ЛАБОРАТОРНАЯ РАБОТА 5

Тема: Использование модели MVC в Android-приложениях

Теоретическая часть

Приложения Android строятся на базе архитектуры, называемой «Модель-Представление-Контроллер» (Model-View-Controller, MVC). Согласно принципов этой архитектуры каждый объект приложения должен быть объектом модели, объектом представления или объектом контроллера.

- Объект модели содержит данные приложения и «бизнес-логику». Классы модели обычно проектируются для моделирования сущностей, с которыми работает приложение, — пользователь, счет в банке, продукт в магазине, фотография на сервере, вопрос «да/нет» и т.п. Объекты модели ничего не знают о пользовательском интерфейсе, их единственной целью является хранение данных и управление ими. Классы моделей обычно создаются разработчиком для конкретной задачи, а все объекты модели в приложении составляют его уровень модели.
- Объекты представлений отображаются на экране и реагируют на определенные действия пользователя, например, касание, фокусирование и прочее. Если что-либо выводится на экран, значит, это объект представления. В Android имеется достаточно широкий набор настраиваемых классов представлений. Кроме того, разработчик также может создавать собственные классы представлений. Объекты представления в приложении образуют уровень представления.
- Объекты контроллеров связывают объекты представления и модели, и содержат «логику приложения». Контроллеры реагируют на различные события, инициируемые объектами представлений, и управляют потоками данных между объектами модели и уровнем представления. В Android контроллер обычно представляется субклассом Activity, Fragment или Service.

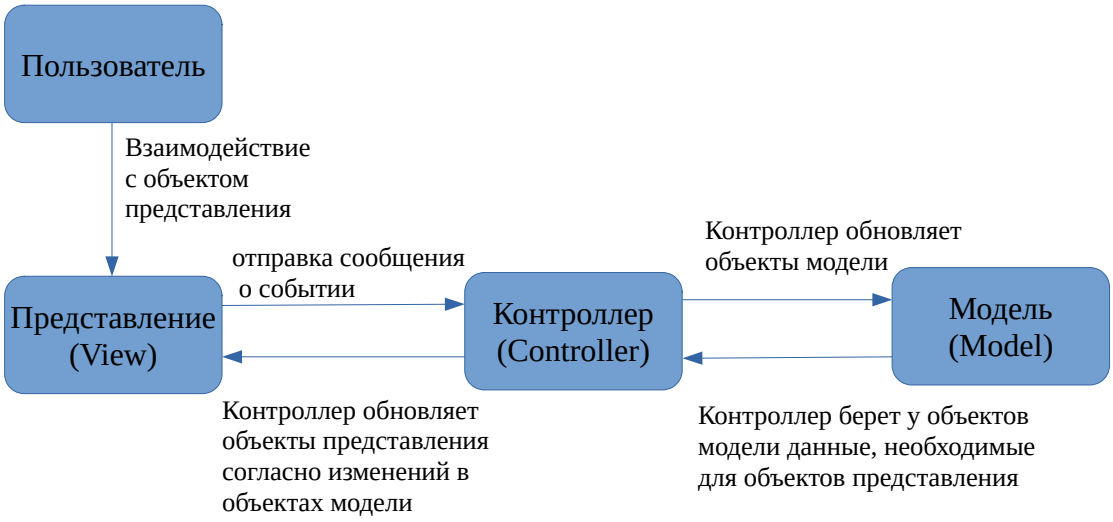

Рис.1. Взаимодействия в MVC

Объекты модели и представлений не взаимодействуют друг с другом напрямую - в любом взаимодействии участвуют «посредники»-контроллеры, получающие сообщения от одних объектов и передающие инструкции другим.

Функциональность приложения может увеличиваться до тех пор, пока не станет слишком сложной для понимания и сопровождения. Разделение кода на классы упрощает проектирование и

понимание приложения в целом, дает возможность разработчику мыслить в контексте классов, а не отдельных переменных и методов. Аналогичным образом разделение классов на уровни модели, представления и контроллера упрощает проектирование и понимание приложения, и предоставляет возможность мыслить в контексте уровней, а не отдельных классов. Добавление или изменение классов выполняется в контексте определенного уровня и на соответствующем уровне, что улучшает архитектуру и ее понимание, а значит последующее сопровождение и возможные модификации.

Также концепции MVC упрощают повторное использование классов. Класс с ограниченными обязанностями лучше подходит для повторного использования, чем класс, в котором реализовано и модель данных, и всю логику задачи, и представление для пользовательского интерфейса.

Рассмотрим простое Android-приложение, реализующее задачу опроса с ответами «Да/Нет» [1]. Приложение, имеет простой интерфейс пользователя, на котором представляется вопрос и несколько кнопок, которые позволяют дать ответ «Да/Нет» («Правильно/Неправильно») на вопрос и сделать переход между вопросами.

Приложение состоит из активности (activity) и макета (layout):

- Активность представлена экземпляром Activity — классом из Android SDK. Она отвечает за взаимодействие пользователя с информацией на экране. Чтобы реализовать необходимую приложению функциональность создаются подклассы Activity. В простом приложении может быть достаточно одного подкласса, а для сложного может потребоваться несколько.
- Макет определяет набор объектов пользовательского интерфейса и их расположение на экране. Макет формируется из определений на языке XML. Каждое определение используется для создания объекта, выводимого на экране, например, кнопка или текст. Как правило, первичный макет именуется с суффиксом activity_ и расширением xml, а также для имени используется имя проекта.

На сегодняшний день официальной системой для разработки Android-приложений является Android Studio [2]. Запустите Android Studio, создайте новый проект и задайте ему имя.

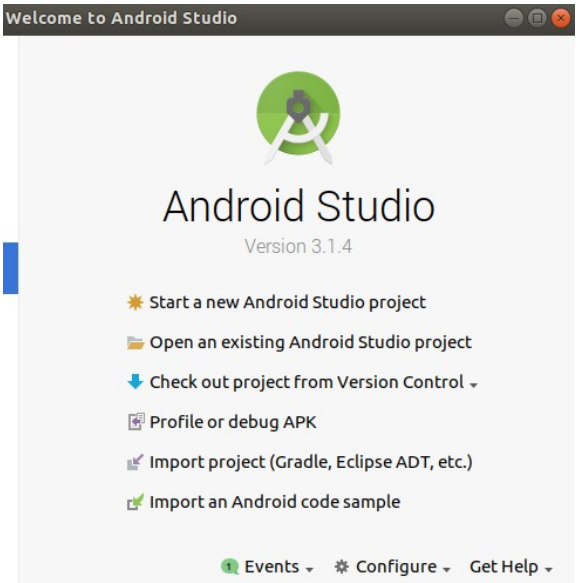

Рис.2. Начало работы в Android Studio

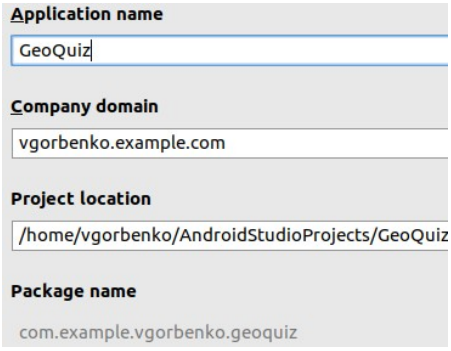

Рис.3. Новый проект в Android Studio с именем приложения GeoQuiz

Выбор формфактора (Phone and Tablet) и минимального SDK можно оставить по-умолчанию. В следующем окне диалога «Добавление в проект Activity» делаем выбор шаблона EmptyActivity. В окне конфигурирования Activity задаем имена Activity Name и Layout Name, которые взаимосвязаны между собой частью именования. Например, задав в Activity Name имя QuizActivity в поле Layout Name получим activity_quiz.xml. Установку флажков оставляем по умолчанию. После нажатия кнопки Finish Android Studio создает и организует проект. В состав проекта входит значительное количество файлов, часть которых можно увидеть при помощи панели навигации (по-умолчанию находится слева).

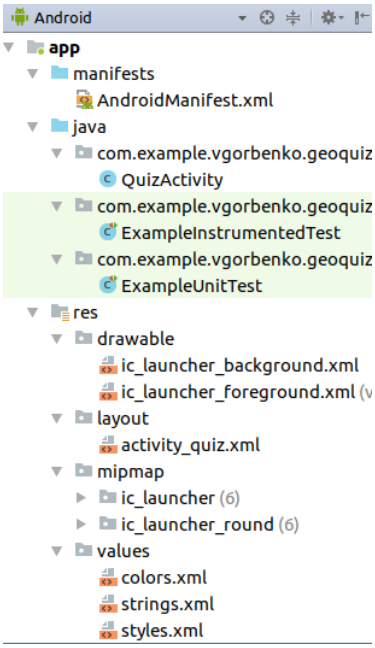

Рис.4. Панель навигации проекта в Android Studio

В действительности, даже начальная структура простого проекта Android-приложения гораздо более сложная, достаточно переключить навигацию в режим просмотра Project

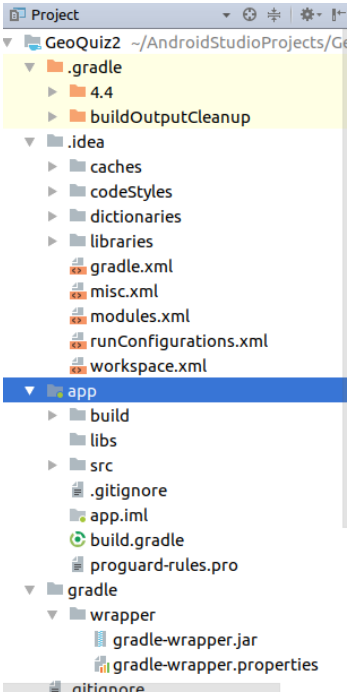

Рис.5. Навигация проекта в режиме Project

По умолчанию новый проект открывается в режиме навигации Android, а в панели редактора открываются два файла — в нашем проекте QuizActivity.java и activity_quiz.xml. В контексте модели MVC activity_quiz.xml относится к уровню представления, а QuizActivity.java — к уровню контроллера.

Рис.6. В редакторе по умолчанию открываются два файла проекта

Редактирование этих файлов позволяет внести необходимые изменения в пользовательский интерфейс приложения и логику обработки действий пользователя. Уровень модели данных реализуется дополнительными классами, которые создаются и добавляются в проект по мере необходимости. Например, для того чтобы в создаваемом проекте можно было оперировать с некоторой выборкой вопросов нужно создать дополнительный класс, например, Question. Для этого, на панели навигации находим пакет, в котором находится файл QuizActivity.java (рис.7) и добавляем в него новый класс (File -> New -> Java Class)

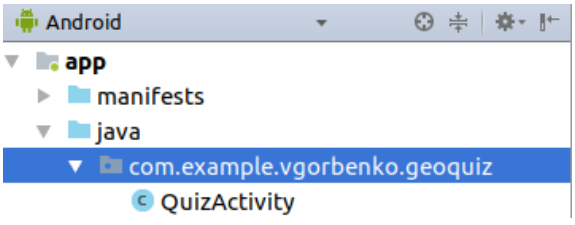

Рис.7. В этот пакет будет добавлен файл с классом модели данных

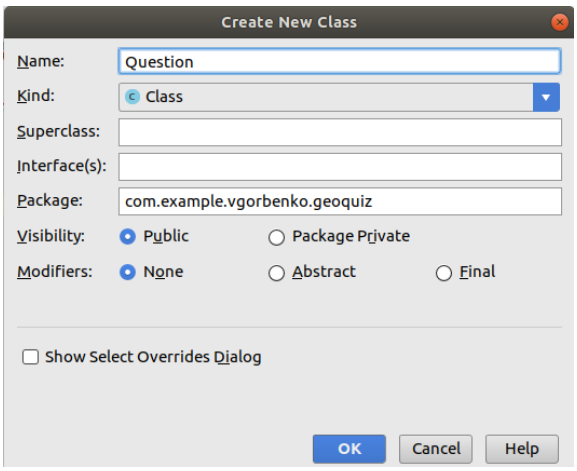

Рис.8. Создание нового класса Question

После нажатия кнопки Ок создается пустой класс. Очевидно, что этот класс в проекте должен позволять создавать объекты, в которых будут вопросы и ответы на них. Поэтому в простом варианте класс должен содержать два поля — строковое (для вопроса) и логическое (для ответа). Однако, в Android-приложениях принято строковые значения содержать в файлах ресурсов, например, в res/values/strings.xml. Каждый элемент в этом файле получит свой числовой идентификатор. Поэтому обращение к строковому ресурсу в рамках проекта будет осуществляться по его идентификатору, а его конкретное значение будет храниться в определенном файле ресурсов. Такой механизм хранения и использования строковых данных позволяет без смены программного кода проекта осуществлять его локализацию или корректировку строковых значений. Поэтому в класс Question добавляются числовое и логическое поля и конструктор, как показано ниже

```
public class Question {
    private int mTextResId;
    private boolean mAnswerTrue;
    public Question(int textResId, boolean answerTrue) {
        mTextResId = textResId;
        mAnswerTrue = answerTrue;
    }
}
```

Для работы с полями в класс необходимо добавить get- и set- методы:

```
public class Question {

    . . . . . .

    public int getTextResId() {
        return mTextResId;
    }
    public void setTextResId(int textResId) {
        mTextResId = textResId;
    }
    public boolean isAnswerTrue() {
        return mAnswerTrue;
    }
    public void setAnswerTrue(boolean answerTrue) {
        mAnswerTrue = answerTrue;
    }

}
```

Теперь в проекте есть все три составляющие MVC.

Уровень представления создаваемого проекта определяется макетом пользовательского интерфейса, описание которого находится в файле activity_quiz.xml. Структурные элементы, из которых составляется пользовательский интерфейс, называют виджетами. Виджет может

выводить текст или графику, взаимодействовать с пользователем или размещать другие виджеты на экране. Кнопки, текстовые поля, флажки и т.п. представляют разновидности виджетов. Кроме того имеются виджеты, которые не имеют видимого графического представления, но используются для организации расположения видимых виджетов. Android SDK включает множество виджетов, которые можно настраивать для получения нужного оформления и поведения. Каждый виджет является экземпляром класса View или одного из его подклассов, например таких, как TextView или Button.

Для ускорения процесса разработки пользовательского интерфейса в Android Studio имеется как редактор текста файла макета, так и визуальный редактор интерфейса. Между редакторами можно переключаться в процессе разработки пользовательского интерфейса, а любые изменения в текстовой части приводят к изменениям в создаваемом интерфейсе и наоборот — любые манипуляции с графическими представлениями виджетов приводят к соответствующим изменениям в текстовой части файла макета. Для корректного отображения графического представления макета необходимо проверить правильность ссылки в иерархии классов в файле res/values/styles.xml. Она должна выглядеть как

```
<!-- Base application theme. →
<style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
```

Если на панели Preview не отображаются виджеты макета, следует откорректировать значение поля `parent` как показано выше.

Для создаваемого приложения файл activity_quiz.xml может быть реализован следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".QuizActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:id="@+id/TextQuiz"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14sp" />
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <Button
                android:id="@+id/buttonTrue"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/button_true" />
            <Button
                android:id="@+id/buttonFalse"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/button_false" />
        </LinearLayout>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:orientation="horizontal">
            <Button
                android:id="@+id/buttonPrev"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/button_prev" />
            <Button
                android:id="@+id/buttonNext"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/button_next" />
        </LinearLayout>
    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Виджеты `LinearLayout` позволяют организовать иерархию в макете и сгруппировать вложенные в них виджеты. Их параметр `orientation` позволяет указать размещение виджетов в группе или задать иерархию структуры. Другие виджеты в этом файле - `TextView` и `Button` описывают конкретные элементы графического интерфейса пользователя. Надписи на кнопках определяются при помощи идентификаторов текста, например, `@string/button_next` , значение которого задано в файле ресурсов res/values/strings.xml. Пример содержания этого файла представлен ниже

```
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="question_1">Столица Турции - Константинополь</string>
    <string name="question_2">Тихий океан больше, чем Атлантический</string>
    <string name="question_3">Суэцкий канал соединяет Красное море\n с Индийским океаном</string>
    <string name="question_4">Река Нил начинается в Египте</string>
    <string name="button_true">Это правда</string>
    <string name="button_false">Это ложь</string>
    <string name="button_next">Следующий</string>
    <string name="button_prev">Предыдущий</string>
    <string name="toast_true">Вы ответили правильно! </string>
    <string name="toast_false">Вы ответили неправильно!</string>>
</resources>
```

В этом файле также содержатся вопросы и сообщения о правильности ответов.

Согласно MVC связывание модели данных и представлений осуществляется при помощи логики контроллера, которая, в создаваемом проекте, представляется в файле QuizActivity.java. Его содержание может быть следующим

```
package com.example.vgorbenko.geoquiz;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

    public class QuizActivity extends AppCompatActivity {
        private Button mTrueButton;
        private Button mFalseButton;
        private Button mNextButton;
        private Button mPrevButton;
        private TextView mTextQuiz;
        private static final String TAG="QuizActivity";
        private static final String KEY_INDEX = "index";

        private Question[] mQuestionBank = new Question[]{
                new Question(R.string.question_1, false),
                new Question(R.string.question_2, true),
                new Question(R.string.question_3, false),
                new Question(R.string.question_4, false)
        };
        private int mCurrentIndex = 0;

        @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
        mTextQuiz = (TextView)findViewById(R.id.TextQuiz);
        updateQuestion();
        addListenerOnClickButtonTrueFalse();
    }

    public void addListenerOnClickButtonTrueFalse(){
        mTrueButton = (Button)findViewById(R.id.buttonTrue);
        mFalseButton = (Button)findViewById(R.id.buttonFalse);
        mNextButton = (Button)findViewById(R.id.buttonNext);
        mPrevButton = (Button)findViewById(R.id.buttonPrev);
        mTextQuiz = (TextView)findViewById(R.id.TextQuiz);

        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                checkAnswer(true);
            }
        });

        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                checkAnswer(false);
            }
        });

        mNextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
                updateQuestion();
            }
        });

        mPrevButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mCurrentIndex = (mCurrentIndex - 1) % mQuestionBank.length;
                if(mCurrentIndex<0)mCurrentIndex=mQuestionBank.length-1;
                updateQuestion();
            }
        });
    }
    private void updateQuestion(){
        int question = mQuestionBank[mCurrentIndex].getTextResId();
        mTextQuiz.setText(question);
    }
    private void checkAnswer(boolean userPressedTrue){
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
        int messageResId = 0;
        if(userPressedTrue == answerIsTrue){
            messageResId = R.string.toast_true;
        }else{
            messageResId = R.string.toast_false;
        }
        Toast.makeText(this,messageResId,Toast.LENGTH_SHORT).show();
    }
}
```

Метод `onCreate(Bundle savedInstanceState)` вызывается при создании Activity , поэтому, все, что находится в этом методе будет выполнено при запуске приложения. А именно два метода `updateQuestion()` и `addListenerOnClickButtonTrueFalse()` позволяют вывести первый вопрос и обеспечить функциональностью кнопки интерфейса: ответы на вопросы «Правильно/Неправильно» и смену вопросов в двух направлениях «Следующий/Предыдущий».

Практическое задание

1. Изучите приведенное выше приложение, реализуйте его в Android Studio и проверьте его работу.

2. Внесите изменения в приложение так, чтобы и формулировки вопросов, и ответы на них извлекались из файла ресурсов strings.xml.

3. Измените всплывающее уведомление так, чтобы оно отображалось в верхней, а не в нижней части экрана. Для изменения способа отображения уведомления используйте метод setGravity класса Toast. Выберите режим Gravity.TOP. Дополнительная информация находится в документации разработчика:
developer.android.com/reference/android/widget/Toast.html#setGravity(int, int, int).

4. Внесите изменения в приложение так, чтобы после ответа на вопрос вместо формулировки вопроса выводился ответ на него. При этом, если ответ пользователем был дан правильно, то ответ должен выводиться зеленым цветом, а если ответ дан неправильный — красным.

5. Сравните работу приложения на эмуляторе и на реальном устройстве.

6. Измените тип виджетов кнопок на графический ImageButton (вместо используемого в приложении Button). Виджет ImageButton является производным от ImageView, в отличие от виджета Button, производного от TextView. Атрибут text кнопки СЛЕДУЮЩИЙ теперь нужно заменить на атрибут src ImageView с указанием на источник графического представления:

```
<Button ImageButton
    android:id="@+id/next_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/arrow_right"
/>
```

Графические файлы размещаются в соответствующий каталог проекта res/drawable с именем, которое будет использоваться для ссылки. Файлы могут иметь расширение .png, .jpg или .gif. Любому такому файлу, добавленному в папку res/drawable, автоматически назначается идентификатор ресурса. Имена файлов должны быть записаны в нижнем регистре и не могут содержать пробелов.

7. Расширьте функциональность приложения добавлением подсчета баллов за правильные ответы на вопросы. Внесите изменения в код так, чтобы вопросы, ответ на которые был уже дан, не показывались при прокрутке вперед/назад.

8. Расширьте базу вопросов и сделайте их выборку для опроса случайной и ограниченной по количеству.

9. Подготовьте отчет по всем пунктам практического задания.

Список литературы

1. Phillips B. Android Programming: The Big Nerd Ranch Guide / B. Phillips, Ch. Stewart, K. Marsicano: 3rd Edition. — Big Nerd Ranch, 2017. — 720 p.
2. Android Studio // https://developer.android.com/studio/