МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет
Кафедра программного обеспечения
информационных систем и технологий

Электронный учебно-методический комплекс

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

для специальности:
1-40 05 01 «Информационные системы и технологии»

Составитель: Тетерюкова И.О.

Минск БНТУ 2024

СОДЕРЖАНИЕ

1. ПРОГРАММА ДИСЦИПЛИНЫ

Программа по учебной дисциплине «Разработка приложений для мобильных устройств» разработана для специальности 1-40 05 01 «Информационные системы и технологии» специализация 1-40 05 01-04 «Информационные системы и технологии (в обработке и представлении информации)».

Целью изучения дисциплины является изучение основных концепций, принципов и правил создания мобильных приложений для платформы Android, а также изучение методов создания и поддержки адаптивного дизайна.

Основными задачами дисциплины являются приобретение студентами навыков формирования структуры приложений, обработки ввода и сохранения данных приложения, настраивания внешнего вида и реализации дизайна в мобильных приложениях с ориентаций на пользователя, создания анимационных эффектов.

2. ТЕОРИТИЧЕСКИЙ РАЗДЕЛ

2.1 Среда разработки. Структура проекта и модули.

Для разработки приложений будет использоваться интегрированная среда разработки Android Studio.

Достоинства Android Studio:

- ✓ Встроенный Android Virtual Device (эмулятор), который сразу позволяет запустить и тестировать приложение
- ✓ Разделенный режим редактирования .xml-файлов. Это позволяет вносить изменения в экраны приложения и сразу же видеть результат, что ускоряет процесс разработки и упрощает верстку экранов.
- ✓ Среда разработки поддерживает работу с различными языками программирования, среди которых C/C++, Java, Kotlin
- ✓ Удобство работы в редакторе кода
- ✓ Рефакторинг кода
- ✓ Внушительная библиотека с готовыми компонентами и шаблонами необходимыми для разработки
- ✓ Доступность для трех основных операционных систем - Windows, Mac OS и Linux.
- ✓ Разработка приложений для самой последней версии операционной системы — Android N
- ✓ Наличие специально созданного руководства по использованию Android Studio, которое можно найти на официальном сайте, для начинающих разработчиков

Для установки Android Studio необходимо зайти на сайт developer.android.com и скачать установочный файл.

В проекте может храниться как один модуль, так и несколько. Каждый модуль в проекте представляет собой отдельное Android-приложение.

Для создания нового модуля нужно выбрать пункт File, затем навести на New и кликнуть New Module. Затем выбрать тип создаваемого модуля, это будет Phone & Tablet Module и внести параметры для модуля – имя приложения, имя модуля или система имен, если модулей в проекте предполагается несколько. Затем выбирать EmptyActivity, а в следующем окне нажать «Finish». Модуль создан.

Если в модуле больше нет необходимости его можно удалить. Для это следует выбрать пункт File и в выпадающем списке строку Project Structure, которая откроет доступ к структуре проекта. Для удаления модуля требуется открыть

вкладку Modules и выбрать Remove Module. После подтверждения во всплывающем диалоговом окне модуль будет удален.

Обратите внимание на два файла, которые содержит модуль. Это MainActivity.java и activity_main.xml. Первый содержит исполняемый код программы на языке Java. Второй файл содержит описание как должно выглядеть приложение на экране устройства, то есть это визуальная часть. Он содержит переключатель между другими режимами отображения файла, режим отображения экрана, выбор ориентации экрана, варианты отображения на разных экранах, список View компонентов - стандартные элементы, которые используются при создании приложений с пользовательским графическим интерфейсом и дерево компонентов.

2.2 Компоненты экрана. Обработчик событий для компонентов экрана.

Android-приложение содержит в себе окна, которые называются Activity. Каждое из них содержит различный набор View-компонентов таких как TextView, Button, Switch и других.

У каждого компонента есть ID – это id компонента, по нему можно обращаться к компоненту в коде, и атрибуты, которые задают параметры для каждого отдельного вида View-компонента.
Изменять атрибуты компонентов можно двумя способами: через окно атрибутов или непосредственно через код.

Для того, чтобы на экране были размещены различные View-компоненты их необходимо разместить в специальном контейнере - ViewGroup.
Существует несколько видов ViewGroup: LinearLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout и другие. Они различаются тем, каким образом они располагают компоненты внутри себя.

Суть вкладывания компонентов, таких как, например, TextView или Button в контейнер ConstraintLayout заключается в их привязке к тем или иным частям экрана, что будет задавать их положение на экране. В контейнере ConstraintLayout есть возможность организовать горизонтальную и вертикальную привязку. Однако привязывать компоненты можно не только к границам экрана, но и к другим View-компонентам. Также привязка может быть комбинированной, то есть одна из границ будет привязана к границе экрана, а вторая к уже существующему компоненту.

Контейнер LinearLayout привязки не использует. Он отображает компоненты, которые выстраивались в зависимости от ориентации в вертикальный или горизонтальный ряд. Ориентация задается в атрибуте orientation (vertical или horizontal). В LinearLayout по умолчанию установлена вертикальная

ориентация – все добавленные компоненты выстраиваются друг за другом сверху вниз.

Контейнер TableLayout отображает компоненты в виде таблицы, расположенными по столбцам и строкам. Он уникален тем, что состоит из других контейнеров, которые называются TableRow. Каждый TableRow состоит из столбцов, которые формируются View-компонентами.

В контейнере RelativeLayout каждый компонент может быть расположен определенным образом относительно уже существующего указанного View:

- ✓ Расположение слева, справа, сверху или снизу указанного компонента.
- ✓ Выравнивание по левому или правому, по верхнему или нижнему краю указанного компонента.
- ✓ Выравнивание по левому или правому, по верхнему или нижнему краю родительского компонента.
- ✓ Выравнивание слева по центру, справа по центу, сверху по центру, снизу по центру относительно родительского компонента.

Главным плюсом контейнера RelativeLayout является одинаковое отображение элементов на разных устройствах – масштабирование и привязки сохраняются. Минусом является кропотливая работа над расположением каждого компонента и работа с кодом.

Особенность контейнера AbsoluteLayout заключается в абсолютном позиционировании компонентов на экране, то есть он не требует ни дополнительных привязок, ни создания дополнительных контейнеров внутри себя. Однако его выбор рекомендуется только в том случае, когда приложение будет использоваться только на какой-то одной модели устройств с одинаковым разрешением экрана. В противном случае придется для каждого разрешения менять расположение компонентов, что значительно замедлит разработку.

Создаем в проекте новый .xml-файл и выбираем необходимый вид контейнера, в который будут вкладываться компоненты. В качестве компонент для примера возьмем TextView и две кнопки. Изменять атрибуты можно двумя способами: через окно атрибутов или непосредственно через код. Рассмотрим работу с компонентами с помощью кода и с изменением только атрибута id, так как изменение других атрибутов будет завесить от вида используемого контейнера.

Id компанента стоит задавать понятным. Пусть для TextView это будет textViewOut. А для кнопок с текстом OK и Cancel, id изменим соответственно их предназначению - btnOk и btnCancel.

Приступим к написанию самого обработчика событий. Пусть при нажатии кнопки OK TextView отображает «Нажата кнопка Ok», а при нажатии кнопки Cancel — отображает «Нажата кнопка Cancel».

Далее работаем с кодом в файле MainActivity.java.
Во-первых, необходимо создать переменные, которые будут отвечать за ранее созданные компоненты. Объявлять переменные необходимо за пределами метода onClick, чтобы к ним можно было обратиться из любого метода.
Во-вторых, для привязки непосредственно View-компонента к переменной необходимо использовать метод findViewById. Этот метод возвращает View-компонент по его id, именно поэтому было важно установить понятный id.

Метод выглядит следующим образом:

```
textViewOut = (TextView) findViewById(R.id.textViewOut);
```

Теперь переменной textViewOut соответствует ранее созданный TextView.

Аналогично сделаем и для кнопок:

```
btnOk = (Button) findViewById(R.id.btnOk);
btnCancel = (Button) findViewById(R.id.btnCancel);
```

Для того, чтобы фиксировать то, когда кнопка нажата, существует метод setOnClickListener. Нужно создать объект, который будет передаваться в этот метод в момент нажатия. Создаем его в том же методе onCreate, где и писали код до этого.

```
View.OnClickListener ocBntOk = new View.OnClickListener()
```

После этого Android Studio сама создаст его тело. Внутри тела необходимо задать изменение компонента TextView. Это реализуется следующим образом:

```
textViewOut.setText("Нажата кнопка OK");
```

Чтобы обработчик нажатия запускался непосредственно при нажатии кнопки, необходимо присвоить его значение требуемой кнопке при помощи метода setOnClickListener:

```
btnOk.setOnClickListener(ocBntOk);
```

Аналогичным образом создадим обработчик нажатия для кнопки Cancel и присвоим его значение с помощью метода setOnClickListener.

При запуске приложения, текст компонента TextView будет соответствовать тому, который задавался при добавлении строки в файл strings.xml. Но стоит нажать на какую-либо из кнопок, он меняется на тот, который был указан в обработчике нажатия.

Итак, краткий механизм обработки событий. Сам компонент, в нашем случае кнопка, свои нажатия обрабатывать не умеет, для этого ей нужен обработчик, который присваивается с помощью метода setOnClickListener. При нажатии на кнопку, обработчик выполняет свой код из метода onClick.
Таким образом, можно выделить следующие этапы: создание обработчика событий, заполнение метода onClick, присвоение обработчика кнопке.

Существует еще один способ реализовать обработчик событий. Для него нужно вернуться к .xml-файлу и указать среди атрибутов кнопки следующее:

android:onClick="имя_метода_в_классе_MainActivity"

То есть для самой кнопки прописывается имя метода, который будет вызываться при нажатии на нее. А в самом методе просто указать действия, которые должны будут выполняться при нажатии на кнопку.
При такой реализации можно не создавать переменные для связи их с View, разве что для TextView, так как в нашем примере он изменяется в зависимости от того, какая кнопка нажата.

2.3 Использование ресурсов приложения. Логи приложения. Всплывающие сообщения.

Android, содержит и использует некоторые ресурсы. В роли таких ресурсов могут выступать:

- ✓ layout-файлы, хранятся в папке res/layout.
- ✓ Различные изображения, которые отображаются в приложении в качестве заднего фона или же просто картинки, хранятся в папке res/drawable.
- ✓ Ресурсами могут быть константы различных типов, например, типа String и Color.

Константы различных типов хранятся в папке res/values, в ней можно обнаружить три файла с расширением .xml.

В файле strings.xml хранятся элементы типа String. Среди них имя приложения и три элемента, которые используются, для отображения текста на экране приложения. Для использования этих элементов нужно в layout-файле найти компонент, к которому планируется привязать компонент, и либо в коде .xml-файла указать в качестве текста имя элемента, например, @string/btnOk, либо в режиме дизайна найти атрибут text и там тоже указать имя элемента.

В файле colors.xml хранятся цвета, которые используются в приложении. Эти цвета можно указывать в качестве фона layout-ов для компонентов экрана. В этом файле по умолчанию хранится 3 цвета, значение которых указано в 16-тиричной системе.
Для добавления нового цвета его значение можно указать несколькими способами.
Первый — это узнать его код в 16-тиричной системе и указать его в свойствах.
Второй — это кликнуть на квадрат с цветом рядом с номером строки кода, откроется редактор цвета. В нем можно выбрать сам цвет на палитре, установить его прозрачность, или же указать напрямую его значение в одной из систем: RGB или 16-тиричная.

Для использования и добавления пользовательских строк и цветов можно создать собственный файл и хранить в нем как строки, так и цвета. Это удобно, когда определенный набор констант должен соответствовать набору компонентов на экране. Для создания такого файла нужно правой кнопкой мыши нажать по папке values и выбрать New -> Values Resource File. В открывшемся диалоговом окне указать имя файла.

В новосозданном файле можно добавить строки для TextView - те строки, которые оповещают о нажатии той или иной кнопки: «Нажата кнопка OK», «Нажата кнопка Cancel». Для этого в файле нужно прописать следующий код:

```
<string name="textOk">Нажата кнопка OK</string>
<string name="textCancel">Нажата кнопка Cancel</string>
```

Также сюда можно добавить цвет, в который будут окрашивать кнопки при нажатии на них. Например, розовый, добавляется с помощью следующего кода:

```
<color name="colorPink">#60ED2DDA</color>
```

После добавления ресурсов необходимо обращаться к ним в коде программы. Для этого нужно перейти в файл MainActiviry.java и заменить в нем строки, записанные напрямую, на ссылки на строки в нашем файле. В скобках нужно указать следующий код:

```
R.string.textOk
R.string.textCancel
```

Для того, чтобы изменять цвет кнопки при ее нажатии в этом же файле нужно добавить переменную кнопки и присвоить ей id той кнопки, цвет которой планируется менять, в нашем случае это кнопка OK. Прописываем следующий код:

```
Button btnOk;
btnOk = (Button) findViewById(R.id.btnOk);
btnOk.setBackgroundColor(getResources().getColor(R.color.colorPink));
```

Теперь при запуске приложения и нажатии кнопки OK помимо изменения текста в компоненте TextView ее цвет меняется на светло-розовый.

Таким образом, все ресурсы, связанные с нажатием на кнопки, хранятся в одном файле. Если потребуется их изменить, это можно будет сделать, найдя его в папке с ресурсами, а не искать соответствующие места в коде программы.

Логи представляют из себя различные сообщения о работе программы в панели консоли при тестировании приложения. Они отображаются в окне LogCat, либо в окне Run. Логи имеют различные уровни важности, среди них такие как: ERROR, WARN, INFO, DEBUG и VERBOSE.

Помимо просмотра системных логов, можно создавать свои. Это делается с помощью специального класса Log и его методов: Log.v(), Log.d(), Log.i(), Log.w() и Log.e().

Для добавления логов нужно менять код MainActivity.java. Для примера будем описывать действия, которые происходят в коде. Будут создаваться DEBUG-логи, поэтому будет использоваться метод Log.d(). Он требует на вход тег и текст выводимого сообщения. Тег — это метка, которая упрощает поиск конкретного сообщения среди системных логов. Для создания тега используется следующий код:

```
private static final String TAG = "testLogs";
```

После добавления тега можно вызывать в коде программы методы, создающие логи. Для этого определим следующие события: поиск view-элементов, присвоение обработчика кнопкам, определение кнопки, вызывающей обработчик, и логи для каждой из двух кнопок.

Код для вызова методов выглядит следующим образом:

```
Log.d(TAG, "Определение кнопки, вызвавшую обработчик");
Log.d(TAG, "кнопка ОК");
Log.d(TAG, "кнопка Cancel");
```

Теперь при запуске приложения и нажатии на кнопки, логи будут содержать не только системные сообщения, но и пользовательские. Это удобно использовать для отладки более масштабных приложений, чтобы проверить отрабатывает ли та или иная часть программного кода.

Всплывающие сообщения могут выполнять аналогичные логам функции, но уже не в консоли, а непосредственно при выполнении приложения. С их помощью можно выводить различные подсказки для пользователя.

Для создания такие сообщений используется класс Toast. Сделаем так, чтобы помимо изменения текста TextView информация о нажатой клавише появлялась во всплывающих сообщениях.

Для этого нужно внутрь метода onClick добавить следующий код:

```
Toast.makeText(this, R.string.textOk, Toast.LENGTH_SHORT).show();
Toast.makeText(this, R.string.textCancel, Toast.LENGTH_SHORT).show();
```

Разберем используемый метод, он имеет следующую структуру:
Toast.makeText(context, text, duration)

Context — используем текущую Activity
Text — текст, который будет отображаться в сообщении, так как он уже добавлен в ресурсы приложения, можно просто указать его ID, избегая написания прямых строк
Duration — время, на протяжении которого сообщение будет показываться, есть две вариации использования: для длительного отображения - Toast.LENGTH_LONG, для короткого - Toast.LENGTH_SHORT.

Для отображения всплывающего окна на экране нужно вызвать метод show(), что и делается в конце кода.

2.4 Создание меню приложения. Контекстное меню.

Меню является важной частью любого мобильного приложения. Рассмотрим два способа добавления пунктов меню. Программный способ, который дает большую гибкость в настройке, и способ с .xml-файлом, который ускоряет процесс разработки и сокращает код.

Разберем простую реализацию меню в виде выпадающего списка при нажатии на соответствующую кнопку.

Для создания меню нужно использовать метод onCreateOptionsMenu, при его вызове ему на вход подается объект типа Menu, в который после и добавляются пункты. Этот метод инициирует первое появление меню на экране, ссылку на меню можно сохранить и использовать в любом месте кода до того момента, пока метод onCreateOptionsMenu не будет вызван вновь.
Реализацию этого обработчика всегда необходимо использовать из родительского класса, потому что она включает в меню дополнительные

системные пункты, которые нам не нужны. Для того, чтобы меню было видимым на экране, метод должен возвращать значение true.

Для добавления пунктов в меню, внутри метода нужно вызвать метод add(), на вход этому методу подается текст пункта меню. Как уже было сказано выше, метод onCreateOptionsMenu должен вернуть результат типа boolean. Если значение будет true, то меню отобразится на экране, а если false - нет.

То есть для отображения пунктов меню, достаточно вызвать метод onCreateOptionsMenu и добавить в него пункты через метод add.

Для того, чтобы нажатие на пункты меню запускало действие необходимо реализовать обработчик событий. В случае, как и с обычными кнопками, обработчиком является Activity, а методом, который это делает, является onOptionsItemSelected. На вход этому методу подается пункт меню, который был нажат — MenuItem. Чтобы определить какое именно меню было нажато, можно использовать метод getTitle(). Добавив в метод код по отображению сообщения, получим меню, пункты которого реагируют на нажатия.

Теперь добавим пункты меню с помощью того же метода add, но уже с другими входными параметрами. Метод выглядит следующим образом:

add(int groupId, int itemId, int order, CharSequence title);

Здесь метод имеет 4 параметра на вход:
groupId — идентификатор группы, которой принадлежит пункт меню,
itemId — идентификатор самого пункта меню,
order — параметр для задания последовательности пунктов меню,
title — отображаемый текст.

И добавим пункты меню с помощью новых параметров. В качестве идентификатора группы указывается просто 0. Если добавить для каждого пункта свою группу, то визуально это никак не будет видно, они не будут выделяться цветом или раздельно располагаться на экране, но может быть использовано при обработке нажатий или при подготовке отображения пунктов меню.

Идентификатор пунктов меню используется для определения какой пункт меню был нажат, он в дальнейшем пригодится при обработке нажатий.
Параметр для задания последовательности пунктов меню определяет позицию пункта меню и используется для определения порядка, в котором пункты будут расположены. Для удобства используется сортировка по возрастанию, то есть чем меньше номер у того или иного пункта, тем выше он будет отображен. Если этот параметр будет совпадать у нескольких пунктов, то

первым будет отображаться тот, который был создан раньше. Параметр текста остается без изменений.

Хоть визуально для пользователя пункты и остались в том же виде, но для разработчика взаимодействовать в дальнейшем с новыми пунктами меню будет проще.

Однако еще одним более простым способом создания меню является обычное редактирование .xml-файла, с условием того что этот файл находит в правильном директории. Для создания нужно щелкнуть правой кнопкой по папке res и выбрать New-> Androd Resourse Directory. В появившемся окне нужно указать имя файла, а в строке Resource type выбрать menu. Теперь можно добавить в этот файл пункты меню. Аналогичным образом при добавлении можно указать идентификаторы группы и самих пунктов, порядок и текст, который будет отображаться.

Для отображения этого меню на экране, нужно обратиться к методу onCreateOptionsMenu. В нем не нужно вручную вводить имя каждого пункта, а достаточно просто связать меню, которое дается на вход и созданный .xml-файл.

С помощью метода getMenuInflater получается MenuInflater, а следом вызывается его метод inflate. На вход передается .xml-файл и объект menu. MenuInflater берет объект menu и наполняет его пунктами согласно .xml-файлу.

При запуске меню выглядит точно так же, однако программно оно реализовано совершенно другим образом.

Помимо обычного меню в мобильных приложениях существует контекстное меню. Его можно вызвать длительным нажатием на каком-либо компоненте на экране. Обычно оно используется для выполнения каких-либо действий с одним из однородных объектов.

Принцип создания контекстного меню в целом напоминает создание обычного меню, но со своими отличиями. Для создания используется метод onCreateContextMenu, на вход которому подаются следующие параметры:
ContextMenu — само меню, в которое добавляются пункты,
View — элемент экрана, для которого контекстное меню вызывается,
ContextMenuInfo — содержит в себе дополнительную информацию, когда контекстное меню вызывается для элемента списка.

Для обработки будет использоваться метод onContextItemSelected, который аналогичен методу onOptionsItemSelected для обычного меню. Ему на вход подается MenuItem — пункт меню, который был нажат.

И еще понадобится третий метод registerForContextMenu, на вход он принимает View, и именно для этой View будет создаваться контекстное меню. В MainActivity.java нужно объявить и в методе onCreate привязать переменные к компонентам экрана, а далее указать, что именно для этих компонентов будет вызываться контекстное меню. Как раз для этого и будет использоваться вышеуказанный метод registerForContextMenu.

2.5 Анимация в приложениях

В мобильных приложениях можно использовать анимация. Существует огромное количество инструментов и способов создания и отображения анимации. Рассмотрим базовые.

Анимация – это трансформация, примененная к View-компонентам: изменение прозрачности, изменение размера, перемещение и поворот.

Все анимации конфигурируются в ресурсах приложения, то есть в .xml-файлах. После этого они считываются и присваиваются в коде программы требуемым View-компонентам.

Так как анимации являются ресурсами приложения, для их создания необходимо в папке res создать новый каталог, который будет хранить в себе анимации (аналогично тому, как это делалось для пунктов меню). Правой кнопкой мыши кликаем по папке res, затем выбираем Android Resource Directory, в окне в графе Resource type выбаем anim. Теперь в новой директории можно создавать файлы с анимацией.

При помощи анимации можно менять прозрачность компонента. Атрибут fromAlpha отвечает за начальное значение прозрачности, toAlpha — за конечное, duration — это длительность анимации в миллисекундах.

При помощи анимации можно изменить размер компонентов. Атрибуты fromXScale и fromYScale отвечают за начальные размеры компонента, то есть его ширину и высоту. Аналогично toXScale и toYScale отражают конечные размеры. Атрибуты pivotX и pivotY являются своеобразными координатами точки, относительно которой будет производиться изменение размера. То есть значения по 50% означают, что точка находится ровно посередине компонента. Атрибут duration это та же длительность анимации.

При помощи анимации можно осуществляться перемещение компонента по экрану в соответствии с прописанными координатами Атрибуты FromXDelta и fromYDelta это начальные координаты компонента, toXDelta и toYDelta — конечные, duration снова выступает в качестве атрибута длительности анимации.

При помощи анимации можно сделать поворот компонента. Последняя рассматриваемая анимация, но не последний созданный .xml-файл, это rotate, то есть поворот компонента. Атрибуты pivotX и pivotY используются для поворота вокруг определенной точки, их нужно прописать по аналогии с анимацией изменения размера. Атрибут fromDegrees указывает начальный угол поворота компонента, а toDegrees — конечный за время, указанное в том же duration.

Можно создавать как отдельные файлы, которые содержат одну анимацию, так и объединять анимации внутри одного файла. Если есть задача, чтобы анимации срабатывали одновременно, для них нужно установить одинаковую длительность в атрибуте duration.

Анимация читается из файла при помощи метода AnimationUtils.loadAnimation, которому на вход подается Activity и имя файла в директории с ресурсами, на выходе получится объект типа Animation. Далее нужно для компонента, к которому применяется анимация, вызвать метод startAnimation, с подачей на вход полученного ранее объекта типа Animation.

2.6 Принципы дизайна мобильного приложения

На сегодняшний день абсолютное большинство мобильных приложений должно обладать привлекательным пользовательским интерфейсом, а также он должен быть удобным и интуитивно понятным. Дизайн не должен использоваться ради дизайна. Из множества красивых, но неподходящих друг к другу элементов не получится создать хороший дизайн.

У каждого мобильного приложения, каким бы функционалом оно не обладало, есть фон. Фон должен выбираться так, чтобы не перетягивать на себя внимание, View-компоненты не должны теряется на фоне. И в целом элементы на экране не должны конкурировать между собой. То есть они не должны перетягивать внимание на себя.

Если стоит задача отобразить какой-либо текст через использование компонент TextView, тогда следует отказаться от фона, который содержит в себе надписи, чтобы не путать пользователя.

View-компоненты на экране, с которыми пользователю предстоит взаимодействовать следует делать достаточно большими, чтобы по ним можно было попадать без особого труда. Также важно, чтобы пользователь видел, когда он нажимает на тот или иной элемент управления.
Текст должен читаться без лишних усилий. Мобильные устройства имеют отличный от привычных для разработчика размер экрана, поэтому не нужно пытаться уместить на экране большое количество текста. Как правило,

используют шрифт не менее 11, чтобы его можно было разобрать на обычном расстоянии без увеличения.

Не стоит так же перегружать элементами экран. Компоненты на экране должны взаимно дополнять друг друга. Благодаря этому дизайн будет смотреться органично и создавать впечатление единой картины.

Не стоит использовать ядовитые цвета, нужно придерживаться небольшого количества цветов. Это значит, что нельзя мешать все цвета на одном экране.

3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

3.1 Лабораторная работа №1. Знакомство с Android Studio.

Цели работы:
Ознакомление с основами интерфейса Android Studio. Настройка эмулятора. Создание первого мобильного приложения.

Задание на лабораторную работу:

• Установить интегрированную среду разработки Android Studio
• Настроить и запустить эмулятор, подходящий под системные требования ПК
• Ознакомиться с разными режимами редактирования .xml-файла
• Создать и запустить свое первое мобильное приложение

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. На каких языках программирования можно вести разработку в Android Studio?
2. Какой главный недостаток Android Studio?
3. Чем нужно руководствоваться при выборе версии операционной системы, для которой будет вестись разработка?
4. Что нужно сделать если при запуске эмулятора отображается черный экран?
5. Какие основные файлы есть в любом мобильном приложении? Какие функции они выполняют?
6. Какие режимы редактирования есть у .xml-файлов?

3.2 Лабораторная работа №2. Компоненты экрана и контейнеры.

Цели работы:
Изучить иерархию компонентов экрана в мобильных приложениях. Познакомиться с видами ViewGroup для мобильных приложений. Разработать приложение с использованием разных контейнеров.

Задание на лабораторную работу:

• Создать новый проект и разработать в нем оболочку для меню уровней какой-нибудь игры с минимальным функционалом (не меньше 20–ти кнопок с номерами уровней, кнопка назад).
• Контейнер для .xml-файла использовать в соответствии с вариантом. При этом все кнопки должны быть равномерно расположены на экране.
Вариант 1 - ConstraintLayout.
Вариант 2 - LinearLayout
Вариант 3 - TableLayout
Вариант 4 – RelativeLayout

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Код компонентов в .xml-файле
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. Что из себя представляет иерархия компонентов в мобильном приложении?
2. Как по-другому называется экран в мобильном приложении?
3. В чем заключается суть хранения компонентов в контейнере ConstraintLayout?
4. В чем заключается суть хранения компонентов в контейнере LinearLayout?
5. В чем заключается суть хранения компонентов в контейнере TableLayout?
6. В чем заключается суть хранения компонентов в контейнере RelativeLayout?
7. В чем заключается суть хранения компонентов в контейнере AbsoluteLayout?
8. Какие виды EditText существуют?
9. Какой атрибут EditText хранит в себе информацию о том, какую клавиатуре при вводе нужно использовать?
10. Как добавить подсказку для поля ввода?

3.3 Лабораторная работа №3. Обработчик событий.

Цели работы:
Научиться описывать компоненты экрана в коде. Научиться обрабатывать нажатия на кнопки в мобильных приложениях Понять и применить на практике принцип хранения ресурсов приложения. Овладеть навыком добавления и отслеживания логов в мобильном приложении.

Задание на лабораторную работу:

• В приложении с предыдущей лабораторной работы добавить обработчики для кнопок с номерами уровней.
• Добавить на экран TextView, который будет изменяться при нажатии на кнопки и выводить информацию о том, какая именно была нажата.
• Весь текст хранить в файлах-ресурсах приложения. Текст с номерами кнопок также перенести в ресурсы.
• Также добавить логи с информацией о том, какая именно кнопка была нажата

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Код компонентов в .xml-файле.
Код методов из .java-файла
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. Какие шаги нужно сделать при добавлении новых компонентов на экран при последующей обработке их в коде?
2. Какой интерфейс нужно имплементировать для создания обработчика событий?
3. С помощью какого метода кнопке присваивается обработчик событий?
4. Почему важно хранить используемые строки в отдельном файле?
5. Можно ли хранить разные ресурсы в одном файле? Почему?
6. Для чего используются логи приложения и что они из себя представляют?

3.4 Лабораторная работа №4. Меню и динамическое добавление компонентов на экран.

Цели работы:
Получить практические навыки создания меню. Научиться создавать контекстное меню. Изучить и применить принцип динамического добавления компонентов на экран мобильного приложения.

Задание на лабораторную работу:

• Разработать приложение, в котором будет функционал для добавления новых компонентов на экран.

• Непосредственно добавление должно осуществляться после нажатия на соответствующую клавишу в меню (меню также нужно добавить).
• Для новых компонентов предусмотреть возможность редактирования их атрибутов (например, цвета) через вызов для них контекстного меню.
• Предусмотреть наличие в меню не только пункта для добавления нового компонента, но и очищения рабочей области, а также выхода из приложения.
• Предусмотреть вывод всплывающих сообщений при добавлении новых компонентов и очистке рабочей области.

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Код компонентов в .xml-файле.
Код методов из .java-файла
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. В чем отличие обычного меню от контекстного?
2. Какие методы используются при создании и обработке нажатий на пункты меню?
3. Какие параметры подаются на вход методу add() при его расширенном вызове?
4. Какими способами можно добавить меню в приложение? Какие у них плюсы и минусы?
5. Какими преимуществами обладает программное создание экрана?
6. Что такое SeekBar? Привести примеры использования.

3.5 Лабораторная работа №5. Разработка мобильного приложения «Калькулятор».

Цели работы:
Разработать полноценное приложение калькулятора. Изучить принцип переключения между экранами в мобильном приложении. Научиться проигрывать анимацию для компонентов экрана. Изучить разные состояния Activity и понять принцип переключения между ними.

Задание на лабораторную работу:

• Разработать приложение калькулятора, у которого будет 3 экрана: начальный, экран стандартных вычислений (4 базовых арифметических) и экран с расширенными вычислениями (продумать самостоятельно).
• Предусмотреть возможность свободного переключения между экранами.
• Использовать в приложении анимацию (например, при выводе ответа или при промежуточных действиях).
• В качестве ввода цифр использовать не вызов клавиатуры, а кнопки с ними (как это сделано в аналогах).
• Предусмотреть возможность стирания некорректного ввода и сброса вычислений.

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Код компонентов в .xml-файле.
Код методов из .java-файла
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. С помощью какого метода можно передать значение EditText в переменную?
2. Как поведет себя приложение при попытке разделить на ноль?
3. В какой папке хранятся анимации приложения?
4. Какой атрибут хранит в себе значение длительности анимации?
5. Какой метод используется при чтении анимации из файла?
6. Какой файл является конфигурацией всего приложения, в котором указываются его параметры?
7. С помощью какого метода происходит вызов Activity?
8. Какие существуют состояния у Activity и когда они вызываются?
9. Что такое Task? В какой момент он создается?

3.6 Лабораторная работа №6. Принципы дизайна мобильных приложений.

Цели работы:
Применить на практике изученные принципы дизайна мобильных приложений

Задание на лабораторную работу:

• Переработать созданный в рамках предыдущей работы калькулятор согласно изученным принципам дизайна.
• Добавить «умный» выход из приложения.

Содержание отчета по лабораторной работе:

Титульный лист
Название и цель работы.
Номер группы, фамилия и инициалы студента.
Задание на лабораторную работу.
Код компонентов в .xml-файле.
Код методов из .java-файла
Скриншоты выполнения заданий.
Выводы.

Контрольные вопросы:

1. Какие принципы мобильного дизайна существуют и в чем их суть?
2. Для чего нужен «умный» выход из приложения?
3. Какой метод используется для замера времени нажатия на кнопку?

4. СПИСОК ЛИТЕРАТУРЫ

1. Д. Гриффитс, Д. Гриффитс Head Rirst. Программирование для Android/. – 2-е изд. – СПб: Питер, 2018. - 912 с.
2. Марсикано, К. Стюарт, Б. Филлипс Android. Программирование для профессионалов/. – 4-е изд. – СПб: Питер, 2022. – 704 с.
3. П. Дейтел, Х. Дейтел, А. Уолд Android для разработчиков/. – 3-е изд. – СПб: Питер, 2016. – 512 с.
4. Я. Ф. Дарвин Android. Сборник рецептов. Задачи и решения для разработчиков приложений. – М: Альфа-книга, 2019. – 768 с.
5. Д. Колисниченко Программирование для Android/. – 3-е изд. – СПб: BHV, 2016. – 512 с.
6. М. Федотенко Разработка мобильных приложений. Первые шаги. – М: Лаборатория знаний, 2020. – 336 с.
7. Л. Пирская Разработка мобильных приложений в среде Android Studio. – Таганрог: Южный федеральный университет, 2019. – 123 с.
8. Л. Гарднер, Д. Григсби Разработка веб-сайтов для мобильных устройств. – СПб.: Питер, 2013. – 447 с.

5. ВОПРОСЫ К ЗАЧЕТУ

1. На каких языках программирования можно вести разработку в Android Studio?
2. Какие основные файлы есть в любом мобильном приложении? Какие функции они выполняют?
3. Какие режимы редактирования есть у .xml-файлов?
4. Как происходит установка эмулятора?
5. Какие компоненты экрана вы знаете?
6. Какие виды контейнеров существуют?
7. Какой интерфейс нужно имплементировать для создания обработчика событий?
8. С помощью какого метода кнопке присваивается обработчик событий?
9. Для чего используются логи приложения и что они из себя представляют?
10. В чем отличие обычного меню от контекстного?
11. Какие методы используются при создании и обработке нажатий на пункты меню?
12. Какими способами можно добавить меню в приложение?
13. Какой атрибут хранит в себе значение длительности анимации?
14. Какой метод используется при чтении анимации из файла?
15. Какие существуют состояния у Activity и когда они вызываются?
16. Какие принципы мобильного дизайна существуют?
17. Для чего нужен «умный» выход из приложения?