, т. е. его услуги больше не требуются. В реализации этого метода необходимо предусмотреть освобождение всех ресурсов, таких как потоки, зарегистрированные слушатели, приемники и т. д. Вызов этого метода является последними вызовом, который может получить сервис.

Жизненный цикл сервиса

Рис. 3.5. Жизненный цикл сервиса

(источник: http://developer.android.com/guide/components/services.html).

На рис. 3.5 показан жизненный цикл сервиса, левая диаграмма показывает жизненный цикл самостоятельного сервиса, правая - жизненный цикл сервиса, привязанного к некоторым компонентам. На рисунке хорошо видно, что жизненный цикл сервиса намного проще жизненного цикла активности. Однако для разработчика понимание того, как именно сервис создается, запускается и завершает свою работу, может оказаться даже более важным, т. к. сервис работает в фоновом режиме и пользователь может и не осознавать, что в некоторых случаях он имеет дело с работой сервисов.

Android принудительно останавливает работу сервисов только, когда ресурсов системы не хватает для активности, которая работает в данный момент на переднем плане. Приоритет работающих сервисов всегда выше, чем у приостановленных или полностью невидимых активностей, а если сервис привязан к выполняющейся активности, то его приоритет еще выше. С другой стороны, со временем приоритет самостоятельно работающего сервиса понижается и его шансы быть принудительно остановленным системой в случае нехватки ресурсов повышаются. В связи с этим имеет смысл проектировать сервис таким образом, чтобы через некоторое время он требовал у системы перезапуска. В случае если система все таки экстренно завершила работу сервиса, она перезапустит его как только освободятся ресурсы.

3.4.3 Контент-провайдеры (Content Providers)

Контент-провайдер управляет доступом к хранилищу данных. Для реализации провайдера в Android приложении должен быть создан набор классов в соответствии с манифестом приложения. Один из этих классов должен быть наследником класса ContentProvider, который обеспечивает интерфейс между контент-провайдером и другими приложениями. Основное назначение этого компонента приложения заключается в предоставлении другим приложениям доступа к данным, однако ничто не мешает в приложении иметь активность, которая позволит пользователю запрашивать и изменять данные, находящиеся под управлением контент-провайдера.

В мобильных приложениях контент-провайдеры необходимы в следующих случаях:

приложение предоставляет сложные данные или файлы другим приложениям;
приложение позволяет пользователям копировать сложные данные в другие приложения;
приложение предоставляет специальные варианты поиска, используя поисковую платформу (framework).

Если приложение требует использования контент-провайдера, необходимо выполнить несколько этапов для создания этого компонента:

1. Проектирование способа хранения данных. Данные, с которыми работают контент-провайдеры, могут быть организованы двумя способами:

Данные представлены файлом, например, фотографии, аудио или видео. В этом случае необходимо хранить данные в собственной области памяти приложения. В ответ на запрос от другого приложения, провайдер может возвращать ссылку на файл.

Данные представлены некоторой структурой, например, таблица, массив. В этом случае необходимо хранить данные в табличной форме. Строка таблицы представляет собой некоторую сущность, например, сотрудник или товар. А столбец - некоторое свойство этой сущности, например, имя сотрудника или цена товара. В системе Android общий способ хранения подобных данных - база данных SQLite, но можно использовать любой способ постоянного хранения.

Больше о хранении данных в Android можно узнать по ссылке: http://developer.android.com/guide/topics/providers/content-provider-creating.html#DataStorage

2. Создание класса-наследника от класса ContentProvider напрямую или через любого его потомка. При этом в реализации класса необходимо переопределить (т. е. написать свою реализацию) обязательные методы.

query() - метод, извлекающий данные из провайдера, в качестве аргументов получает таблицу, строки и столбцы, а также порядок сортировки результата, возвращает объект типа Cursor.

insert() - метод, добавляющий новую строку, в качестве аргументов получает таблицу, и значения элементов строки, возвращает URI добавленной строки.

update() - метод, обновляющий существующие строки, в качестве аргументов получает таблицу, строки для обновления и новые значения элементов строк, возвращает количество обновленных строк.

delete() - метод, удаляющий строки, в качестве аргументов принимает таблицу и строки для удаления, возвращает количество удаленных строк.

getType() - метод, возвращающий String в формате MIME, который описывает тип данных, соответствующий URI. Подробнее: http://developer.android.com/guide/topics/providers/content-provider-creating.html#MIMETypes
onCreate() - метод, вызываемый системой, сразу после создания провайдера, включает инициализацию провайдера. Стоит отметить, что провайдер не создается до тех пор, пока объект ContentResolver не попытается получить к нему доступ.

Созданный контент-провайдер управляет доступом к структурированным данным, выполняя обработку запросов от других приложений. Все запросы, в конечном итоге, вызывают объект ContentResolver, который в свою очередь вызывает подходящий метод объекта ContentProvider для получения доступа. Все вышеперечисленные методы, кроме onCreate(), вызываются приложением-клиентом. И все эти методы имеют такую же сигнатуру, как одноименные методы класса ContentResolver. Подробнее о классе ContentProvider: http://developer.android.com/guide/topics/providers/content-provider-creating.html#ContentProvider

3. Определение строки авторизации провайдера, URI для его строк и имен столбцов. Если от провайдера требуется управление намерениями, необходимо определить действия намерений, внешние данные и флаги. Также необходимо определить разрешения, которые необходимы приложениям для доступа к данным провайдера. Все эти значения необходимо определить как константы в отдельном классе, этот класс в последствии можно предоставить другим разработчикам.

Подробнее об URI:

http://developer.android.com/guide/topics/providers/content-provider-creating.html#ContentURI

Подробнее о намерениях:

http://developer.android.com/guide/topics/providers/content-provider-creating.html#Intents
3.4.4 Приемники широковещательных сообщений (Broadcast Receivers)

Каждый широковещательный приемник является наследником класса BroadcastReceiver. Этот класс рассчитан на получение объектов-намерений отправленных методом sendBroadcast().

Можно выделить две разновидности широковещательных сообщений:

Нормальные широковещательные сообщения передаются с помощью Context.sendBroadcast в асинхронном режиме. Все приемники срабатывают в неопределенном порядке, часто в одно и то же время.

Направленные широковещательные сообщения передаются с помощью Context.sendOrderedBroadcast только одному приемнику в один момент времени. Как только приемник сработает, он может передать сообщение следующему приемнику, а может прервать вещание так, что больше ни один приемник это сообщение не получит.

Даже в случае нормального широковещания могут сложиться ситуации, в которых система будет передавать сообщения только одному приемнику в один момент времени. Особенно это актуально для приемников, которые требуют создания процессов, чтобы не перегружать систему новыми процессами. Однако в этом случае ни один приемник не может прервать широковещание.

Объект типа BroadcastReceiver действителен только во время вызова метода onRecieve(), как только метод выполнен, система завершает работу объекта и больше не активирует его.

3.5 Манифест приложения

Корневой каталог каждого приложения под Android должен содержать файл AndroidManifest.xml (в точности с таким названием). Манифест приложения содержит всю необходимую информацию, используемую системой для запуска и выполнения приложения. Основная информация, содержащаяся в манифесте:

Имя Java пакета приложения, которое используется как уникальный идентификатор приложения.

Описание компонентов приложения: активностей, сервисов, приемников широковещательных сообщений и контент-провайдеров, которые составляют приложение. Для каждого компонента приложения определено имя соответствующего класса и объявлены их основные свойства (например, с какими сообщениями-намерениями они могут работать). Эта информация позволяет системе Android узнать какие компоненты и при каких условиях могут быть запущены.

Определение процессов, в которых будут выполняться компоненты приложения.

Объявление полномочий, которыми должно обладать приложение для доступа к защищенным частям API и взаимодействия с другими приложениями.

Объявление полномочий, которыми должны обладать другие приложения для взаимодействия с компонентами данного.

Список вспомогательных классов, которые предоставляют информацию о ходе выполнения приложения. Эти объявления содержатся в манифесте пока идет разработка и отладка приложения, перед публикацией приложения они удаляются.

Определение минимального уровня Android API для приложения.

Список библиотек связанных с приложением.

В файле манифеста только два элемента: <manifest> и <application> являются обязательными и при этом встречаются ровно по одному разу. Остальные элементы могут встречаться несколько раз или не появляться совсем, в этом случае манифест определяет пустое приложение.

Следующий листинг демонстрирует общую структуру файла манифеста.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
  <uses-permission />
  <permission />
  <permission-tree />
  <permission-group />
  <instrumentation />
  <uses-sdk />
  <uses-configuration />
  <uses-feature />
  <support-screens />
  <compatible-screens />
  <supports-gl-texture />
```

```
  <application>
    <activity>
      <intent-filter>
        <action />
        <category />
        <data />
      </intent-filter>
      <meta-data />
    </activity>
    <activity-alias>
      <intent-filter> … </intent-filter>
      <meta-data />
    </activity-alias>
    <service>
      <intent-filter> … </intent-filter>
      <meta-data />
    </service>
    <receiver>
      <intent-filter> … </intent-filter>
      <meta-data />
    </receiver>
    <provider>
      <grant-uri-permission />
      <meta-data />
      <path-permission />
    </provider>
    <uses-library />
  </application>
</manifest>
```

Листинг 3.1. Структура файла AndroidManifest.xml

В манифесте элементы одного уровня, такие как <activity>, <service>, <receiver>, <provider>, могут следовать друг за другом в любой последовательности. Элемент <activity-alias> является исключением из этого правила, он должен следовать за соответствующей активностью.

Более предметно разговор о файле манифеста и его основных элементах пойдет в лабораторных работах.

Подробности: http://developer.android.com/guide/topics/manifest/manifest-intro.html

3.6 Ресурсы

При разработке мобильных приложений необходимо выработать привычку отделять ресурсы приложения от кода. К ресурсам приложения могут относиться: изображения, строки, цвета, компоновки элементов пользовательского интерфейса (layout) и т. д. Отделение ресурсов от кода позволяет использовать альтернативные ресурсы для различных конфигураций устройств: язык, разрешение экрана и т. д. Для обеспечения совместимости с различными конфигурациями, ресурсы необходимо сгруппировать в директории по типу ресурсов и конфигурации устройства, полученные директории поместить в папку res/.

Для любого типа ресурсов можно определить две группы. Первая определяет ресурсы, которые будут использоваться независимо от конфигурации устройства или в том случае, когда под конфигурацию нет подходящих альтернативных ресурсов. Эта группа называется ресурсы по умолчанию (default). Вторая группа определяет ресурсы, подходящие для определенной конфигурации устройства, размещается в директории с названием, обозначающим данную конфигурацию. Такие ресурсы называются альтернативными.

а) используется компоновка по умолчанию (приложение не содержит альтернативы) б) каждое устройство использует соответствующую компоновку
Рис. 3.6. Использование ресурсов

Каждый тип ресурсов необходимо размещать в специальной поддиректории папки res/. Рассмотрим основные из этих поддиректорий:
animator/ - содержит XML файлы, которые определяют свойства анимации;
anim/ - содержит XML файлы, которые определяют анимацию преобразований;
color/ - содержит XML файлы, которые определяют списки цветов;
drawable/ - содержит графические файлы или XML файлы, которые компилируются в графические ресурсы;
layout/ - содержит XML файлы, которые определяют компоновку элементов пользовательского интерфейса;
menu/ - содержит XML файлы, которые определяют все меню приложения;
values/ - содержит XML файлы, которые определяют простые значения, таких ресурсов как, строки, числа, цвета.

Следует отметить, что файлы ресурсов нельзя размещать в папку res/ напрямую, они обязательно должны размещаться в соответствующем каталоге, иначе будет выдана ошибка компиляции.

Все ресурсы, которые содержатся в рассмотренных поддиректориях являются ресурсами по умолчанию. Понятно, что различные типы устройств могут требовать различных типов ресурсов. Например, для устройств с разными размерами экрана компоновки элементов пользовательского интерфейса должны отличаться. Рис 3.6 показывает варианты внешнего вида приложения с использованием только компоновки по умолчанию (а) и с использованием альтернативных компоновок (б). Даже на схеме понятно, что при правильном подходе приложение, изменяющее свой внешний вид в зависимости от размера экрана привлекательнее, чем остающееся неизменным.

Чтобы определить зависимые от конфигурации альтернативы для множества ресурсов:

необходимо создать директорию в каталоге res/, присвоить этой директории имя в следующей форме: имя_ресурса-спецификатор_конфигурации, где
имя_ресурса - имя директории, соответствующего ресурса по умолчанию (см. выше);
спецификатор_конфигурации - имя, определяющее конфигурацию, для которой используются данные ресурсы. Полный список доступных спецификаторов: http://developer.android.com/guide/topics/resources/providing-resources.html
необходимо сохранить ресурсы в новой директории, файл ресурсов должен называться в точности так же, как соответствующий файл ресурсов по умолчанию.

Например, если компоновка элементов пользовательского интерфейса сохранена, как ресурс

по умолчанию, в папке res/layout/, можно (скорее даже нужно) определить альтернативную компоновку элементов пользовательского интерфейса, соответствующую горизонтальной (альбомной) ориентации экрана смартфона и сохранить ее в папке res/layout-land/. Android автоматически определит подходящую компоновку, сверяя текущее состояние устройства с именами папок в каталоге /res.

Все ресурсы после определения могут быть доступны по ссылке на их ID, которые определены в автоматически генерируемом классе R. Для каждого типа ресурсов в R классе существует подкласс, например, R.drawable для всех графических ресурсов. ID ресурса всегда имеет две составляющие:

тип ресурса - все ресурсы группируются по типам, например, string, drawable, layout;
имя ресурса - либо имя файла без расширения, либо значение атрибута android:name в XML файле для простого значения.

Получить доступ к ресурсу можно двумя способами:

в коде: можно использовать выражения вида R.тип_ресурса.имя_ресурса, например, R.string.hello;
в XML: используется специальный XML синтаксис, который соответствует ID определенному в R классе, например, @string/hello.