Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине “Технологии разработки приложений для мобильных устройств”

на тему:

«Мобильное приложение для организации персональных финансов»

Выполнил:

студент группы ПИ20-1 факультета информационных технологий и анализа больших данных

Журавлев К.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

доцент, к.т.н. Крахмалев О.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2022 г

Введение

Тема моей курсовой работы – «Мобильное приложение для организации персональных финансов». В современном мире мобильные приложения подобного типа и функционала являются вполне востребованными, поскольку личные денежные средства, а также контроль за их движениями являются неотъемлемой частью повседневной жизни каждого гражданина. Очень важно прокачивать свою финансовую грамотность, всегда знать сколько, когда или в каких обстоятельствах была потрачена та или иная сумма денег. В этом и помогают приложения для учета личных финансов, в них человек может отмечать все свои траты в настоящем времени, вдобавок всегда знать, на что именно были потрачены денежные средства, а также выделить себе цель на будущее – накопить на что-нибудь. В действительности, когда человек будет в курсе того, на что уходят его деньги, он сможет более ответственно подойти к вопросу их траты, отказаться от лишних, ненужных транзакций. В наши дни большинство банковских приложений снабжены похожими функциями, как приложения для учета финансов, но в меньшем объеме. В них вся история трат оформляется в автоматическом режиме и человек так же имеет возможность просмотреть когда, где и на что он потратил деньги, но в большинстве случаев это не будет раскрыто достаточно детально. Так что, на самом деле, основная функция приложения для финансов встроена в большинство банковских мобильных приложений, так еще и выполняется в автоматическом режиме, но все же намного лучше будет потратить не так много своего времени, заполнить расходы в приложении для учета персональных финансов, и иметь более детальную картину происходящего, а значит иметь больше контроля над потоком своих денежных средств.

Глава I.I Обзор IDE Android Studio

Приложение для моей курсовой работы выполнено в IDE Android Studio на языке программирования Java.

Android Studio – это IDE, созданная для разработки приложений для Android устройств, выпущенная компанией Google в 2013 году. Android Studio предназначена для создания приложений только под операционную систему android.

Изначально, Android Studio поддерживала только язык программирования Java, давно применяемый для работы с приложениями на Android, но в 2017 список ЯП пополнил новенький Kotlin, который также опередил Java по популярности. Несмотря на это, всё еще предоставляется с комфортом разрабатывать приложения для Android на Java, хоть и найти решения некоторых проблем будет достаточно проблематично, так как методы разработки устаревают, а основная масса разработчиков пользуется Kotlin и новые вопросы генерирует по нему.

Справедливо отметить, что Android Studio постоянно получает новые обновления, эмуляторы устройств пополняются новыми видами, а процессы разработки упрощаются.

Глава I.II Преимущества и недостатки Android Studio

Главным недостатком среды разработки Android Studio со дня выхода и по сей день является низкая производительность приложения и эмулятора в большей части.

По своему опыту могу сказать, что не один раз возникали трудности с написанием мобильных приложений, причём на различных устройствах (ноутбук, компьютер). Изначально вообще не работали виртуальные эмуляторы на ноутбуке, сколько бы раз я не переустанавливал и сколько бы обучающих видео по установке не пересмотрел. Насколько я понимаю, это связано с видеокартой AMD, установленной на моем ноутбуке, и в принципе на ощутимой части современных ноутбуков. На стационарном компьютере помощнее и с видеокартой другой компании таких проблем не возникало, но даже не смотря на мощь компьютера в сравнении с ноутбуков, иногда среда разработки очень долго прогружала какие-либо действия или просто зависала.

Постоянные обновления, конечно, можно считать преимуществом, но они тоже не всегда могут идти на пользу. Достаточно рассмотреть последнее изменение Android Studio - теперь окно эмулятора закреплено в интерфейсе самого приложения, хотя раньше эмулятор открывался отдельным окном. Из-за чего теперь при работе над приложением приходится решать, что тебе важнее, поработать над кодом, или в пикселях рассматривать эмулятор. Кажется незначительной проблемой, но когда уже сотый раз проверяешь написанный код, то бесконечно менять размеры окна может хорошенько надоесть.

Также среда разработки Android Studio занимает немало места в памяти компьютера, что конечно относительно для каждого отдельного пользователя, но в моем случае с ноутбуком это довольно критично.

Из плюсов – довольно обширный набор функций, удобен в использовании, а также быстрое обновление приложения при изменении кода.

Глава II. Разработка

Основные файлы приложения:

MainActivity.java

activity\_main.xml

list\_item.xml

Обо всём по порядку.

Выполнение приложения по умолчанию начинается с класса MainActivity, который изначально открыт в нашей среде разработки Android Studio.

Вообще все, что происходит на экране Android-смартфона, определяется выполняемой в данный момент активностью. Класс MainActivity является стандартным классом java, который начинается с определения пакета данного класса: package com.example.finance;

Следом идет объявление сторонних библиотек, функционал которых использован в нашем MainActivity.

Затем идёт объявление самого класса, который по умолчанию наследуется от класса AppCompatActivity – по сути, отдельная страница приложения. Также в MainActivity изначально находится единственный метод onCreate(), в котором и создается интерфейс нашего приложения.

public class MainActivity extends AppCompatActivity {

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);

В метод setContentView() передается ресурс, содержащий разметку для нашего графического интерфейса.

В данном случае, это файл activity\_main.xml (res/layout/activity\_main.xml), который открыт в среде разработки по умолчанию.

int mBudget = 0;  
class Wasted {  
 int summ;  
 String comment;  
  
 public Wasted(int summ, String comment) {  
 this.summ = summ;  
 this.comment = comment;  
 }

Создаем класс с помощью генератора, так как каждое событие по трате денег в нашем приложении мы считаем за отдельную сущность, и таким образом, с помощью этого класса, с ними будет проще работать и брать в ListView.

class WastedAdapter extends BaseAdapter {  
 @Override  
 public int getCount() {  
 return wastedList.size();  
 }  
  
 @Override  
 public Object getItem(int i) {  
 return wastedList.get(i);  
 }  
  
 @Override  
 public long getItemId(int i) {  
 return i;  
 }  
  
 @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 if (view == null) {  
 view = ((LayoutInflater)getBaseContext().getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*)).inflate(R.layout.*list\_item*, viewGroup, false);  
 }  
  
 ((TextView) view.findViewById(R.id.*waste\_text*))  
 .setText(String.*valueOf*(wastedList.get(i).summ));  
 ((TextView) view.findViewById(R.id.*waste\_exp*))  
 .setText(wastedList.get(i).comment);  
 return view;  
 }  
}

Выше описываем Адаптер на основе базового адаптера. Он нам понадобится для ListView, в нижнем части этого кода расписано, что мы берём sum и comment, и у нас выводится сумма денег и пояснение к тратам в ListView. Его мы также создаем вручную, особо ничего в нём не меняя, разве что в конце описывается то, что будут нести в себе ячейки в ListView.

budget\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 final EditText budget\_text = new EditText(MainActivity.this);  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setView(budget\_text);  
 builder.setPositiveButton("Ок", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 change\_budget(Integer.*parseInt*(budget\_text.getText().toString()));  
 }  
 });  
 builder.create();  
 builder.show();  
 }  
});

Приведенная выше часть кода – создание обработчика событий для кнопки «Ввести бюджет». Мы вызываем всплывающее окно ввода и парсим введенные в него данные, которые в конечном итоге превращаем в значение, которое в следствии высвечивается на экране.

EditText waste\_text = findViewById(R.id.*summ*);  
 EditText explanation = findViewById(R.id.*explane*);  
 Button wasted\_button = findViewById(R.id.*waste\_btn*);  
 wasted\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 int wasted\_value = Integer.*parseInt*(waste\_text.getText().toString());  
 String waste\_explanation = explanation.getText().toString();  
 wastedList.add(new Wasted(wasted\_value, waste\_explanation));  
 change\_budget(mBudget - wasted\_value);  
 wastedAdapter.notifyDataSetChanged();  
 }  
 });  
}  
void change\_budget (int new\_budget) {  
 mBudget = new\_budget;  
 budget\_view.setText("Текущий Бюджет: " + mBudget);

В этой части кода описывается кнопка «Ввести расходы». В обработчике событий таким же образом парсим введенную трату, и, здесь же, в обработчике событий описываем создание ячейки в ListView и меняем значение бюджета (вычитаем из бюджета расходы).

Change\_budget соответственно отвечает за поле, где высвечивается бюджет.

Чуть выше я уже затрагивал адаптеры, сейчас о них чуть подробнее. Адаптер нам нужен для упрощенного связывания данных с элементом управления. Адаптеры используются при работе с виджетами, активностями и так далее, в нашем случае это ListView.

Для создания списка расходов в моей программе используется ListView. В свою очередь, сам ListView состоит из какого-то конкретного количества элементов TextView, идущих друг за другом по порядку. Не всегда мы имеем понимание, сколько этих TextView понадобится в нашем списке, и эту проблему решает адаптер, который самостоятельно берёт предоставляемые пользователем данные и размещает их в ListView списке по порядку, так же самостоятельно создавая нужные элементы TextView и помещая в них введенный текст. Существует множество готовых адаптеров для различных ситуаций, но конкретно для случая моей программы, был написан свой адаптер на основе BaseAdapter.

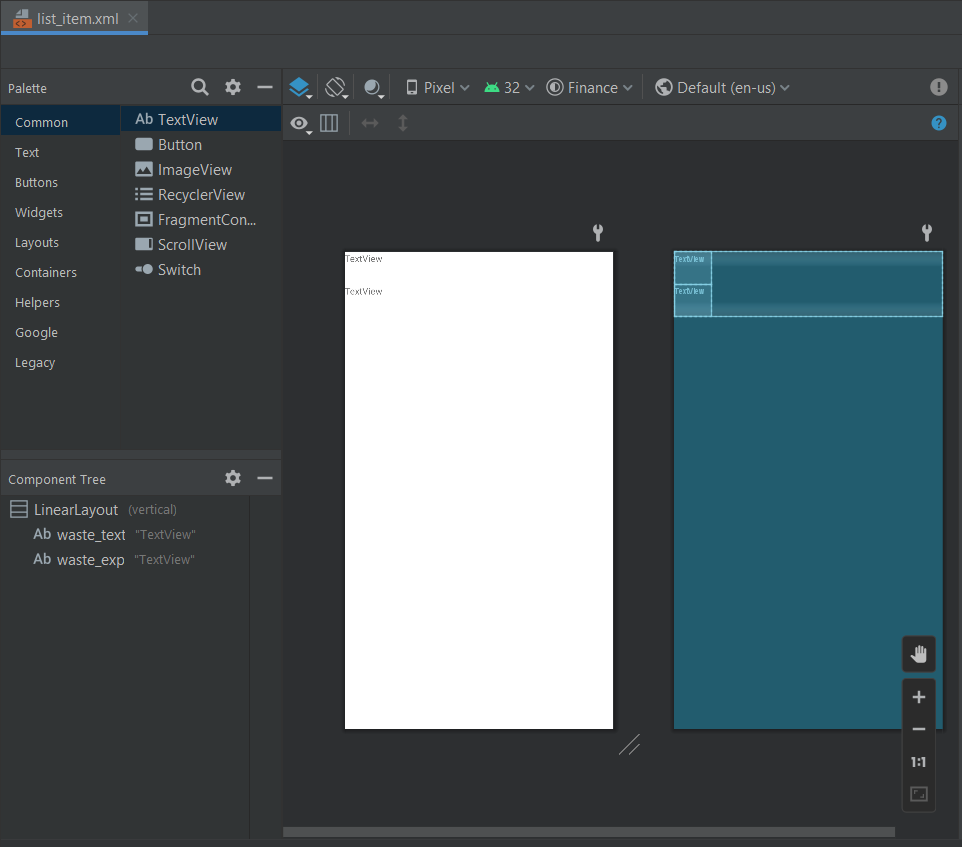
Структура списка описана в файле list\_item.xml

Рисунок 2.1 list\_item.xml

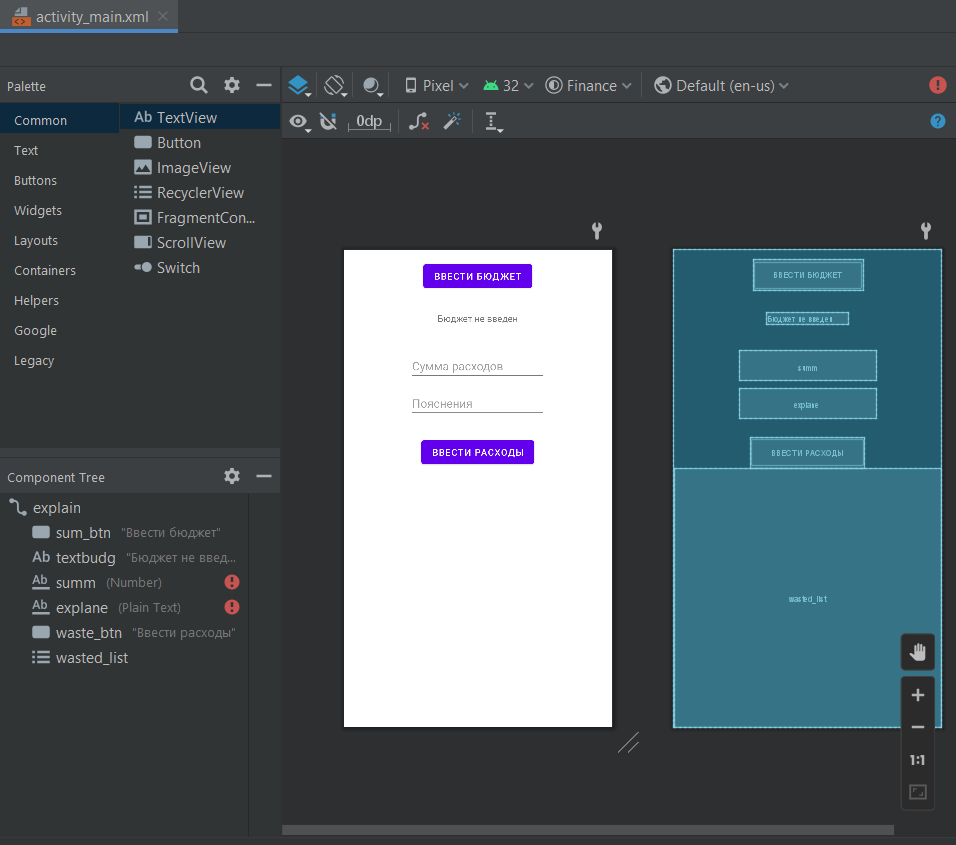
А файл activity\_main.xml отвечает за общую стилистику приложения

Рисунок 2.2 activity\_main.xml

В конечном итоге, приложение выглядит следующим образом: Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.3 Главный экран приложения

Здесь мы можем увидеть кнопку «Ввести бюджет», после нажатия на которую открывается поле для ввода этого самого бюджета, который далее будет показываться чуть ниже кнопки с его вводом (до первого ввода бюджета там будет располагаться надпись «бюджет не введен»). Кнопка «Ввести бюджет» не прибавляет введенное значение к уже существующему бюджету, а конкретно устанавливает отображаемую сумму.

Чуть ниже располагаются два поля, куда пользователь должен вводить сумму расходов и описание этих расходов соответственно. Предусмотрен сценарий ввода пользователем значений, отличных от числовых в поле ввода «Сумма расходов» - ввести что-то кроме цифр просто не получится. По нажатию на кнопку «Ввести расходы» запись добавляется в лист, который расположен в оставшейся нижней половине экрана, а из бюджета вычитается соответствующая сумма. При достаточном заполнении списка предусмотрен скролл.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.4 Скролл в списке

В случае выхода из бюджета, его состояние пойдет в отрицательное значение.

Заключение

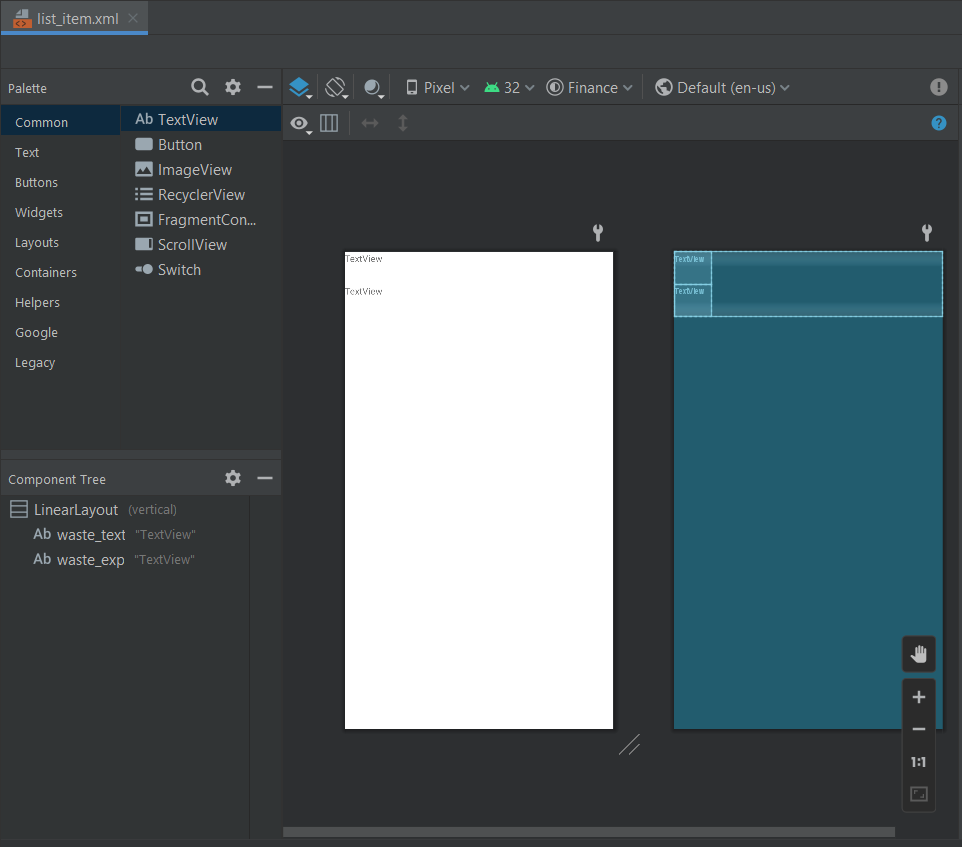
В заключение могу сказать, что созданное мною приложение естественно не претендует на конкурентность с другими приложениями этого предназначения, данное приложение имеет свои недостатки, такие как вылет приложения, если числовое значение бюджета становится слишком большим (тут надо бы пользователю не дать возможности уйти в огромный минус), или такой же вылет при попытке ввода в бюджет значений, отличных от цифровых.

Также есть огромный простор для развития приложения, как в дизайнерском плане, так и в функциональном.

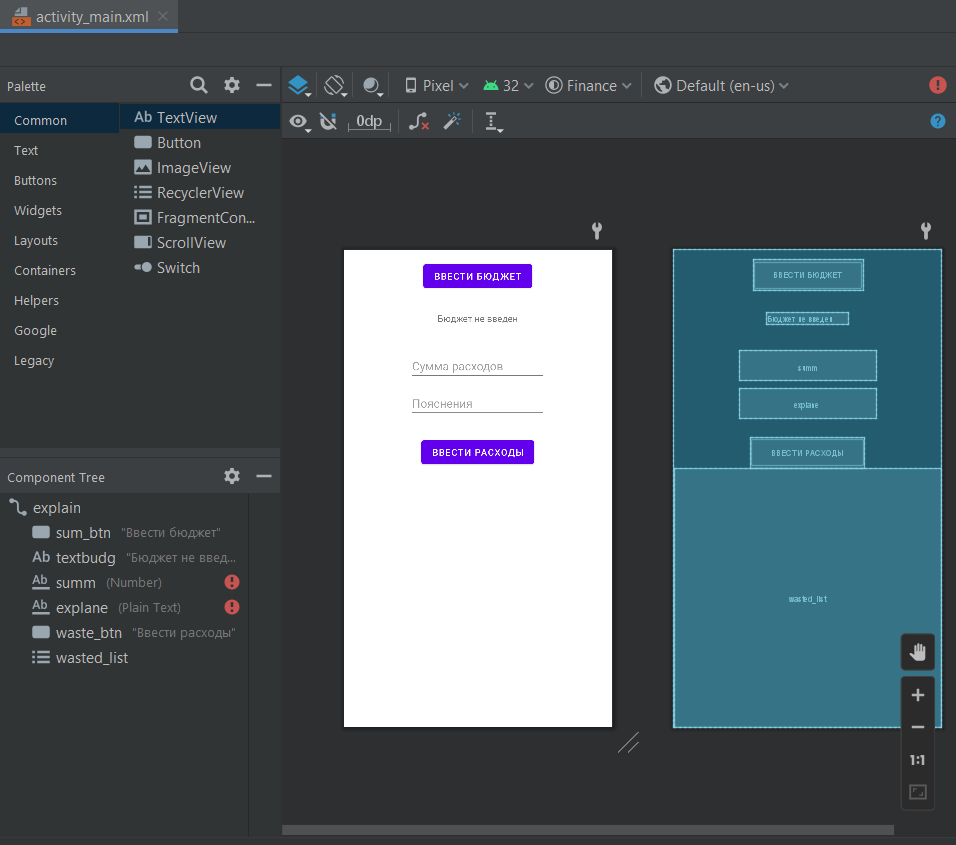
**Список использованной литературы**

1. metanit.com/java/android/
2. startandroid.ru/ru/uroki.html
3. koroteev.site/md/
4. Б. Филлипс, Б. Харди | Android. Программирование для профессионалов (2016) 640с
5. developer.android.com/studio
6. Р. Майер. Android Программирование приложений для планшетных компьютеров и смартфонов. Пер. с англ. : “Эксмо 2011. 671с

Приложение



Приложение 1. list\_item.xml



Приложение 2. Activity\_main.xml

Изображение выглядит как текст

Автоматически созданное описание

Приложение 3. Главный экран приложения

Изображение выглядит как текст

Автоматически созданное описание

Приложение 4. Скролл в списке