1. Основные компоненты андроид приложения.

● Activity (Операция) один экран с пользовательским интерфейсом, например: экран со списком контактов, экран с настройками

● Service (Служба) не имеет пользовательского интерфейса, работает в фоновом режиме неограниченное время, например: воспроизведение музыки, получение данных по сети

● Broadcast Receiver (Приемник широковещательных сообщений) нужен для реакции на события, происходящие в системе вне приложения, например: экран выключился, аккумулятор разряжен, сделан фотоснимок

● Content Provider (Поставщик контента) Посредством поставщика контента другие приложения могут запрашивать или даже изменять данные, например: для запроса списка контактов

2. Что такое Activity? Этапы реализации.

отвечает за окно, которое приложение использует для отрисовки и получения событий от системы

3. Типы View.

1. UI Components( TextView EditText ImageView Button)

2. Container Views( LinearLayout RelativeLayout FrameLayout ScrollView ConstraintLayout)

4.Основные атрибуты объектов в xml-файлах и их значения. Какие из них являются

обязательными?

Атрибуты объектов views позволяют определить положение, размер и другие свойства объектов view. У каждого объекта view должны быть заданы ширина и высота.

У каждого объекта view много свойств.

Padding — задает пространство внутри границ view

Margin — задает пространство вокруг view за его границами

5. Типы макетов (layout). Принцип выбора макета при реализации.

● LinearLayout – Объекты располагаются в одну колонку или строку

● RelativeLayout – Можно указать расположение объектов относительно друг друга

● FrameLayout – Содержит единственный дочерний объект View

● ScrollView – Позволяет прокручивать объекты, находящиеся внутри

● ConstraintLayout – Позволяет более гибко располагать объекты

6. Для чего нужен AndroidManifest.xml? Что обязательно указывается в этом файле?

В этом файле, который должен находиться в корневой папке приложения, должны быть объявлены все компоненты приложения

Манифест - это набор правил, по которым работает приложение. Файл манифеста находится в корневой папке — AndroidManifest.xml - и содержит важную информацию, без которой система не сможет запустить приложение.

7. Класс R. Примеры использования.

● Класс R генерируется во время компиляции – Создает константы для динамической идентификации содержимого папки res

● SetContentView – размещает объекты на экране приложения в соответствии с описанием в xml-файле

● findViewById – находит view по его id, указанному в xml-файле и размещает его в соответствующем объекте java.

8. Типы ресурсов и их расположение в проекте, способы обращения к ресурсам.

Цвета res/values/colors/

Строки res/values/strings.xml

Меню res/menu/menu.xml

Изображения res/drawable/

и тд.

Обращение к ресурсам

● setContentView(R.layout.layout\_file\_name)

● findViewById(R.id.some\_id)

● getString(R.string.string\_name)

9. Поддержка разных платформ, языков и экранов.

Поддержка разных платформ

minSDK — минимальная версия SDK, на которой может запускаться приложение

targetSDK — версия SDK, на которой тестируется приложение

Поддержка языков

● Создание директории для другого языка в формате dirname-country

● Создание одноименных файлов ресурсов с ресурсами для другого языка

Предоставить альтернативные ресурсы, которые будут поддерживать определённые конфигурации устройств (например, устройства с различными языками или разными размерами экрана).

10. Основные этапы создания меню.

● Создать .xml файл для описания элементов меню

● Установить атрибуты

● Инициализировать меню в .java файле с помощью OnCreateOptionsMenu

● Обработать нажатия на элементы меню с помощью OnItemClicked

11. Для чего нужны разрешения? Примеры использования. Принцип выбора разрешений,

которые необходимо будет потребовать от пользователя.

Permissions

● Каждое приложение на андроид работает в своем собственном экземпляре (instance) среды выполнения (runtime) – чтобы ни одно приложение не могло выполнять действия, которые могут повлиять на другие приложения, систему или пользователя

● => доступ к интернету, контактам, местоположению, .. требует разрешения

● Нужно выбирать минимум разрешений

● Добавление разрешения в AndroidManifest.xml

14. Средства для реализации списков (конечных и псевдо-бесконечных), их отличия

ScrollView — прокрутка экрана  
Когда View не умещаются в рамках одного экрана их можно разместить в прокручивающемся контейнере ScrollView. Проблема в том, что все элементы, в том числе которые не видны в данный момент, существуют в памяти.

Основные недостатки ListView  
Сложно добавить анимации и декорации элементам списка,  
Компонент ListView отвечает не только за переиспользование View, но и за порядок и структуру списка. Это значит, что реализовать горизонтальный список — проблематично,

Основные отличия RecyclerView от ListView  
RecyclerView отвечает только за переиспользование View, способ отображения данных задается с помощью отдельного менеджера — LayoutManager   
За анимации отдельных элементов при изменении, добавлении, удалении отвечает отдельный компонент ItemAnimator  
Принудительно используется шаблон ViewHolder. Это позволяет оптимизировать производительность за счет сохранения ссылок на View в контейнере.

15. Схема работы RecyclerView.

Схематично работу RecyclerView можно представить следующим образом. На экране отображаются видимые элементы списка. Когда при прокрутке списка верхний элемент уходит за пределы экрана и становится невидимым, его содержимое очищается. При этом сам "чистый" элемент помещается вниз экрана и заполняется новыми данными, иными словами переиспользуется,

При использовании RecyclerView в приложении требуется создание производных типов следующих классов:

* **RecyclerView.Adapter** — Предоставляет привязку из набора данных вашего приложения (относящегося к вашему приложению) к представлениям элементов, отображаемым в RecyclerView.
* **RecyclerView.ViewHolder** — Кэширует ссылки на представления в файле макета элемента, чтобы поиск ресурсов не повторялся без необходимости.
* **RecyclerView.LayoutManager** — Размещает элементы в RecyclerView.

16. Способы переходов между экранами приложения и между приложениями. Передача

данных при переходах.

– Explicit intents (явные объекты intent) ● для запуска компонента того же приложения

– Implicit intents (неявные объекты intent) ● для запуска компонента из другого приложения

Переключение между экранами

1. Создание нового класса Activity и описания layout File → new → Activity → EmptyActivity

2. Указать в манифесте родительскую Activity для вновь созданной Activity с помощью тэега meta-data

3. Запустить новую Activity

4. Передача данных в запускаемую Activity

17. Жизненный цикл приложения: состояния и методы, которые вызываются при

переходах между состояниями. Сценарии использования приложения пользователем,

при которых приложение будет переходить из одного состояния в другое.



Когда приложение оказывается в разных состояниях

● Active – в фокусе, активно (видно пользователю)

● Visible – частично скрыто (например, появилось диалоговое окно поверх основной Activity) ● Background – приложение в фоновом режиме (пользователю видно другое приложение)

18. //Способы сохранения данных при повороте экрана и сворачивании приложения.

См 19 1 и 2

19. Способы сохранения данных в зависимости от типа сохраняемых данных и времени

их хранения.

1. onSavedInstanceState

– Сохраняет данные в Bundle (пары ключзначение)

– Используется для сохранения данных в процессе работы приложения:

● Для восстановления после поворота экрана

● Для восстановления, если приложение было убито системой из-за нехватки памяти

2. SharedPreferences

– Сохраняет данные в указанный файл (пары ключ-значение)

– Используется для сохранения данных навсегда, вплоть до удаления приложения с телефона, например:

● Для сохранения адреса последней открытой вебстраницы

3. База данных.

– В Android можно сохранить данные в SQLite; например, заметки, т.к.:

● Их может быть несколько

● Их число меняется

● Каждая заметка состоит из заголовка и описания, то есть содержит больше одного значения

– Используется для сохранения данных вплоть до удаления приложения

4. Internal / External Storage. (на телефон / на SD-карту)

– Используется для сохранения данных большого размера

– Используется для сохранения данных вплоть до удаления приложения

5. Сервер (облако)

– Используется для сохранения данных, для которых предполагается доступ с разных устройств

– Используется для сохранения данных с разных устройств без потери даже в случае удаления приложения с устройства

20. Класс Fragment. Когда удобно использовать фрагменты?

● Фрагмент — часть Activity (или что-то вроде вложенной Activity), которая:

– Имеет свой жизненный цикл

– Может добавляться и удаляться в процессе работы приложения

21. Основные этапы создания настроек. Средства для обновления состояния приложения

при изменении настроек и проверки допустимости введенных значений. Отображение

в настройках текущего выбранного значения или другого описания.

1. Создать отдельную Activity для настроек

– Создать Activity (xml и java файлы)

– Создать пункт меню «Настройки» в ActionBar

– По нажатии на пункт меню «Настройки» открывать созданную Activity

2. Создать PreferenceFragment

– Добавить зависимость в build.gradle

– Создать фрагмент для настроек

– Добавить стиль в values/styles.xml

– Подключить фрагмент к Activity, созданной для настроек

Реагирование на изменения в настройках

● Главная Activity дожна реализовывать OnSharedPreferenceChangeListener

● В методе onCreate надо его зарегистировать, а в методе onDestroy — наоборот

setSummary Используется для отображения текущего значения в настройках

onPreferenceChange Вызывается до сохранения значения в настроках (до onSharedPreferenceChanged) Используется для проверки допустимости введенного значения в EditPreference

**22. Средства для реализации хранения данных в базе. Основные этапы реализации.**

23. Что для системы android является внутренним и внешним хранилищем? Выбор

хранилища для установки приложения. Использование разрешений.

● Внутреннее хранилище – Сохраненные файлы доступны только приложению – Удаляются при удалении приложения

● Внешнее хранилище – Сохраненные файлы доступны всем – Удаляются при удалении приложения при условии, что файлы лежат в папке, полученной методом getExternalFilesDir()

Варианты installLocation:

– preferExternal - приложение устанавливается во внешнюю память, если она доступна и не заполнена

– auto — предпочтительна установка во внутренне хранилище, пользователь может перемещать приложение между внутренним и внешним хранилищем

Внешнее хранилище (чтение и запись)

1. Указать разрешение в манифесте:

● Для чтения (требуется не везде): – READ\_EXTERNAL\_STORAGE

● Для записи: – WRITE\_EXTERNAL\_STORAGE

24. Для чего нужны поставщики контента (content providers)? Основные этапы создания.

Используются для:

– Обеспечения доступа к данным другим приложениям

– Отправки данных в виджеты

– Загрузки данных в UI (с помощью CursorLoader)

– Выдачи своих подсказок при поиске (с помощью AbstractThreadedSyncAdapter)

1) Создать класс ContentProvider

2) Реализовать методы: – onCreate – getType – query – insert - для обработки запросов – delete – update

3) Создать UriMatcher

4) Указать в манифесте компоненту для поставщика и требуемые разрешения

25. Для чего нужны поставщики контента (content providers)? Способы задания

допустимых URI-запросов и сопоставления с входящим запросом.

См 24.

Сопоставление URI контента с шаблоном запроса

Выполняется с помощью класса UriMatcher:

\* - строка любой длины с любыми символами

# - строка любой длины с цифрами

26. Для чего нужны поставщики контента (content providers)? Использование разрешений.

Примеры доступных поставщиков в системе android.

см.24

Можно добавить разрешение на чтение или запись (если требуется)

Поставщики контента в системе Android ● AlarmClock ● Contacts ● CalendarContract ● MediaStore ● Settings ● Telephony ● UserDictionary ● VoicemailContract

27. Сервисы (services): область применения, типы, этапы реализации.

● Представляет собой компонент, который работает в фоновом режиме и выполняет длительные операции, связанные с работой удаленных процессов

● Не имеет пользовательского интерфейса

● Примеры: – Воспроизведение музыки в фоновом режиме (пользователь может в это время заходить в другие приложения) – Получение данных по сети (без блокирования взаимодействия пользователя с Activity)

● Может быть запущена из приложения (из Activity)

Создание сервиса

● Объявление в AndroidManifest.xml

● Реализация класса сервиса, расширяющего класс Service или IntentService

● Запуск и остановка сервиса – Создание Intent – Запуск сервиса – Остановка сервиса

28. Приемники широковещательных сообщений (broadcast receivers): область применения,

использование разрешений, этапы реализации. Примеры событий, распространяемых

системой андроид.

● Представляет собой компонент, который реагирует на объявления распространяемые по всей системе – Объявления могут посылаться как системой Android, так и приложением

● Примеры объявлений от системы: – Экран был выключен – Аккумулятор раряжен – Был сделан фотоснимок

● Приложение может послать объявления, например, о том, что были загружены данные и их можно использовать

Создание BroadcastReceiver

● Объявление в AndroidManifest.xml

● Реализация класса Создание BroadcastReceiver

● Запуск и остановка

● Использование разрешений