**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 18**

по дисциплине «Разработка мобильных приложений»

**Тема: Unit 2 Пользовательский интерфейс “** **Эспрессо для тестирования пользовательского интерфейса”**

Студент группы РИЗ-300028у: А.С. Севостьянов

Преподаватель: С.И.Тимошенко,

доц., к.т.н.

**Екатеринбург 2022**

содержание

[1 Постановка задачи 3](#_Toc118041527)

[2 Результаты проделанной работы 4](#_Toc118041528)

[3 Анализ полученных результатов 5](#_Toc118041529)

## 1 Постановка задачи

Изменить проект, чтобы создать тесты Espresso. Протестировать ввод и вывод текста в приложении. Протестировать щелчок по элементу spinner и проверьте его вывод. Записать тест Эспрессо в RecyclerView.

.

## Результаты проделанной работы

Программный код, файл MainActivity.

package com.example.android.recyclerview;  
  
import android.os.Bundle;  
import android.support.design.widget.FloatingActionButton;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.LinearLayoutManager;  
import android.support.v7.widget.RecyclerView;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
  
import java.util.LinkedList;  
  
*/\*\*  
 Реализует базовый RecyclerView, который отображает список сгенерированных слов.  
 Щелчок по элементу помечает его как щелкнутый.  
 Нажатие кнопки fab добавляет новое слово в список.  
 \*/*public class MainActivity extends AppCompatActivity {  
  
 private final LinkedList<String> mWordList = new LinkedList<>();  
  
 private RecyclerView mRecyclerView;  
 private WordListAdapter mAdapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 FloatingActionButton fab = findViewById(R.id.*fab*);  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 int wordListSize = mWordList.size();  
 // Add a new word to the wordList.  
 mWordList.addLast("+ Word " + wordListSize);  
 // Notify the adapter, that the data has changed.  
 mRecyclerView.getAdapter().notifyItemInserted(wordListSize);  
 // Scroll to the bottom.  
 mRecyclerView.smoothScrollToPosition(wordListSize);  
 }  
 });  
  
 // Поместите исходные данные в список слов.  
 for (int i = 0; i < 20; i++) {  
 mWordList.addLast("Word " + i);  
 }  
  
 // Создайте представление утилизатора.  
 mRecyclerView = findViewById(R.id.*recyclerview*);  
 // Создайте адаптер и укажите данные, которые будут отображаться.  
 mAdapter = new WordListAdapter(this, mWordList);  
 // Подключите адаптер с видом на утилизатор.  
 mRecyclerView.setAdapter(mAdapter);  
 // Предоставьте виду recycler менеджер компоновки по умолчанию.  
 mRecyclerView.setLayoutManager(new LinearLayoutManager(this));  
 }  
  
 */\*\*  
 \* Расширяет меню и добавляет элементы на панель действий, если она присутствует.  
 \*/* @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Раздуть меню; это добавляет элементы на панель действий, если она присутствует.  
 getMenuInflater().inflate(R.menu.*menu\_main*, menu);  
 return true;  
 }  
  
 */\*\*  
 \* Обрабатывает щелчки по элементам панели приложений.  
 \*/* @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
// Обработайте элемент панели действий, нажав здесь. Панель действий будет  
 // автоматически обрабатывать нажатия на кнопку "Домой"/"Вверх", пока  
 // как вы указываете родительское действие в AndroidManifest.xml .  
 int id = item.getItemId();  
  
 // Этот комментарий подавляет предупреждение Android Studio об упрощении  
 // операторы возврата.  
 //отсутствие проверки, упрощаемое утверждение  
 if (id == R.id.*action\_settings*) {  
 return true;  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
}

Программный код файл WordListAdapter.

package com.example.android.recyclerview;  
  
import android.content.Context;  
import android.support.v7.widget.RecyclerView;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import java.util.LinkedList;  
  
*/\*\*  
 \* Показывает, как реализовать простой адаптер для RecyclerView.  
 \* Демонстрирует, как добавить обработчик щелчков для каждого элемента в ViewHolder.  
 \*/*public class WordListAdapter extends  
 RecyclerView.Adapter<WordListAdapter.WordViewHolder> {  
  
 private final LinkedList<String> mWordList;  
 private final LayoutInflater mInflater;  
  
 class WordViewHolder extends RecyclerView.ViewHolder  
 implements View.OnClickListener {  
 public final TextView wordItemView;  
 final WordListAdapter mAdapter;  
  
 */\*\*  
 \* Создает новый пользовательский держатель представления для удержания представления для отображения в  
 \* RecyclerView.  
 \*/* public WordViewHolder(View itemView, WordListAdapter adapter) {  
 super(itemView);  
 wordItemView = itemView.findViewById(R.id.*word*);  
 this.mAdapter = adapter;  
 itemView.setOnClickListener(this);  
 }  
  
 @Override  
 public void onClick(View view) {  
 // Получить положение элемента, на который был сделан щелчок.  
 int mPosition = getLayoutPosition();  
  
 // Используйте это для доступа к затронутому элементу в mWordList.  
 String element = mWordList.get(mPosition);  
 // Измените мир в списке слов.  
  
 mWordList.set(mPosition, "Clicked! " + element);  
// Уведомить адаптер о том, что данные изменились, чтобы он мог  
// обновить RecyclerView для отображения данных.  
 mAdapter.notifyDataSetChanged();  
 }  
 }  
  
 public WordListAdapter(Context context, LinkedList<String> wordList) {  
 mInflater = LayoutInflater.*from*(context);  
 this.mWordList = wordList;  
 }  
 @Override  
 public WordListAdapter.WordViewHolder onCreateViewHolder(ViewGroup parent,  
 int viewType) {  
 // Раздуть представление элемента.  
 View mItemView = mInflater.inflate(  
 R.layout.*wordlist\_item*, parent, false);  
 return new WordViewHolder(mItemView, this);  
 }  
  
 @Override  
 public void onBindViewHolder(WordListAdapter.WordViewHolder holder,  
 int position) {  
 // Извлеките данные для этой позиции.  
 String mCurrent = mWordList.get(position);  
 // Добавьте данные в держатель представления.  
 holder.wordItemView.setText(mCurrent);  
 }  
  
 */\*\*  
 \* Возвращает общее количество элементов в наборе данных, хранящемся адаптером.  
 \*/* @Override  
 public int getItemCount() {  
 return mWordList.size();  
 }  
}

На рисунке 1 показано начальное состояние приложения, выведен циклом список слов, и есть действующая кнопка Fab с помощью которой мы может добавлять новое слово в конец списка.

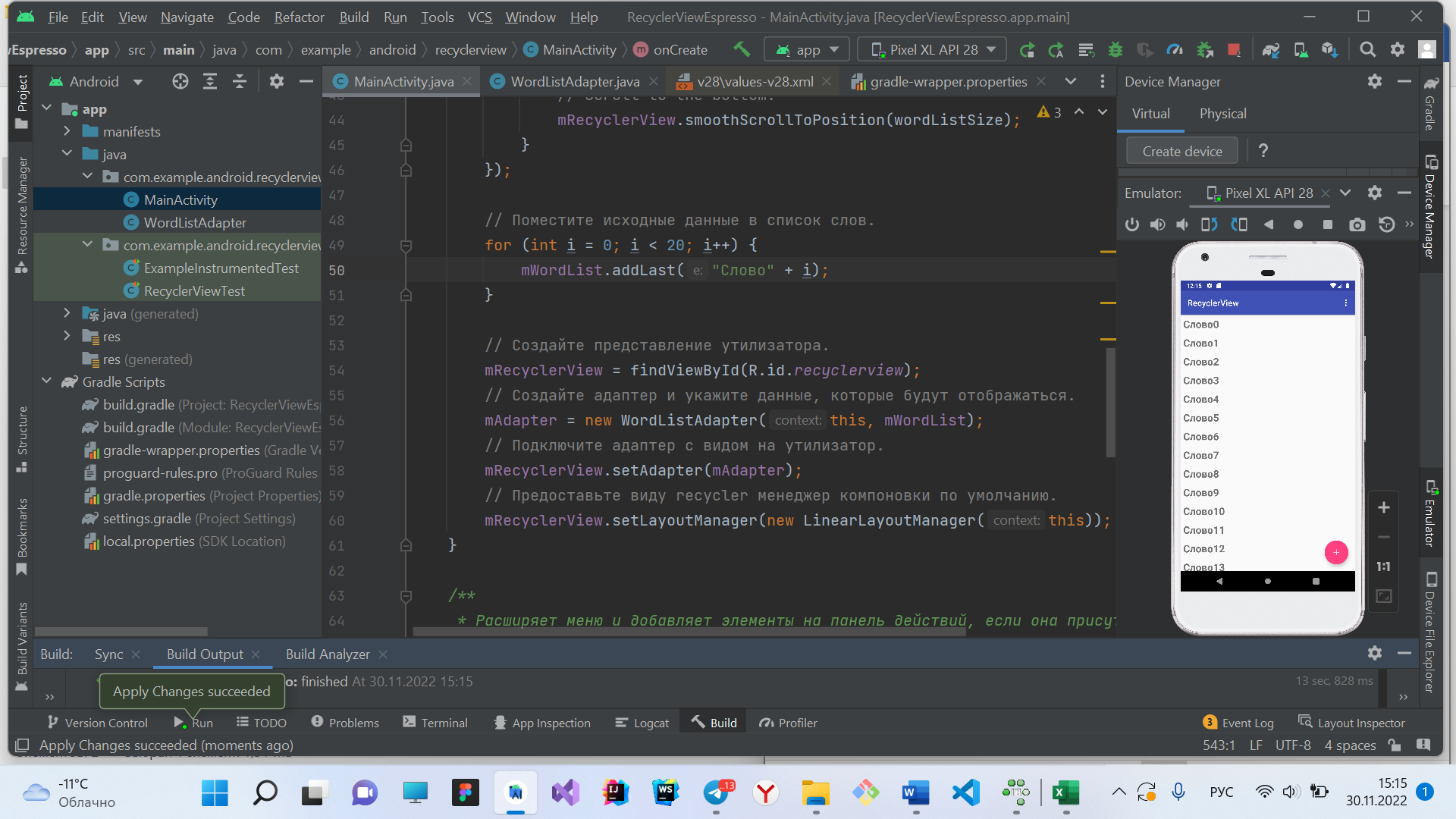


Рисунок 1 – Список слов

На рисунке 2 показ результат работы кнопки Fab.

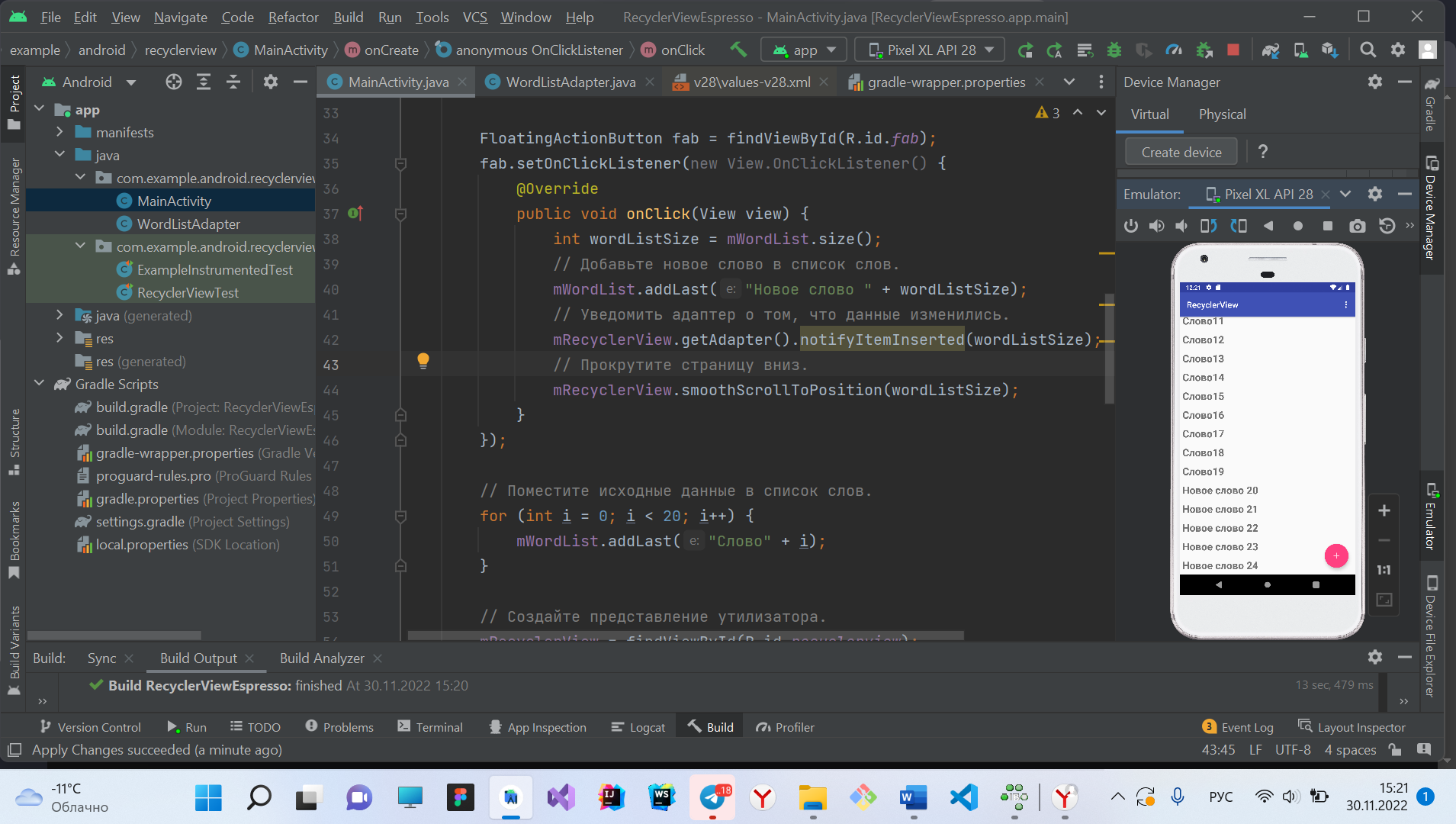


Рисунок 2 – Результат работы кнопки Fab

## **Анализ полученных результатов**

В ходе изучения данной лабораторной работы я научился, настраивать эспрессо в проекте приложения. Писать тесты на эспрессо. Проверять пользовательский ввод и проверить правильность вывода.

Как использовать функцию Record Espresso Test в Android Studio