Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

Н. С. Богданова

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

ПРАКТИКУМ
по выполнению лабораторных работ
для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Гомель 2021

УДК 004.42(075.8)
ББК 32.972я73
Б73

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 12 от 29.05.2020 г.*)

Рецензент: доц. каф. «Информатика» ГГТУ им. П. О. Сухого
канд. физ.-мат. наук, доц. *О. В. Лукьяненко*

Богданова, Н. С.

Б73 Разработка приложений для мобильных устройств : прктикум по выполнению лаборатор. работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / Н. С. Богданова. – Гомель : ГГТУ им. П. О. Сухого, 2021. – 44 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Практикум составлен в соответствии с учебной программой по дисциплине «Разработка приложений для мобильных устройств» для студентов специальности 1-40 04 01 «Информатика и технологии программирования». Предлагается шесть лабораторных работ для закрепления учебного материала по данной дисциплине. Рассматриваются современные методы, языки, технологии и инструментальные средства проектирования и разработки программных продуктов для платформы Android.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования.

УДК 004.42(075.8)
ББК 32.972я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью изучения дисциплины «Разработка приложений для мобильных устройств» является формирование у студентов теоретических знаний и приобретение навыков, необходимых для создания современных мобильных приложений, Дисциплина предназначена для обучения студентов приемам, методам и технологиям разработки для мобильных устройств на платформе Android, основным принципам программирования мобильных систем.

Основными задачами изучаемой дисциплины являются:

– приобретение знаний по основным понятиям платформы Android;

– формирование навыков работы с основными инструментальными средствами конструирования и создания прикладных программных продуктов для платформы Android;

– изучение принципов решения задач с использованием современных методов программирования;

– усвоение языковых средств, используемых для создания мобильных приложений.

Изучение курса «Разработка приложений для мобильных устройств» предполагает получение студентами теоретических знаний и приобретение навыков, необходимых для разработки современных мобильных приложений на высоком профессиональном уровне.

Целью практикума по выполнению лабораторных работ для по курсу «Разработка приложений для мобильных устройств» для студентов специальности 1-40 04 01 – «Информатика и технологии программирования» является изучение основных понятий платформы Android, принципов программирования мобильных приложений, методов, технологий и инструментальных средств разработки мобильных приложений на платформе Android, жизненного цикла активности, организации Android-приложения, а также изучение принципов обращения с базами данных в мобильных приложениях и их долговременное хранение.

Практикум предлагает для выполнения шесть лабораторных работ. В каждой работе есть небольшая теоретическая часть, которая позволит студентам лучше справиться с поставленными задачами.

ЛАБОРАТОРНАЯ РАБОТА № 1

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ ДЛЯ ПЛАТФОРМЫ ANDROID

Цель работы: изучить основы построения Android приложения, создания проекта Android, построение макета пользовательского интерфейса и запуск на устройстве.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ОС Android — полнофункциональная платформа с открытым кодом на базе Linux, разрабатываемая компанией Google. Это мощная платформа разработки, включающая все необходимое для построения современных приложений из кода Java и XML. Более того, построенные приложения могут устанавливаться на множестве разных устройств — телефонах, планшетах и не только.

На рисунке 1 представлена архитектура ОС Android приложения.

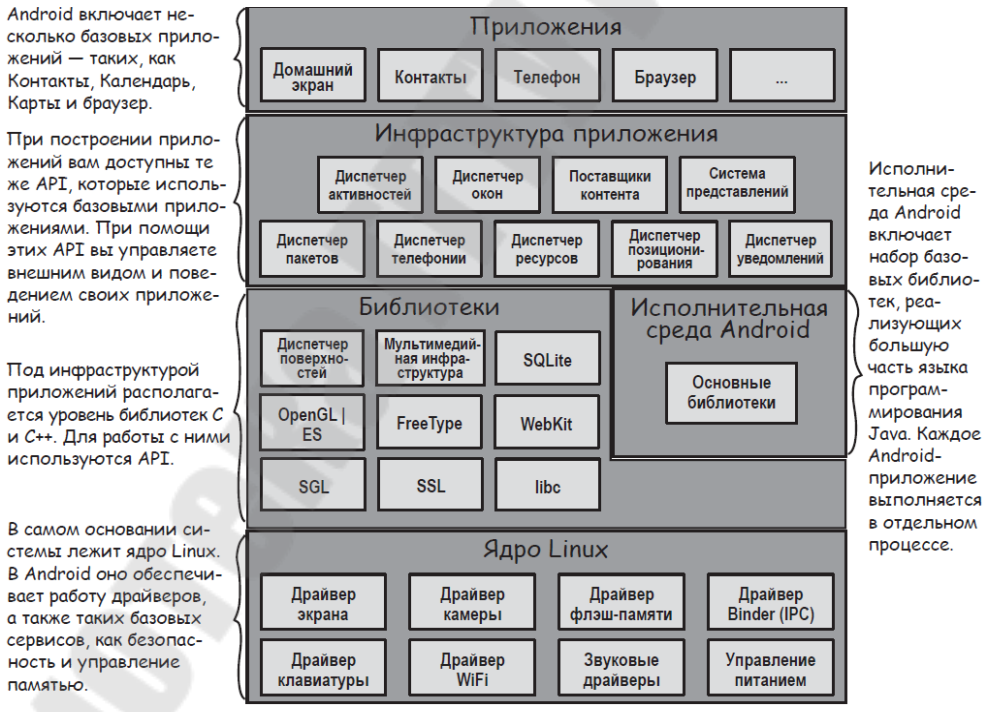

Рисунок 1 – Архитектура ОС Android приложения

Набор инструментов и библиотек API – Android SDK, предназначенный для компьютеров с архитектурой процессора x86 под операционными системами Windows XP, Windows Vista, Windows 7, Mac OS X (10.4.8 или выше) и Linux.

Среда исполнения Java Runtime Environment (JRE).

Комплект разработчика Java Development Kit (JDK).

Среда разработки Android Studio и SDK Tools.

Построение Anroid приложения осуществляется в 4 этапа: подготовка среды; построение приложения; запуск приложения; изменение приложения

Android Studio — версия IDEA, которая включает версию Android SDK и дополнительные инструменты графических интерфейсов, упрощающие разработку приложений.

Кроме редактора и доступа к инструментам и библиотекам из Android SDK, Android Studio предоставляет шаблоны, упрощающие создание новых приложений и классов, а также средства для выполнения таких операций, как упаковка приложений и их запуск.

Действия по созданию проекта показаны на рисунках 2 и 3.

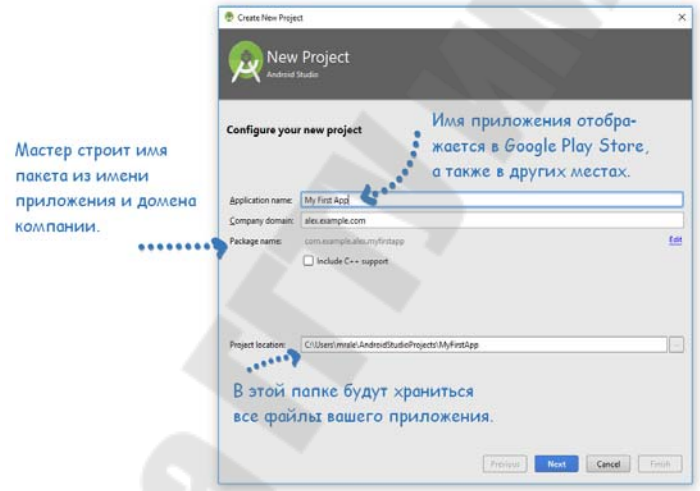

Рисунок 2 – Настройка проекта

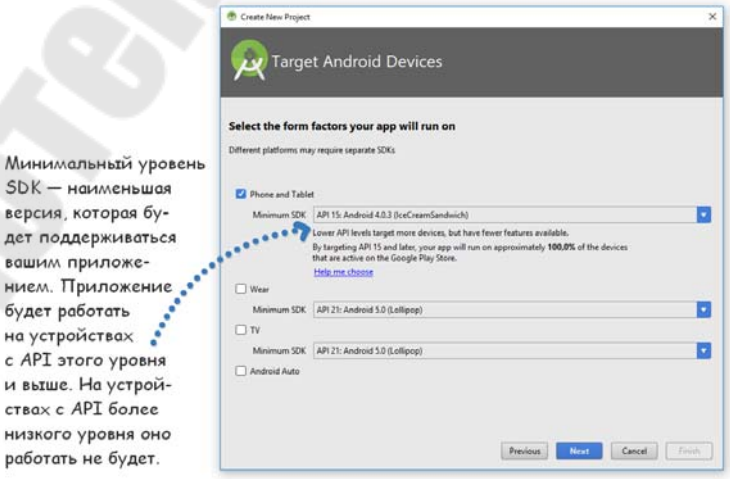

Рисунок 3 – Выбор уровня API

Каждое Android-приложение состоит из экранов, а каждый экран состоит из активности и макета.

Активность — одна четко определенная операция, которую может выполнить пользователь. Например, в приложении могут присутствовать активности для составления сообщения электронной почты, поиска контакта или создания снимка. Активности обычно ассоциируются с одним экраном и программируются на Java.

Макет описывает внешний вид экрана. Макеты создаются в виде файлов в разметке XML и сообщают Android, где располагаются те или иные элементы экрана.

Работа по созданию и настройки активности и вид проекта показаны на рисунках 4–6.

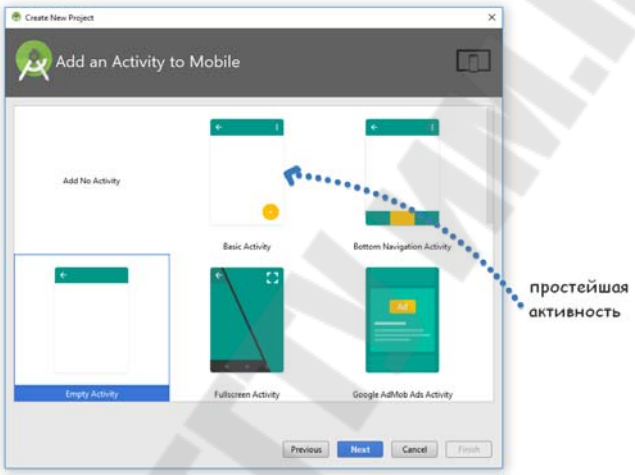

Рисунок 4 – Создание активности

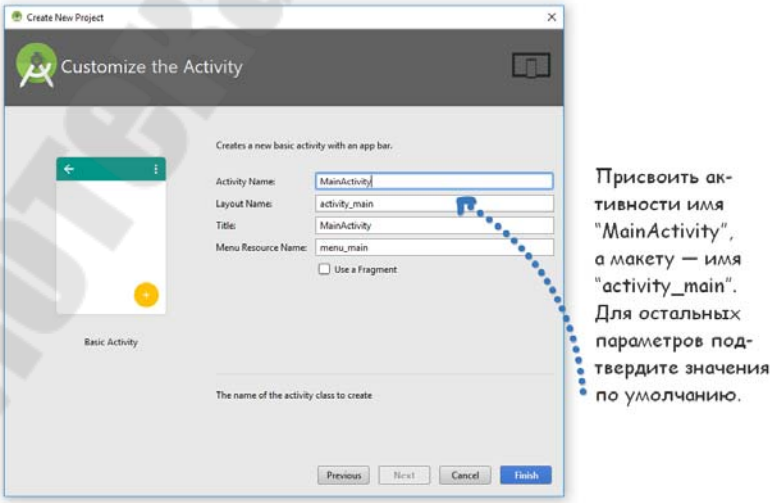

Рисунок 5 – Настройка активности

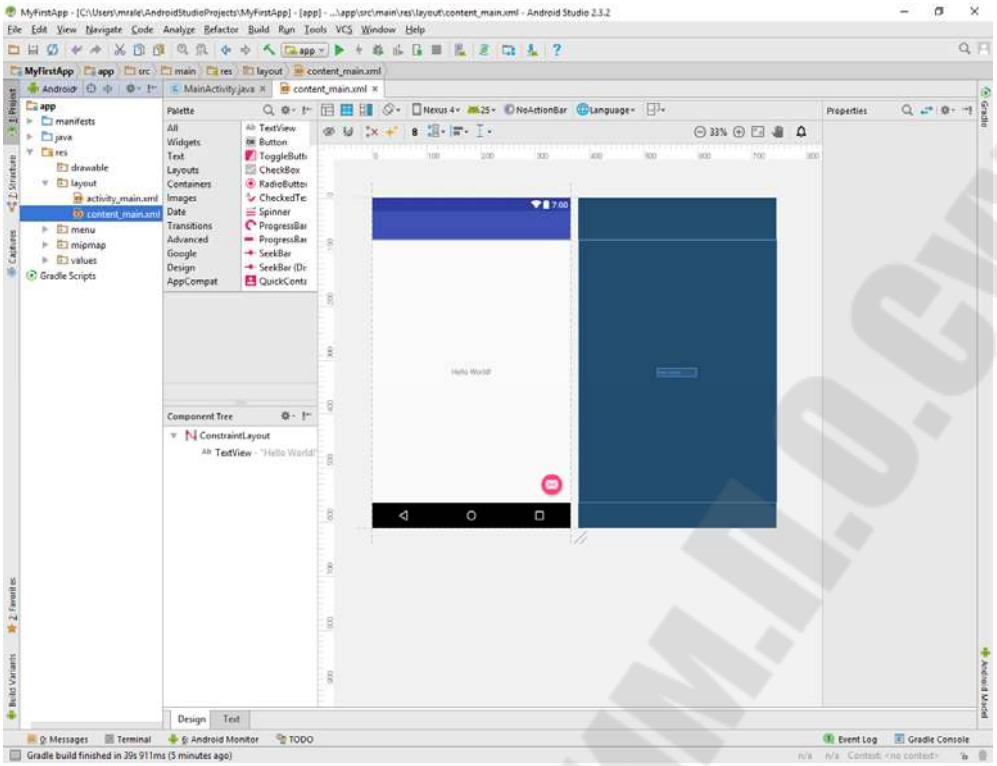

Рисунок 6 – Вид проекта

Ключевые файлы и папки проекта:

Папка res/ предназначена для системных ресурсов.

Папка layout/ содержит макеты, а папка values/ — файлы ресурсов, содержащие используемые в программе значения (например, строки).

В каждом Android-приложении должен присутствовать файл с именем AndroidManifest.xml. Этот файл (манифест) содержит необходимую информацию о приложении: из каких компонентов оно состоит, какие библиотеки необходимы для его работы и другие объявления.

Файл MainActivity.java определяет активность.

Файл activity_main.xml определяет макет.

Файл strings.xml состоит из строковых пар «идентификатор/ значение».

Запуск приложения в эмуляторе Android:

1. Открыть AVD Manager. Меню Tools пункт Android и затем AVD Manager.

2. Выбрать тип устройства. К примеру, выбрать в меню Category пункт Phone, затем выберите в списке Nexus 4. Щелкнуть на кнопке Next (рисунок 7).

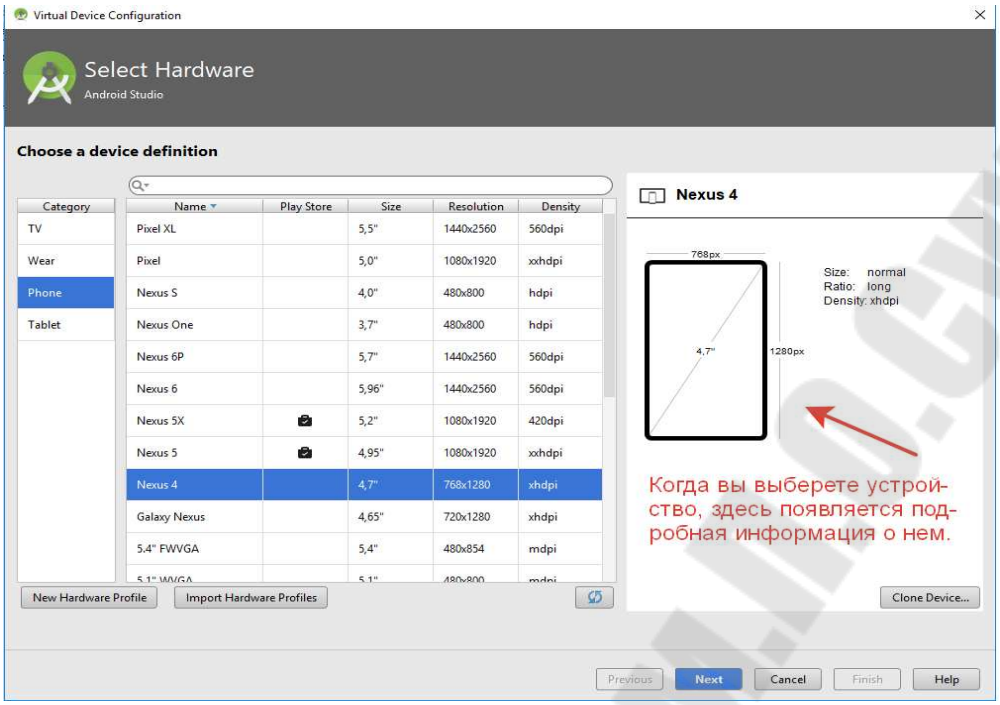

Рисунок 7 – Выбор типа устройства

3. Выбрать образ системы. Выбрать версию Android, которая должна поддерживаться AVD, и тип процессора (ARM или x86) (рисунок 8).

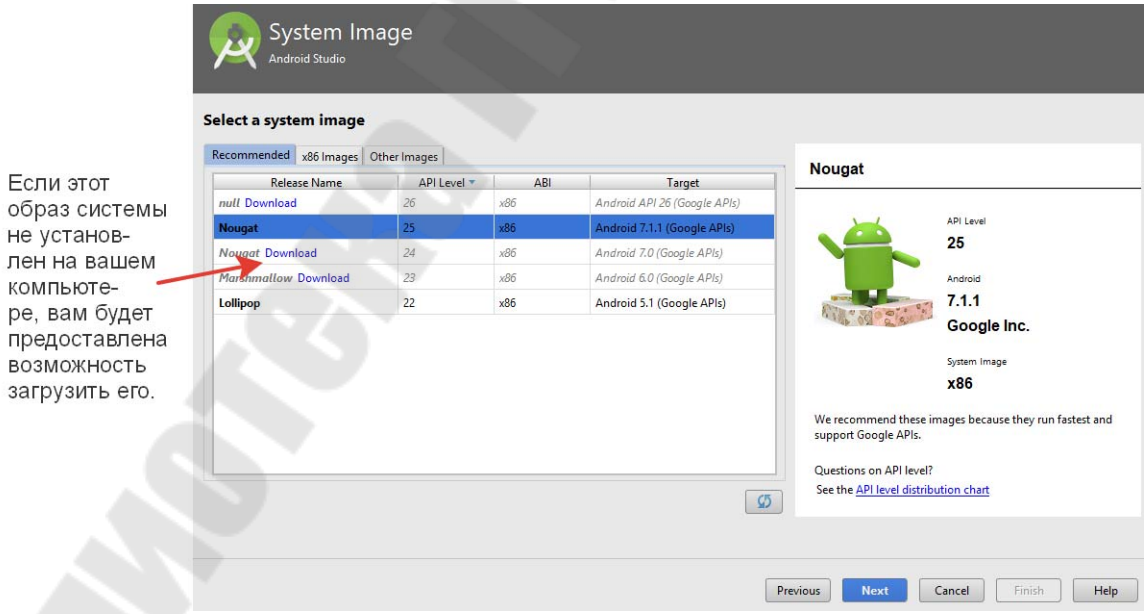

Рисунок 8 – Выбор образа системы

4. Проверка конфигурации AVD (рисунок 9).

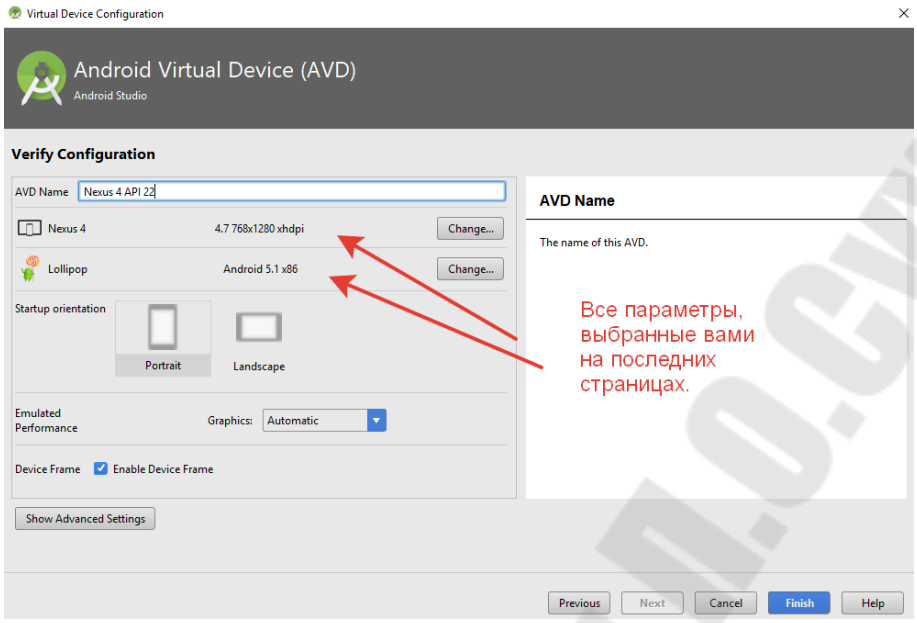

Рисунок 9 – Проверка конфигурации AVD

5. Запуск приложения в эмуляторе. Выбрать команду Run app из меню Run. Когда будет предложено выбрать устройство, убедиться в том, что установлен переключатель Launch emulator.

Этапы, которые происходят при запуске приложения:

Android Studio запускает эмулятор, загружает AVD и устанавливает приложение.

Когда приложение запустится, на базе MainActivity.java создается объект активности.

Активность указывает, что она использует макет activity_main.xml.

Активность приказывает Android вывести макет на экран.Текст “Hello world!” появляется на экране.

Задание

Разработать в Android Studio приложение, которое должно:

a) содержать текстовое поле и кнопку;

b) при нажатии на кнопку в текстовое поле выводить ФИО и группу студента.

Выполнить тест-драйв, запустив приложение на эмуляторе.

Требования к отчету

Отчет по лабораторной работе должен содержать:

– название и цель работы;

– вид макета приложения в визуальном редакторе;
– вид приложения в эмуляторе;
– код файла ***activity_main.xml*** в редакторе кода и файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
– выводы о проделанной работе.

Контрольные вопросы

1. Что представляет собой ОС Android?
2. Программные средства для разработки приложений под платформу Android.
3. Что содержит пакет Android Software Development Kit (SDK)?
4. Архитектура ОС Android приложения.
5. Основы построения Android приложения.
6. Создание проекта Android.
7. Что такое «уровень API»? Примеры API.
8. Построение макета пользовательского интерфейса.
9. Активности и макеты. Для чего служат?
10. Вид проекта. Структура папок.
11. Перечислите основные файлы и папки проекта.
12. Запуск на устройстве.
13. Этапы, которые происходят при запуске приложения.
14. Изменение приложения. Два способа просмотра и редактирования файлов макетов в Android Studio.

ЛАБОРАТОРНАЯ РАБОТА № 2

СОЗДАНИЕ ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ, ИСПОЛЬЗОВАНИЕ ПРОЕКТИРОВАНИЕ MVC В РАЗРАБОТКЕ ANDROID ПРИЛОЖЕНИЙ

Цель работы: изучить этапы построения интерактивного Android приложения, возможность работы с множественными активностями и интентами и использования шаблона проектирования MVC.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Обновление макета

Добавление компонентов графического интерфейса осуществляется двумя способами: в разметке XML или в визуальном редакторе.

Добавления кнопки в визуальном редакторе.

Слева от визуального редактора располагается палитра с компонентами графического интерфейса, которые можно перетаскивать мышью на макет. Раздел Widgets и в нем компонент кнопки (Button). Необходимо щелкнуть на нем и перетащить на макет в визуальном редакторе.

Изменения, внесенные в визуальном редакторе, отражаются в XML.

Кнопки и надписи — субклассы одного класса Android View.

Тот факт, что кнопки и надписи имеют так много общих свойств, вполне логичен — оба компонента наследуют от одного класса Android View.

Элемент RelativeLayout.

Код макета начинается с элемента – <RelativeLayout>. Элемент <RelativeLayout> сообщает Android, что компоненты графического интерфейса в макете должны отображаться относительно друг друга.

Все изменения, вносимые в разметке XML макета, отражаются в визуальном редакторе.

Примеры:

android:id

Имя, по которому идентифицируется компонент. Свойство ID используется для управления работой компонента из кода активности, а также для управления размещением компонентов в макете:

```
android:id="@+id/button"
```

android:text

Свойство сообщает Android, какой текст должен выводиться в компоненте. В случае <Button> это текст, выводимый на кнопке:

```
android:text="New Button"
```

android:layout_width, android:layout_height

Эти свойства задают базовую ширину и высоту компонента. Значение "wrap_content" означает, что размеры компонента должны подбираться по размерам содержимого:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

Для добавления строкового ресурса в файл *strings.xml* необходимо выполнить следующие действия:

```
<string name="имя_строки">значение</string>,
```

где имя_строки — идентификатор строки, а значение — собственно строковое значение.

Синтаксис добавления массива строк выглядит так:

```
<string-array name="имя_массива_строк">
<item>значение1</item>
<item>значение2</item>
<item>значение3</item>
...
</string-array>,
```

где имя_массива — имя массива, а значение1, значение2, значение3 — отдельные строковые значения, входящие в массив.

Для обращения к массиву строк в макете используется синтаксис, сходный с синтаксисом получения строкового значения. Вместо конструкции

```
"@string/имя_строки"
```

используется конструкция

```
"@array/имя_массива",
```

где имя_массива — имя массива.

Подключение активности.

Кнопка должна вызвать метод.

Чтобы щелчок на кнопке приводил к вызову метода активности, необходимо внести изменения в двух файлах:

Изменения в файле макета *activity_name.xml*.

Необходимо указать, какой метод активности должен вызываться при щелчке на кнопке.

Изменения в файле активности *NameActivity.java*.

Необходимо написать метод, который будет вызываться при щелчке.

OnClick и метод, вызываемый при щелчке

Чтобы сообщить Android, какой метод должен вызываться при щелчке на кнопке, достаточно всего одной строки разметки XML. Все, что для того нужно — добавить атрибут android:onClick в элемент <button> и указать имя вызываемого метода:

```
android:onClick="clickMethod"
```

Все активности также должны реализовать метод onCreate(). Метод onCreate() вызывается при создании объекта активности и используется для настройки основных параметров — например, выбора макета, с которым связывается активность. Это делается при помощи метода setContentView().

При этом в активности должен существовать соответствующий метод:

```
public void clickMethod(View view){ }
```

Чтобы метод мог реагировать на щелчки на кнопке, он должен быть объявлен открытым (public), возвращать void и получать один параметр типа View.

Параметр **View** определяет компонент графического интерфейса, инициировавший вызов метода (в данном случае это кнопка). Компоненты графического интерфейса — такие, как кнопки и надписи, — все являются специализациями View.

Программирование логики.

Использование findViewById() для получения ссылки на компонент.

Для получения ссылки на два компонента графического интерфейса можно воспользоваться методом findViewById(). Метод findViewById() получает идентификатор компонента в виде параметра и возвращает объект View. Далее остается привести возвращаемое значение к правильному типу компонента (например, TextView или Button).

Получив ссылку на объект View, можно вызывать его методы.

Метод findViewById() предоставляет Java-версию компонента графического интерфейса. Это означает, что можно читать и задавать свойства компонента при помощи методов, предоставляемых классом Java.

Допустим. вы хотите, чтобы в надписи brands отображался текст “Mauro”. Класс TextView содержит метод setText(), используемый для задания свойства text. Он используется следующим образом:

```
brands.setText(" Mauro ");
```

Множественные активности и интенты.

Задачей называются две и более активности, объединенные в цепочку.

Элемент <EditText> определяет текстовое поле с возможностью редактирования и ввода текста. Класс текстового поля наследует от класса Android View.

Новая активность в Android Studio создается командой File → New... → Activity.

Для каждой создаваемой активности в файле *AndroidManifest.xml* должна быть создана запись.

Интент представляет собой разновидность сообщений, используемых для организации взаимодействия между компонентами Android.

Явный интент предназначен для конкретного компонента. Явный интент создается командой:

```
Intent intent = new Intent(this, Tar-
                get.class);
```

Активности запускаются вызовом startActivity(intent). Если ни одна подходящая активность не найдена, метод инициирует исключение ActivityNotFoundException.

Используйте метод putExtra() для включения дополнительной информации в интент.

Используйте метод getIntent() для получения интента, запустившего активность.

Используйте методы get*Extra() для чтения дополнительной информации, связанной с интентом. Метод getStringExtra() читает String, getIntExtra() читает int, и т. д.

Действие описывает стандартную операцию, которую может выполнять активность. Так, для отправки сообщений используется обозначение Intent.ACTION_SEND.

Чтобы создать неявный интент с указанием действия, используйте запись:

```
Intent intent = new Intent(action);
```

Для описания типа данных в интенте используется метод setType().

Android производит разрешение интентов на основании имени компонента, действия, типа данных и категорий, указанных в интенте. Содержимое интента сравнивается с фильтрами интентов из файла *AndroidManifest.xml* каждого приложения. Чтобы активность получала неявные интенты, она должна включать категорию DEFAULT.

Метод createChooser() позволяет переопределить стандартное диалоговое окно выбора активности в Android. При использовании этого метода можно указать текст заголовка, а у пользователя нет возможности назначить активность по умолчанию. Если метод не находит ни одной активности, способной получить переданный интент, он выводит сообщение. Метод createChooser() возвращает объект Intent.

Для чтения значений строковых ресурсов используется синтаксис getString(R.string.stringname).

Взаимодействия MVC при получении ввода от пользователя представлено на рисунке 10.

Рисунок 10 – Взаимодействия MVC при получении ввода от пользователя

Задание

1) В соответствии с вариантом разработать в Android Studio интерактивное приложение, работающее с одной активностью, которое должно:

a) содержать текстовое поле, кнопку, раскрывающийся список;

b) при выборе наименования определенного вида из списка и по щелчку на кнопке в текстовое поле выводились соответствующие данные по выбранному виду.

Выполнить тест-драйв, запустив приложение на эмуляторе.

2) Разработать согласно варианту в Android Studio интерактивное приложение, работающее с несколькими активностями и интентами, которое должно:

a) содержать текстовое поле, кнопку;

b) при щелчке по кнопке приложение должно предложить выбрать активность, используемую для передачи данных.

Выполнить тест-драйв, запустив приложение на эмуляторе.

3) Разработать Android приложение согласно варианту с использованием архитектуры "Модель-Представление-Контроллер", выполняющее ввод данных, вывод и редактирование в соответствии с вариантом. Для выполнения каждого пункта задания (a-d) использовать отдельную Activity и модель. Выполнить тест-драйв, запустив приложение на эмуляторе.

Варианты заданий:

Задания № 1–2:

Вариант № 1. Student: Из списка групп университета, выдать фамилии студентов, соответствующие группе.

Вариант № 2. Patient: Выдать сведения о пациентах, имеющих определенный диагноз.

Вариант № 3. Abiturient: Выдать сведения об абитуриентах, получивших определенный балл.

Вариант № 4. Book: Выдать книги, выпущенных заданным издательством.

Вариант № 5. Book: Выдать книги, написанных заданным автором.

Вариант №6. Phone: Выдать сведения об абонентах, которые пользовались междугородной связью.

Вариант № 7. Car: Выдать список автомобилей заданной марки.

Вариант № 8. Avtosalon: Выдать список автомобилей заданного года выпуска.

Вариант № 9. Product: Выдать список товаров для заданного наименования.

Вариант № 10. Train: Выдать список поездов, следующих до заданного пункта назначения.

Вариант № 11. Bus: Выдать список автобусов для заданного номера маршрута.

Вариант № 12. Airlines: Выдать список рейсов для заданного дня недели.

Задание № 3:

Вариант № 1. Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Создать массив объектов. Вывести:

a) список студентов заданного факультета;
b) списки студентов для каждого факультета и курса;
c) список студентов, родившихся после заданного года;
d) список учебной группы.

Вариант № 2. Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета.

Создать массив объектов. Вывести:

a) список покупателей в алфавитном порядке;
b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Вариант № 3. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз.

Создать массив объектов. Вывести:

a) список пациентов, имеющих данный диагноз;
b) список пациентов, номер медицинской карты у которых находится в заданном интервале.

Вариант № 4. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки.

Создать массив объектов. Вывести:

a) список абитуриентов, имеющих неудовлетворительные оценки;
b) список абитуриентов, средний балл у которых выше заданного;
c) выбрать заданное число и абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Вариант № 5. Book: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет.

Создать массив объектов. Вывести:

a) список книг заданного автора;

b) список книг, выпущенных заданным издательством;

c) список книг, выпущенных после заданного года.

Вариант № 6. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации.

Создать массив объектов. Вывести:

a) список квартир, имеющих заданное число комнат;

b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;

c) список квартир, имеющих площадь, превосходящую заданную.

Вариант № 7. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров.

Создать массив объектов. Вывести:

a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное;

b) сведения об абонентах, которые пользовались междугородной связью;

c) сведения об абонентах в алфавитном порядке.

Вариант № 8. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество.

Создать массив объектов. Вывести:

a) список товаров для заданного наименования;

b) список товаров для заданного наименования, цена которых не превосходит заданную;

c) список товаров, срок хранения которых больше заданного.

Вариант № 9. Train: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс).

Создать массив объектов. Вывести:

a) список поездов, следующих до заданного пункта назначения;

b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

c) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

Вариант № 10. Bus: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег.

Создать массив объектов. Вывести:

a) список автобусов для заданного номера маршрута;

b) список автобусов, которые эксплуатируются больше 10 лет;

c) список автобусов, пробег у которых больше 100000 км.

Вариант № 11. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер.

Создать массив объектов. Вывести:

a) список автомобилей заданной марки;

b) список автомобилей заданной модели, которые эксплуатируются больше и лет;

c) список автомобилей заданного года выпуска, цена которых больше указанной.

Вариант № 12. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество.

Создать массив объектов. Вывести:

a) список товаров для заданного наименования;

b) список товаров для заданного наименования, цена которых не превосходит заданную;

c) список товаров, срок хранения которых больше заданного.

Требования к отчету

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- вид макета приложения в визуальном редакторе;
- вид приложения в эмуляторе;
- код файла ***activity_main.xml*** в редакторе кода и код файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
- выводы о проделанной работе.

Контрольные вопросы

1. Элементы Button, Spinner?
2. Класс Android View?
3. Как создается массив строковых значений? Обращение к массиву.
4. Что необходимо сделать, чтобы при щелчке на кнопке вызывался метод?
5. Класс R.java*?*
6. Методы findViewById(), setText(), getSelectedItem().
7. Что необходимо выполнить, чтобы добавить вспомогательный класс в проект Android?
8. Множественные активности интенты.
9. Этапы создания интерактивного приложения.
10. Использование шаблона проектирование MVC в разработке Android приложений. Архитектура "Модель-Представление-Контроллер" и Android.
11. Преимущества MVC.
12. Обновление уровня представления.
13. Обновление уровня контроллера.
14. Иерархия представлений.
15. Атрибуты виджетов.
16. Объекты View.
17. Ссылки на ресурсы в XML.
18. Добавление слушателя для компонентов.
19. Анонимные внутренние классы.

ЛАБОРАТОРНАЯ РАБОТА № 3

ЖИЗНЕННЫЙ ЦИКЛ АКТИВНОСТИ, СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Цель работы: изучить методы жизненного цикла активности и получить практические навыки по созданию пользовательского интерфейса.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Управление жизненным циклом операций путем реализации методов обратного вызова имеет важное значение при разработке надежных и гибких приложений. На жизненный цикл операции напрямую влияют его связи с другими операциями, его задачами и стеком переходов назад.

Существует всего три состояния операции:

Возобновлена

Операция выполняется на переднем плане экрана и отображается для пользователя. (Это состояние также иногда называется «Выполняется».)

Приостановлена

На переднем фоне выполняется другая операция, которая отображается для пользователя, однако эта операция по-прежнему не скрыта. То есть поверх текущей операции отображается другая операция, частично прозрачная или не занимающая полностью весь экран. Приостановленная операция полностью активна (объект Activity по- прежнему находится в памяти, в нем сохраняются все сведения о состоянии и информация об элементах, и он также остается связанным с диспетчером окон), однако в случае острой нехватки памяти система может завершить ее.

Остановлена

Операция полностью перекрывается другой операцией (теперь она выполняется в фоновом режиме). Остановленная операция по-прежнему активна (объект Activity по-прежнему находится в памяти, в нем сохраняются все сведения о состоянии и информация об элементах, но объект больше не связан с диспетчером окон). Однако операция больше не видна пользователю, и в случае нехватки памяти система может завершить ее.

Если операция приостановлена или полностью остановлена, система может очистить ее из памяти путем завершения самой операции (с помощью метода finishQ) или просто завершить ее процесс. В

случае повторного открытия операции (после ее завершения) ее потребуется создать полностью.

При переходе операции из одного вышеописанного состояния в другое, уведомления об этом реализуются через различные методы обратного вызова. Все методы обратного вызова представляют собой привязки, которые можно переопределить для выполнения подходящего действия при изменении состояния операции. Указанная ниже базовая операция включает каждый из основных методов жизненного цикла (рисунок 12).

При реализации этих методов жизненного цикла всегда вызывают реализацию суперкласса, прежде чем выполнять какие-либо действия.

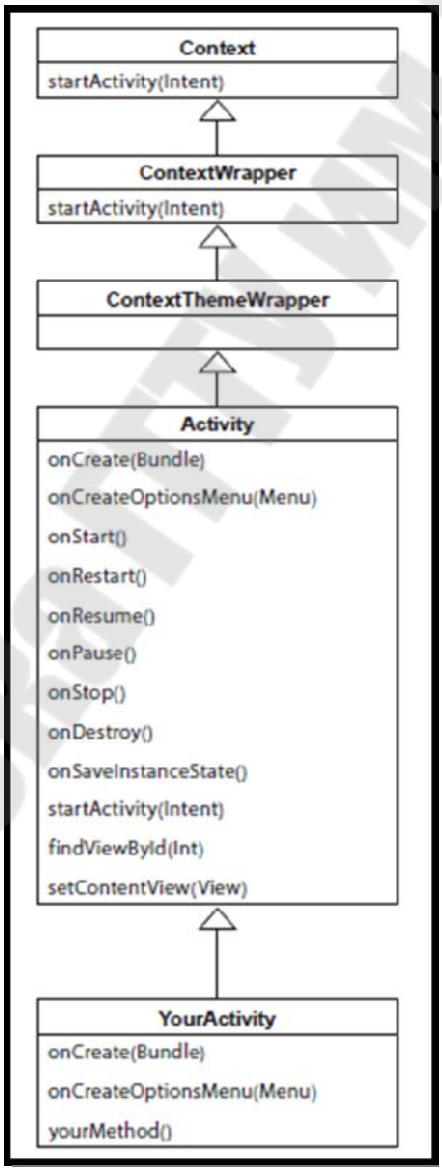

Рисунок 12 – Методы жизненного цикла активности

Вместе все эти методы определяют весь жизненный цикл операции. С помощью реализации этих методов можно отслеживать три вложенных цикла в жизненном цикле операции.

Весь жизненный цикл операции происходит между вызовом метода onCreatet) и вызовом метода onDestrov(). Ваша операция должна выполнить настройку «глобального» состояния (например, определение макета) в методе onCreate(), а затем освободить все оставшиеся в onDestrov() ресурсы. Например, если в вашей операции имеется поток, выполняющийся в фоновом режиме, для загрузки данных по сети, операция может создать такой поток в методе onCreate(). а затем остановить его в методе onDestrov().

Видимый жизненный цикл операции происходит между вызовами методов onStart() и onStop(). В течение этого времени операция отображается на экране, где пользователь может взаимодействовать с ней. Например, метод onStop() вызывается в случае, когда запускается новая операция, а текущая больше не отображается. В промежутке между вызовами этих двух методов можно сохранить ресурсы, необходимые для отображения операции для пользователя. Например, можно зарегистрировать объект BroadcastReceiver в методе onStart() для отслеживания изменений, влияющих на пользовательский интерфейс, а затем отменить его регистрацию в методе onStop(), когда пользователь больше не видит отображаемого. В течение всего жизненного цикла операции система может несколько раз вызывать методы onStart() и onStop(), поскольку операция то отображается для пользователя, то скрывается от него.

Жизненный цикл операции, выполняемый на переднем плане, происходит между вызовами методов onResume() и onPause(). В течение этого времени операция выполняется на фоне всех прочих операций и отображается для пользователя. Операция может часто уходить в фоновый режим и выходить из него — например, метод onPause() вызывается при переходе устройства в спящий режим или при появлении диалогового окна. Поскольку переход в это состояние может выполняться довольно часто, код в этих двух методах должен быть легким, чтобы не допустить медленных переходов и не заставлять пользователя ждать.

Указывается, может ли система в любое время завершить процесс, содержащий операцию, после возвращения метода без выполнения другой строки кода операции. Для трех методов в этом столбце указано «Да»: (onPause(), onStop() и onDestroy()). Поскольку метод

onPause() является первым из этих трех после создания операции, метод onPause() является последним, который гарантированно будет вызван перед тем, как процесс можно будет завершить; если системе потребуется срочно восстановить память в случае аварийной ситуации, то методы onStop() и onDestroy() вызвать не удастся. Поэтому следует воспользоваться onPause(), чтобы записать критически важные данные (такие как правки пользователя) в хранилище постоянных данных. Однако следует внимательно подходить к выбору информации, которую необходимо сохранить во время выполнения метода onPause(), поскольку любая блокировка процедур в этом методе может вызвать блокирование перехода к следующей операции и тормозить работу пользователя.

Завершить операцию можно в период между возвратом onPause() и вызовом onResume(). Его снова не удастся завершить, пока снова не будет вызван и возвращен onPause().

Примечание. Операцию, которую технически невозможно завершить в соответствии с определением, по-прежнему может завершить система, однако это может произойти только в чрезвычайных ситуациях, когда нет другой возможности.

Задание

1. Разработать Android приложение, выполняющее ввод данных, вывод и редактирование в соответствии с вариантом. При запуске программы выполнять чтение данных из файла. При окончании работы программы сохранять данные в файл. Из изменения состояния жизненного цикла Activity записывать информацию в журнал. Выполнить кастомизацию элементов TextView и EditText. Выполнить запуск приложения на эмуляторе.

2. Переработать приложение, разработанное в предыдущем задании. Организовать разметку типа **ConstraintLayout**. Приложение должно содержать следующие элементы управления: Button, TextView, AutoCompleteTextView или MultiAutoCompleteTextView (текстовые поля с автозаполнением), ListView. Организовать навигацию с использованием списковых представлений (ListView), при этом при помощи адаптера осуществить связывание с массивами данных. Содержать следующие элементы по смыслу логики разрабатываемого приложения: ToggleButton, Checkbox, RadioButton, DatePicker и TimePicker, SeekBar. Выполнить кастомизацию графического компонента согласно варианту.

Варианты заданий:

Вариант № 1. Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Создать массив объектов. Вывести:

список студентов заданного факультета;

списки студентов для каждого факультета и курса;

список студентов, родившихся после заданного года;

список учебной группы.

Кастомизация компонента: добавить кнопку cancel к виджету AutoCompleteTextView. Функциональность такого виджета состоит в следующем. До тех пока в поле ввода ничего не введено кнопка cancel невидима. При вводе текста в поле AutoCompleteTextView отображается кнопка, с помощью которой можно удалить весь текст. По нажатию кнопки «Click to retrive value» извлекается текст из поля редактирования и отображается с помощью Toast.

Вариант № 2. Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета.

Создать массив объектов. Вывести:

список покупателей в алфавитном порядке;

список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Кастомизация компонента: создать кнопку, содержащую текст и изображение динамически.

Вариант № 3. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз.

Создать массив объектов. Вывести:

список пациентов, имеющих данный диагноз;

список пациентов, номер медицинской карты у которых находится в заданном интервале.

Кастомизация компонента: создать кнопку, содержащую текст и изображение динамически.

Вариант № 4. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки.

Создать массив объектов. Вывести:

список абитуриентов, имеющих неудовлетворительные оценки;

список абитуриентов, средний балл у которых выше заданного;

выбрать заданное число и абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Кастомизация компонента: организовать всплывающие подсказки - Toast. Выводить Toast с заданным временем.

Вариант № 5. Book: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет.

Создать массив объектов. Вывести:

список книг заданного автора;

список книг, выпущенных заданным издательством;

список книг, выпущенных после заданного года.

Кастомизация компонента: добавить кнопку cancel к виджету AutoCompleteTextView. Функциональность такого виджета состоит в следующем. До тех пока в поле ввода ничего не введено кнопка cancel невидима. При вводе текста в поле AutoCompleteTextView отображается кнопка, с помощью которой можно удалить весь текст. По нажатию кнопки «Click to retrive value» извлекается текст из поля редактирования и отображается с помощью Toast.

Вариант № 6. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации.

Создать массив объектов. Вывести:

список квартир, имеющих заданное число комнат;

список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;

список квартир, имеющих площадь, превосходящую заданную.

Кастомизация компонента: создать кнопку, содержащую текст и изображение динамически.

Вариант № 7. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров.

Создать массив объектов. Вывести:

сведения об абонентах, у которых время внутригородских разговоров превышает заданное;

сведения об абонентах, которые пользовались междугородной связью;

сведения об абонентах в алфавитном порядке.

Кастомизация компонента: добавить кнопку cancel к виджету AutoCompleteTextView. Функциональность такого виджета состоит в следующем. До тех пока в поле ввода ничего не введено кнопка cancel невидима. При вводе текста в поле AutoCompleteTextView отображается кнопка, с помощью которой можно удалить весь текст. По нажатию кнопки «Click to retrive value» извлекается текст из поля редактирования и отображается с помощью Toast.

Вариант № 8. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество.

Создать массив объектов. Вывести:

список товаров для заданного наименования;

список товаров для заданного наименования, цена которых не превосходит заданную;

список товаров, срок хранения которых больше заданного.

Кастомизация компонента: организовать всплывающие подсказки - Toast. Выводить Toast с заданным временем.

Вариант № 9. Train: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс).

Создать массив объектов. Вывести:

список поездов, следующих до заданного пункта назначения;

список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

Кастомизация компонента: создать кнопку, содержащую текст и изображение динамически.

Вариант № 10. Bus: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег.

Создать массив объектов. Вывести:

а) список автобусов для заданного номера маршрута;

список автобусов, которые эксплуатируются больше 10 лет;

список автобусов, пробег у которых больше 100000 км.

Кастомизация компонента: организовать всплывающие подсказки - Toast. Выводить Toast с заданным временем.

Вариант № 11. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер.

Создать массив объектов. Вывести:

список автомобилей заданной марки;

список автомобилей заданной модели, которые эксплуатируются больше и лет;

список автомобилей заданного года выпуска, цена которых больше указанной.

Кастомизация компонента: добавить кнопку cancel к виджету AutoCompleteTextView. Функциональность такого виджета состоит в следующем. До тех пока в поле ввода ничего не введено кнопка cancel невидима. При вводе текста в поле AutoCompleteTextView отображается кнопка, с помощью которой можно удалить весь текст. По нажатию кнопки «Click to retrive value» извлекается текст из поля редактирования и отображается с помощью Toast.

Вариант №12. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество.

Создать массив объектов. Вывести:

список товаров для заданного наименования;

список товаров для заданного наименования, цена которых не превосходит заданную;

список товаров, срок хранения которых больше заданного.

Кастомизация компонента: создать кнопку, содержащую текст и изображение динамически.

Требования к отчету

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- вид макета приложения в визуальном редакторе;
- вид приложения в эмуляторе;
- код файла ***activity_main.xml*** в редакторе кода и код файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
- выводы о проделанной работе.

Контрольные вопросы

1. Жизненный цикл Activity.
2. Регистрация событий жизненного цикла Activity.
3. Создание сообщений в журнале.

4. Использование LogCat.
5. Повороты и жизненный цикл активности.
6. Создание макета для альбомной ориентации.
7. Переопределение onSavelnstanceState(Bundle).
8. Из чего состоит пользовательский интерфейс?
9. С помощью каких объектов формируется графический интерфейс пользователя? Дайте характеристику.
10. Инициализация представлений.
11. Перечислите виды макетов. Дайте характеристику каждому из видов.
12. Дайте определение и характеристику компонентам графического интерфейса.

ЛАБОРАТОРНАЯ РАБОТА № 4

МОДУЛЬНАЯ СТРУКТУРА ПРИЛОЖЕНИЯ

Цель работы: изучить технологию разработки мобильного приложения с помощью использования фрагментов.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Фрагменты впервые появились в Android версии 3.0 (API уровня 11), главным образом, для обеспечения большей динамичности и гибкости пользовательских интерфейсов на больших экранах, например, у планшетов. Поскольку экраны планшетов гораздо больше, чем у смартфонов, они предоставляют больше возможностей для объединения и перестановки компонентов пользовательского интерфейса. Фрагменты позволяют делать это, избавляя разработчика от необходимости управлять сложными изменениями в иерархии представлений. Разбивая макет деятельности на фрагменты, разработчик получает возможность модифицировать внешний вид активности в ходе выполнения и сохранять эти изменения в стеке обратных переходов, которым управляет активность.

Фрагмент (класс Fragment) представляет поведение или часть пользовательского интерфейса в активности (класс Activity). Разработчик может объединить несколько фрагментов в одну активность для построения многопанельного пользовательского интерфейса и повторного использования фрагмента в нескольких активностях. Фрагмент можно рассматривать как модульную часть активности. Такая часть имеет свой жизненный цикл и самостоятельно обрабатывает события ввода. Кроме того, ее можно добавить или удалить непосредственно во время выполнения активности. Это нечто вроде вложенной активности, которую можно многократно использовать в различных активностях.

Фрагмент всегда должен быть встроен в активность, и на его жизненный цикл напрямую влияет жизненный цикл активности. Например, когда деятельность приостановлена, в том же состоянии находятся и все фрагменты внутри нее, а когда активность уничтожается, уничтожаются и все фрагменты. Однако пока активность выполняется (это соответствует состоянию ***возобновлена*** жизненного цикла), можно манипулировать каждым фрагментом независимо, например, добавлять или удалять их. Когда разработчик выполняет такие транзакции с фрагментами, он может также добавить их в стек

переходов, которым управляет деятельность. Каждый элемент стека переходов в активности является записью выполненной транзакции с фрагментом. Стек переходов позволяет пользователю вернуть транзакцию с фрагментом (выполнить навигацию в обратном направлении), нажимая кнопку ***Назад.***

Когда фрагмент добавлен как часть разметки активности, он находится в объекте ViewGroup внутри иерархии представлений деятельности и определяет собственный макет представлений. Разработчик может вставить фрагмент в макет активности двумя способами. Для этого следует объявить фрагмент в файле разметки деятельности как элемент **<fragment>** или добавить его в существующий объект ViewGroup в коде приложения. Впрочем, фрагмент не обязан быть частью разметки активности. Можно использовать фрагмент без интерфейса в качестве невидимого рабочего потока активности.

Фрагмент, как и активность, связывается с макетом. Если внимательно подойти к его проектированию, для управления всеми аспектами интерфейса может использоваться код Java. Если код фрагмента содержит все необходимое для управления его макетом, вероятность того, что фрагмент можно будет повторно использовать в других частях приложения, значительно возрастает.

Следует разрабатывать каждый фрагмент как модульный и повторно используемый компонент активности. Поскольку каждый фрагмент определяет собственный макет и собственное поведение со своими обратными вызовами жизненного цикла, разработчик может включить один фрагмент в несколько активностей. Поэтому он должен предусмотреть повторное использование фрагмента и не допускать, чтобы один фрагмент непосредственно манипулировал другим.

Это особенно важно, потому что модульность фрагментов позволяет изменять их сочетания в соответствии с различными размерами экранов. Если приложение должно работать и на планшетах, и на смартфонах, можно повторно использовать фрагменты в различных конфигурациях разметки, чтобы оптимизировать взаимодействие с пользователем в зависимости от доступного размера экрана.

Например, на смартфоне может возникнуть необходимость в разделении фрагментов для предоставления однопанельного пользовательского интерфейса, если разработчику не удается поместить более одного фрагмента в одну активность.

Для создания фрагмента необходимо создать подкласс класса Fragment (или его существующего подкласса). Класс Fragment имеет код, во многом схожий с кодом Activity. Он содержит методы обрат-

ного вызова, аналогичные методам деятельности, такие как onCreate(), onStart(), onPause() и onStop(). *На практике, если требуется преобразовать существующее приложение Android так, чтобы в нем использовались фрагменты, достаточно просто переместить код из методов обратного вызова активности в соответствующие методы обратного вызова фрагмента.*

Задание

Переработать приложение, разработанное в лабораторной работе №3. Организовать приложение с помощью возможностей использования фрагментов. Приложение должно содержать как минимум три фрагмента, один из которых должен быть вложенным в любой другой из двух оставшихся фрагментов. Использовать разные макеты для различных устройств (с большим и маленьким экраном), учитывать повороты устройства. Создать диалоговые окна оповещений.

Требования к отчету

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- вид макета приложения в визуальном редакторе;
- вид приложения в эмуляторе;
- код файла ***activity_main.xml*** в редакторе кода и код файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
- выводы о проделанной работе.

Контрольные вопросы

1. Что представляет собой фрагмент?
2. Философия проектирования фрагмента.
3. Создание фрагмента.
4. Добавление фрагмента в активность.
5. Добавление фрагмента, не имеющего пользовательского интерфейса.
6. Управление фрагментами.
7. Выполнение транзакций с фрагментами.
8. Взаимодействие с активностью.
9. Создание обратного вызова события для активности.
10. Управление жизненным циклом фрагмента.
11. Согласование с жизненным циклом активности.

ЛАБОРАТОРНАЯ РАБОТА № 5

СОЗДАНИЕ ПАНЕЛИ ДЕЙСТВИЙ

Цель работы: изучить методы создания панели действий.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

При создании приложения в основном используются экраны трех видов:

Экраны верхнего уровня

Как правило, это первая активность приложения, которую видит пользователь.

Экраны категорий

На экранах категорий выводятся данные, относящиеся к определенной категории, — часто в виде списка. На них пользователь может перейти к экранам детализации/редактирования.

Экраны детализации/редактирования

На этих экранах выводится подробная информация о конкретных записях, пользователь может редактировать существующие или создавать новые записи.

Применение действий для навигации.

В приложениях Android активные средства навигации обычно добавляются на панель действий. Эта панель часто располагается в верхней части многих активностей. На ней размещаются наиболее часто используемые действия, поэтому обычно кнопки на панели лучше всего описываются всевозможными глаголами: «Создать», «Найти», «Изменить» и т. д.

Панель действий выполняет в приложениях несколько функций:

1. Используется для вывода имени приложения или активности, чтобы пользователь знал, в какой точке приложения он находится. Например, в почтовом клиенте на панели действий может выводиться текущая папка (Входящие, Корзина и т. д.).

2. Привлекает внимание пользователя к размещенным на ней ключевым действиям — например, публикации контента или выполнению поиска.

3. Используется для перехода к другим активностям, в которых выполняются нужные действия.

Чтобы добавить в приложение панель действий, необходимо выбрать **тему**, включающую панель действий. **Тема** (theme) пред-

ставляет собой визуальный стиль, применяемый ко всей активности или приложению и обеспечивающий целостность его внешнего вида и поведения. Тема управляет такими аспектами, как цвет фона активности и панели действий, оформление текста и т. д. Android содержит набор встроенных тем, которые можно использовать.

Если приложения должны работать в API уровня 11 и выше, необходимо добавить панель действий, применяя класс темы *Theme.Holo* или один из его субклассов. В большинстве случаев следует использовать именно этот вариант. Для API уровня 21 и выше имеется дополнительная возможность использования одной из более современных тем *Theme.Material*.

С течением времени в Android появляются новые возможности. Но что, если использовать новейшие виджеты Android на устройстве двух-трехлетней давности? В таких случаях можно воспользоваться библиотеками поддержки Android. В основном эти библиотеки обеспечивают обратную совместимость, то есть возможность использования новых возможностей Android на старых устройствах.

Некоторые возможности Android реализованы только в библиотеках поддержки, и если использовать их в своих приложениях, без библиотек поддержки не обойтись. Например, DrawerLayout API позволяет создать навигационную выдвижную панель, которую можно «вытащить» из-за края экран.

Пакет библиотек поддержки Android содержит несколько библиотек, каждая из которых ориентирована на определенный базовый уровень API и включает конкретную функциональность. В имени библиотеки поддержки содержится минимальный номер версии Android, с которым совместима библиотека. Для каждой из этих библиотек выходят обновления, в которых реализуются новые возможности и исправления ошибок.

Классы библиотек поддержки хранятся в пакетах с именами *android.support.v**. Например, классы библиотеки v4 хранятся в пакете android.support.v4.

Выдвижная панель реализуется при помощи особой разновидности макетов *DrawerLayout*. DrawerLayout управляет двумя представлениями:

1. Представление для основного контента. Обычно здесь используется фрейм FrameLayout, чтобы можно было отображать и переключать фрагменты.

2. Представление для выдвижной панели (обычно списковое представление ListView).

По умолчанию DrawerLayout отображает представление, которое почти не отличается от обычной активности:

Значок выдвижной панели. Необходимо щелкнуть по значку или провести пальцем, чтобы открыть панель (рисунок 13).

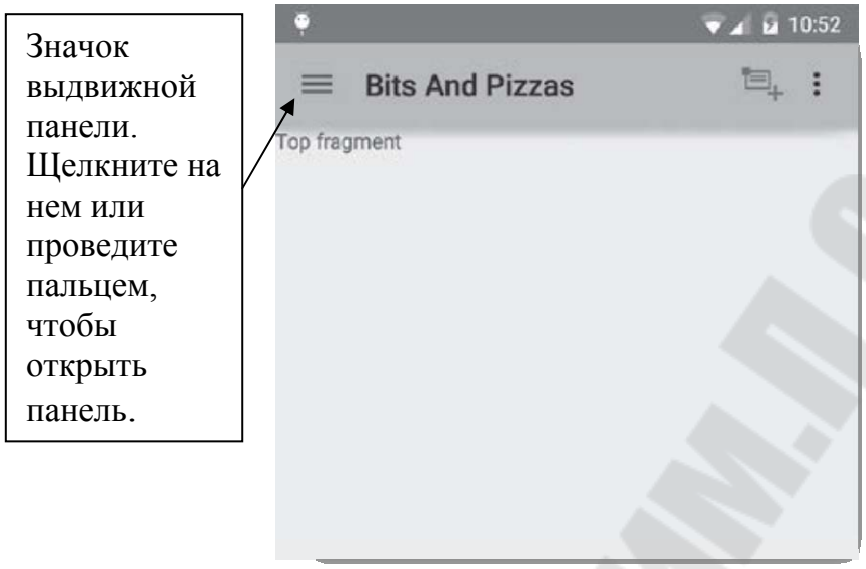

Рисунок 13 – Значок выдвижной панели

Если щелкнуть на значке выдвижной панели или провести пальцем от края экрана, представление выдвижной панели «накрывает» основную информацию (рисунок 14):

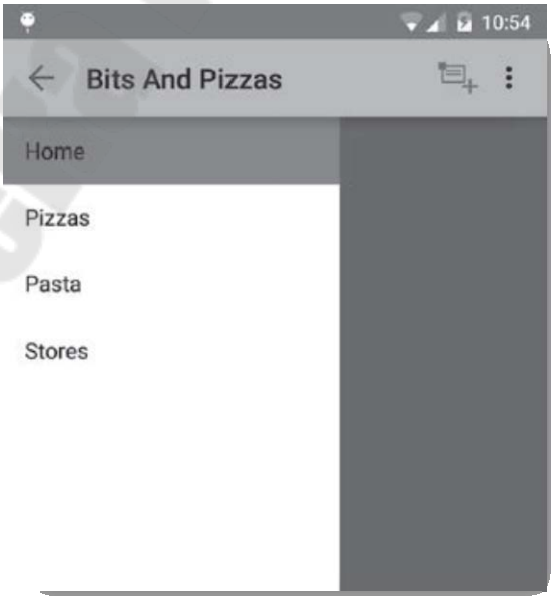

Рисунок 14 – Вид выдвижной панели

Задание

Переработать приложение, разработанное в лабораторной работе №4 следующим образом:

1) Создать панель действий. Панель действий должна выполнять следующие функции: вывода имени приложения и активити; расположения ключевых действий приложения (активити действия/редактирования).

2) Организовать приложение так, чтобы была добавлена на панель действий кнопка «Вверх». Переходы по кнопке Вверх основаны исключительно на иерархической структуре приложения, при организации учитывать разный уровень API, применить темы для разного уровня API.

4) Организовать навигационную выдвижную панель, содержащую ссылки на основные навигационные точки приложения.

Требования к отчету

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- вид макета приложения в визуальном редакторе;
- вид приложения в эмуляторе;
- код файла ***activity_main.xml*** в редакторе кода и код файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
- выводы о проделанной работе.

Контрольные вопросы

1. Применение действий для навигации.
2. Функции панели действий.
3. Использование тем.
4. Добавление элементов действий на панель действий.
5. Передача информации с панели действий.
6. Навигация с кнопкой Вверх.
7. Организация выдвижной панели (drawers).

ЛАБОРАТОРНАЯ РАБОТА № 6

ДОЛГОВРЕМЕННОЙ ХРАНЕНИЕ ДАННЫХ

Цель работы: изучить методы программирования долговременного хранения данных мобильного приложения и Internet коммуникаций.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Android автоматически создает для каждого приложения папку, в которой хранятся базы данных этого приложения. Когда создается база данных для приложения *Starbuzz*, она будет храниться в следующей папке:

/data/data/com.hfad.starbuzz/databases

В этой папке приложение может хранить несколько баз данных. Каждая база данных состоит из двух файлов. Имя первого — файла базы данных — соответствует имени базы данных: например, starbuzz. Это основной файл баз данных SQLite, в нем хранятся все данные. Второй файл — файл журнала. Его имя состоит из имени базы данных и суффикса -journal — например, starbuzz-journal. В файле журнала хранится информация обо всех изменениях, внесенных в базу данных. Если в работе с данными возникнет проблема, Android использует данные журнала для отмены (или отката) последних изменений.

Android включает классы SQLite. Помощник SQLite создается расширением класса SQLiteOpenHelper. Он предоставляет средства для создания и управления базами данных.

Класс SQLiteDatabase предоставляет доступ к базе данных. Его можно сравнить с классом SQLConnection в JDBC.

Класс Cursor предназначен для чтения и записи в базу данных. Его можно сравнить с классом ResultSet в JDBC.

Создание базы данных.

Помощник SQLite управляет базой данных. Класс SQLiteOpenHelper упрощает задачи создания и сопровождения баз данных.

Рассмотрим некоторые типичные задачи, в решении которых способствует помощник SQLite:

1. Создание базы данных

При первой установке приложения файл базы данных не существует. Помощник SQLite проследит за тем, чтобы файл базы данных был создан с правильным именем и с правильной структурой таблиц.

2. Обеспечение доступа к базе данных

Приложению не обязательно знать все подробности о том, где хранится файл базы данных. Помощник SQLite предоставляет удобный объект, представляющий базу данных, и приложение работает с базой через этот объект — тогда, когда сочтет нужным.

3. Сопровождение баз данных

Может случиться так, что структура базы данных изменится со временем. Помощник SQLite преобразует старую версию в новую с учетом самых последних изменений в структуре базы данных.

Чтобы создать помощника SQLite, необходимо написать класс, расширяющий SQLiteOpenHelper. При этом необходимо переопределить методы onCreate() и onUpgrade()

Например, чтобы создать таблицу DRINK, необходимо выдать соответствующую команду на языке SQL. Команда CREATE_TABLE сообщает, какие столбцы должны присутствовать в таблице, и данные какого типа должны в этих столбцах храниться. Столбец _id является первичным ключом таблицы, а специальное ключевое слово AUTOINCREMENT означает, что при занесении в таблицу новой строки SQLite автоматически сгенерирует для нее уникальный целочисленный идентификатор.

Помощник SQLite отвечает за то, чтобы база данных SQLite была создана в момент ее первого использования. Сначала на устройстве создается пустая база данных, после чего вызывается метод onCreate() помощника SQLite.

При вызове метода onCreate() передается объект SQLiteDatabase.

Класс SQLiteDatabase содержит несколько методов для вставки, обновления и удаления данных.

Если потребуется заполнить таблицу SQLite данными, то необходимо воспользоваться методом insert() класса SQLiteDatabase. Этот метод вставляет данные в базу и возвращает идентификатор записи. Если метод не смог создать запись, он возвращает значение –1.

Чтобы использовать метод insert(), необходимо указать таблицу и вставляемые значения. Для определения вставляемых значений создается объект ContentValues, в котором данные сохраняются в виде пар «имя/значение». Для добавления пар «имя/значение» в объект ContentValues используется метод put() этого объекта.

Для обновления существующей информации в SQLite используется метод update() класса SQLiteDatabase. Этот метод вносит изменения в записи, хранящиеся в базе данных, и возвращает количество обновленных записей. Чтобы использовать метод update(), необходимо указать таблицу, обновляемые значения и условия их обновления.

Метод delete() класса SQLiteDatabase работает по тому же принципу, как и только что рассмотренный метод update().

Задание

Переработать приложение, разработанное в лабораторной работе №5 в соответствии с вариантом следующим образом: организовать функционал, выполняющий сохранение всех данных приложения в БД и запросов. Создать отдельный метод для обновления БД «updateMyDatabase()». Выполнить запуск приложения на эмуляторе.

Варианты заданий:

Вариант № 1. **Student**: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон. **Group**: id_group, название, курс, факультет.

Вывести: a) список студентов заданного факультета; b) списки студентов для каждого факультета и курса; c) список студентов, родившихся после заданного года; d) список учебной группы. e) список студентов, отчисленных с определенного факультета за последний год.

Вариант № 2. **Customer**: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. **Credit**: id_credit, наименование, срок погашения, сумма.

Вывести: a) список покупателей в алфавитном порядке; b) список покупателей, у которых номер кредитной карточки находится в заданном интервале; c) список покупателей, у которых оформлен кредит на определенную сумму и на определенный срок; d) список покупателей, у которых сумма задолженности превышает 25% кредита.

Вариант № 3. **Patient**: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. **Hospital**: id_Hospital, наименование **Department**: id_depertament, наименование.

Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты у которых находится в заданном интервале; c) список пациентов, поступивших в определен-

ную больницу в определенное отделение за последний месяц; d) список пациентов и их диагнозов, которые проходили в определенном отделении больницы лечение в настоящую дату.

Вариант № 4. **Abiturient**: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. **Courses**: id_ courses, наименование, вуз.

Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число и абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл); d) Список иногородних абитуриентов; e) Список абитуриентов, проходивших подготовительные курсы определенного вида, определенного вуза.

Вариант № 5. **Book**: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет. **Publishing**: id_ publishing, наименование издательства, год основания.

Вывести: a) список книг заданного автора; b) список книг, выпущенных заданным издательством; c) список книг, выпущенных после заданного года; d) список издательств, выпустивших книги, превышающих заданную цену; e) список издательств, выпустивших книги определенного автора.

Вариант № 6. **House**: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. **Construction_company**: id_ construction company, название.

Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную; d) Список домов, построенных определенной строительной компанией и имеющих определенный срок эксплуатации; e) Список строительных компаний, построивших определенный тип зданий.

Вариант № 7. **Phone**: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Время городских и междугородных разговоров. **Telephone_company**: id_ telephone company, название.

Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах,

которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке; d) Сведения об абонентах, пользующихся услугами определенной телефонной компании.

Вариант № 8. **Product**: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. **Wholesale_company**: id_ wholesale_company, наименование, направление.

Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного; d) Список товаров, приобретенных у определенной оптовой компании.

Вариант № 9. **Train**: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс).

Вывести: a) список поездов, следующих до заданного пункта назначения; b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа; c) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места; d) Список поездов, отправляющихся со станции в течении часа и в настоящий момент; e) Сведения о количестве мест определенного типа по определенному рейсу.

Вариант № 10. **Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. **Autopark**: id_ autopark, название.

Вывести: a) список автобусов для заданного номера маршрута; 4 b) список автобусов, которые эксплуатируются больше 10 лет; c) список автобусов, пробег у которых больше 100000 км; d) Список автобусов, приобретенных определенным автопарком

Вариант № 11. **Car**: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. **Car_show**: id_ car show, название, сказываемые услуги.

Вывести: a) список автомобилей заданной марки; b) список автомобилей заданной модели, которые эксплуатируются больше 10 лет; c) список автомобилей заданного года выпуска, цена которых больше указанной; d) Список автомобилей, обслуживаемых в опреде-

ленном автосалоне в определенный период времени по определенной услуге.

Вариант № 12. **Product**: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество.

Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного; d) Список товаров определенного наименования; e) Список товаров определенного производителя; f) Список товаров, находящихся сейчас на складе.

Требования к отчету

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- вид макета приложения в визуальном редакторе;
- вид приложения в эмуляторе;
- код файла ***activity_main.xml*** в редакторе кода и код файлов, в которые вносились изменения после автоматической генерации проекта Android studio;
- выводы о проделанной работе.

Контрольные вопросы

1. Долговременное хранение данных мобильного париложения.
2. Локальные базы данных и SQLite.
3. Хранение серий и позиций в БД.
4. Запрос списка серий из БД.
5. Вывод списка серий с использованием CursorAdapter.
6. Создание новых серий.
7. Работа с существующими сериями.
8. Использование Internet коммуникаций в устройствах Android

СПИСОК ИСТОЧНИКОВ

1. Майер, Р. Android 4. Программирование приложений для планшетных компьютеров и смартфонов : [перевод с английского] / Рето Майер. – Москва : Эксмо, 2014. – 814 с.

2. Android. Программирование для профессионалов / Б. Харди [и др.] ; пер. с англ. Е. Матвеев. – 2- изд. – Санкт-Петербург [и др.] : Питер, 2016. – 636 с.

3. Дейтел, П. Android для разработчиков / Пол Дейтел, Харви Дейтел, АлександерУолд ; [пер. с англ. Е. Матвеев]. – 3-е изд.. – Санкт-Петербург [и др.] : Питер, 2017. – 512 с.

Богданова Наталья Сергеевна

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

Практикум по выполнению лабораторных работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 13.10.21.

Рег. № 44Е.
http://www.gstu.by