|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА — Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе**

по дисциплине

«Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-33-21 | Макаров П.А. |
| **Проверил:**  Ассистент | Новичков Д.Е. |

2023 г.

**СОДЕРЖАНИЕ**

[1. СТРУКТУРА ПРОЕКТА 3](#_Toc136645860)

[2. РЕАЛИЗАЦИЯ ПРОЕКТА 5](#_Toc136645861)

[2.1 Жизненный цикл Activity 5](#_Toc136645862)

[2.2 Всплывающие уведомления (Toast) 10](#_Toc136645863)

[2.3 Логирование 11](#_Toc136645864)

[2.4 Код проекта 12](#_Toc136645865)

[ВЫВОД 14](#_Toc136645866)

# **СТРУКТУРА ПРОЕКТА**

Файл AndroidManifest.xml является одним из самых важных в проекте Android приложения. В нем содержится информация о пакетах приложения, компонентах, таких как Activity, Service и других. Файл AndroidManifest.xml выполняет следующие задачи:

* Описывает разрешения, необходимые приложению для работы, например на доступ к камере или внутреннему хранилищу;
* Определяет как будут запускаться приложение, например, какое Activity должно быть запущено при нажатии на иконку в списке приложений.

Папка java содержит исходный код приложения. Классы могут быть расположены в различных пакетах, но обязательно внутри папки java.

В папке res расположены все используемые приложением ресурсы, включая изображения, различные xml файлы, анимации, звуковые файлы и многие другие. Внутри папки res эти все ресурсы распределены по своим папкам:

Папка drawable содержит файлы с изображениями, которые будет использоваться в приложении;

Папка layout располагает xml файлами, которые используются для построения пользовательского интерфейса Android приложения;

В mipmap папке хранят только значки приложения. values хранит те xml файлы, в которых определяются простые значения типа строк, массивов, целых чисел, размерностей, цветов и стилей.

Скрипты Gradle используются для автоматизации сборки проекта. Android Studio выполняет сборку приложения в фоновом режиме без какого-либо вмешательства со стороны разработчика. Этот процесс сборки осуществляется с использованием системы Gradle — инструментария для автоматической сборки с помощью набора конфигурационных файлов. Gradle скрипты написаны на языке groove.

Ниже представлена структура реализуемого проекта (Рисунок 1).

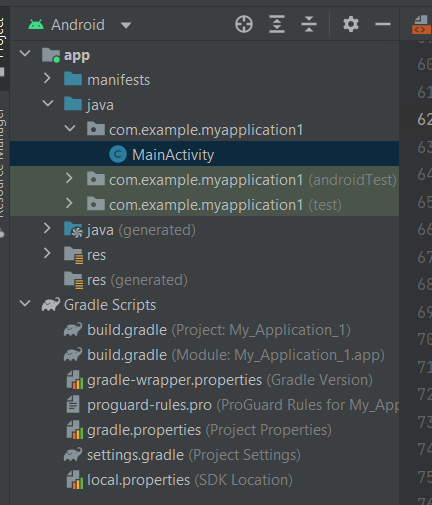


Рисунок 1 — Структура проекта

# **РЕАЛИЗАЦИЯ ПРОЕКТА**

## **Жизненный цикл Activity**

На Рисунках 1-5 предоставлены некоторые всплывающие сообщения, обозначающие жизненные циклы приложения:

1. При запуске и создании приложения появляется всплывающее уведомление “Hello onCreate!” (Рисунок 1).
2. Когда проект создан, но его окно ещё не видно пользователю, выходит всплывающее уведомление “Hello onStart!” (Рисунок 2).
3. Когда пользователь взаимодействует с созданным окном выходит всплывающее уведомление “Hello onResume!” (Рисунок 3).
4. Когда пользователь переходит на другие окна, то система выводит всплывающее уведомление “Hello onPause!” (Рисунок 4).
5. Когда пользователь выходит из приложения, или данная активность была удален, то система выводит всплывающее уведомление “Hello onStop!”
6. Когда пользователь выходит из активити и возвращается снова, то система выводит всплывающее уведомление “Hello onRestart!”.
7. Когда пользователь полностью выходит из активити, то выполняется метод “onDestroy()” и активити прекращает работу.

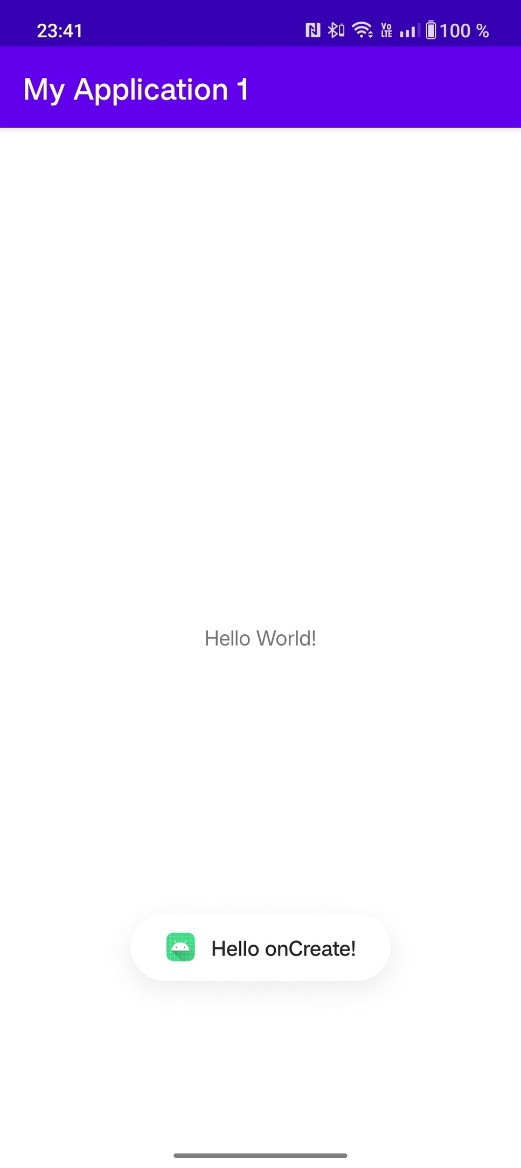


Рисунок 1 — Всплывающее сообщения “Hello onCreate!”

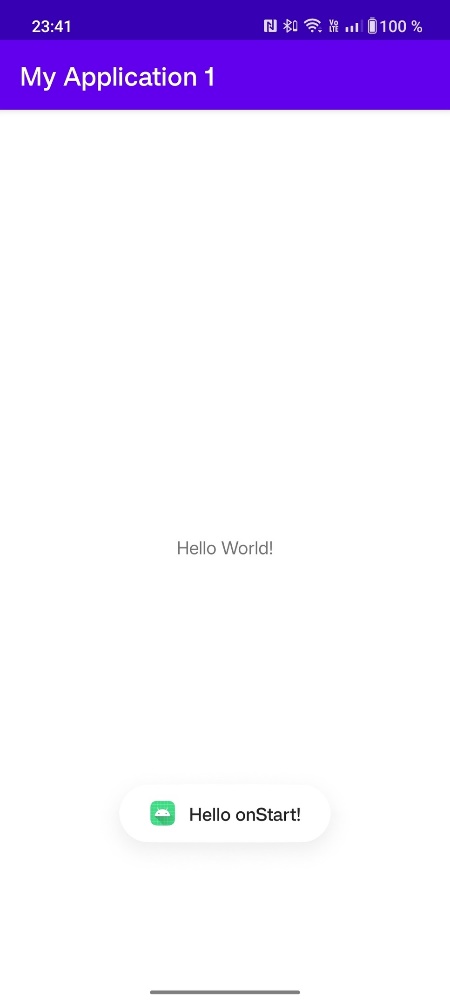


Рисунок 2 — Всплывающее сообщения “Hello onStart!”

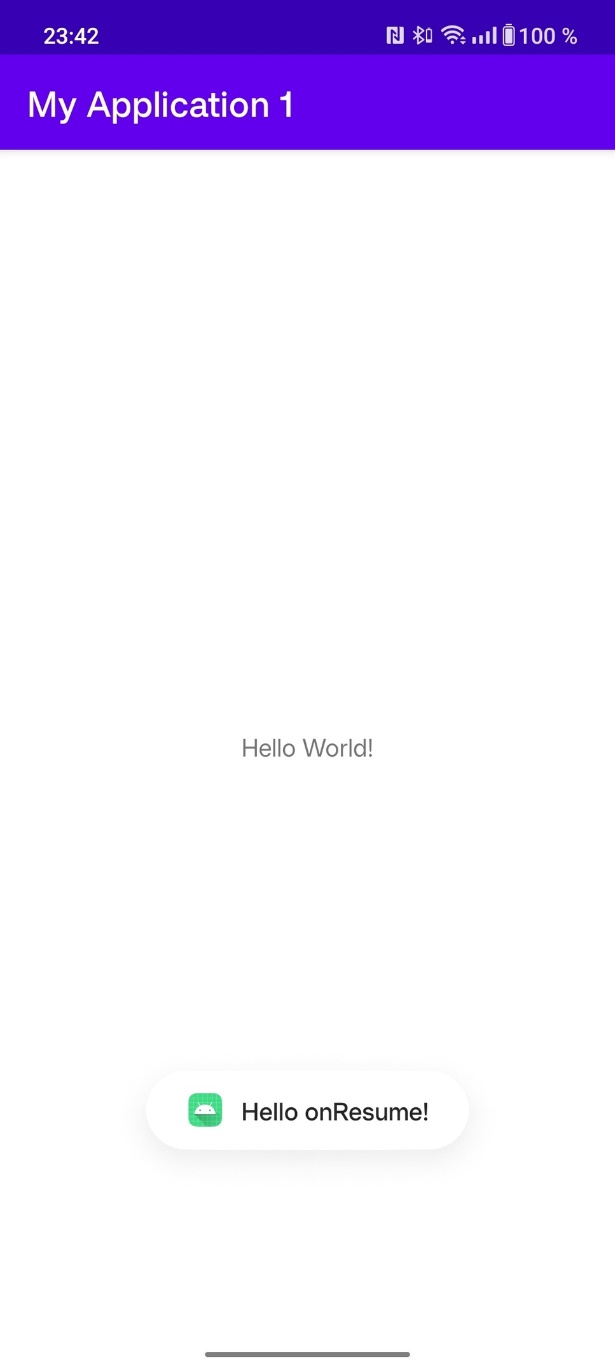


Рисунок 3 — Всплывающее сообщения “Hello onResume!”

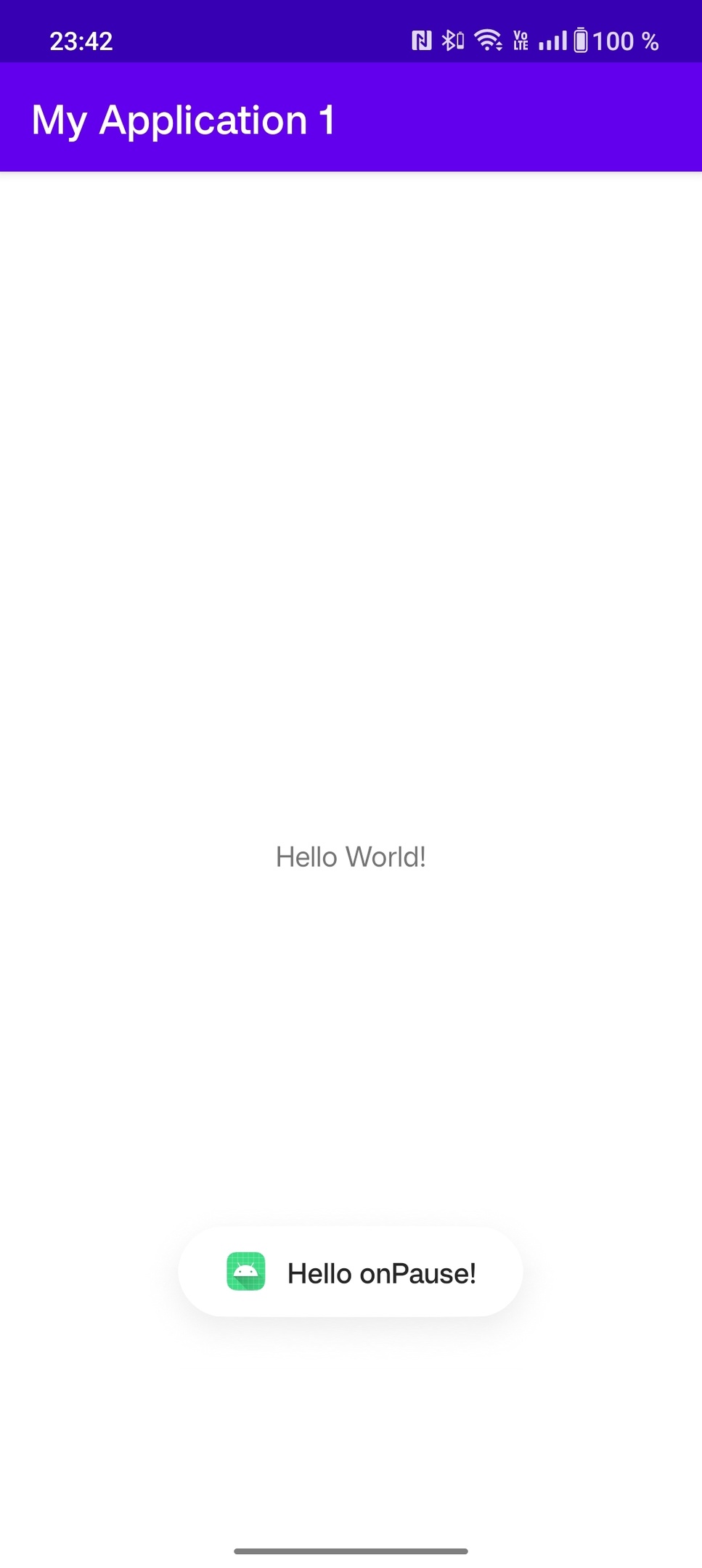


Рисунок 4 — Всплывающее сообщения “Hello onPause!”

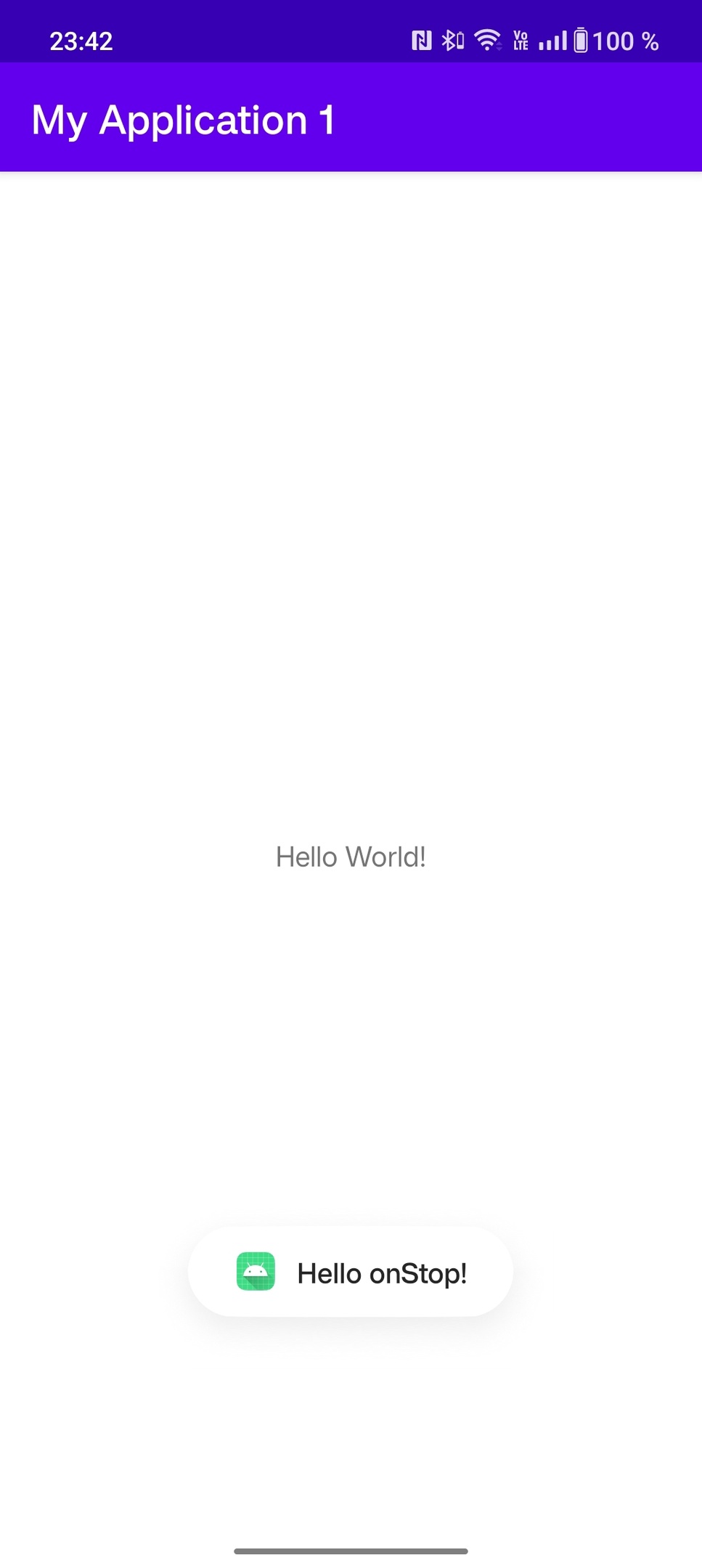


Рисунок 5 — Всплывающее сообщения “Hello onStop!”

## **Всплывающие уведомления (Toast)**

Благодаря всплывающим уведомлениям можно указать какое-либо совершаемое действие (как это было показано в прошлых примерах) или предупредить о чём-то пользователя. На рисунке 6 изображен код, в котором указан текст содержимого уведомления, время его длины и то, что оно должно быть показано.

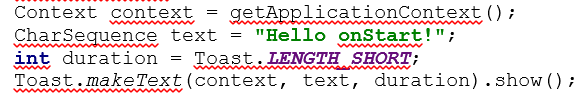


Рисунок 6 — Код для всплывающего уведомления

## **Логирование**

Окно Logcat помогает отлаживать ваше приложение, отображая журналы с вашего устройства в режиме реального времени — например, сообщения, которые вы добавили в свое приложение с помощью класса, сообщения от служб, работающих на Android, или системные сообщения, например, при сборке мусора. Когда приложение выдает исключение, Logcat показывает сообщение, за которым следует соответствующая трассировка стека, содержащая ссылки на строку кода.

Существует несколько специальных методов, которые легко запомнить по первым буквам, указывающие на категорию:

1. Log.e() - ошибки (error);
2. Log.w() - предупреждения (warning);
3. Log.i() - информация (info);
4. Log.d() - отладка (degub);
5. Log.v() — подробности

Данные категории можно найти снизу Android Studio возле Терминала (Рисунок 7).

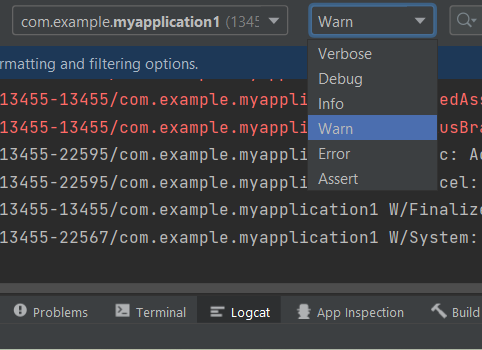


Рисунок 7 — Logcat

## **Код проекта**

Ниже представлен код реализуемого проекта, на листине 1.

Листинг 1 — Код проекта

|  |
| --- |
| **import** android.util.Log; **import** android.view.View; **import** android.widget.Toast;  **public class** MainActivity **extends** AppCompatActivity {  **private static final** String ***TAG*** = **"MyApp"**;  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  Context context = getApplicationContext();  CharSequence text = **"Hello onCreate!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onCreate"**);  Log.*w*(***TAG***, **"warning in onCreate"**);  Log.*i*(***TAG***, **"info in onCreate"**);  Log.*d*(***TAG***, **"debug in onCreate"**);  Log.*v*(***TAG***, **"verbose in onCreate"**);  }  **public void** b\_exitClick(View view) {  System.*exit*(0);  }  @Override  **protected void** onStart() {  **super**.onStart();  Context context = getApplicationContext();  CharSequence text = **"Hello onStart!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onStart"**);  Log.*w*(***TAG***, **"warning in onStart"**);  Log.*i*(***TAG***, **"info in onStart"**);  Log.*d*(***TAG***, **"debug in onStart"**);  Log.*v*(***TAG***, **"verbose in onStart"**);  }   @Override  **protected void** onRestart() {  **super**.onRestart();  Context context = getApplicationContext();  CharSequence text = **"Hello onRestart!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onRestart"**);  Log.*w*(***TAG***, **"warning in onRestart"**);  Log.*i*(***TAG***, **"info in onRestart"**);  Log.*d*(***TAG***, **"debug in onRestart"**);  Log.*v*(***TAG***, **"verbose in onRestart"**);  }   @Override  **protected void** onResume() {  **super**.onResume();  Context context = getApplicationContext();  CharSequence text = **"Hello onResume!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onResume"**);  Log.*w*(***TAG***, **"warning in onResume"**);  Log.*i*(***TAG***, **"info in onResume"**);  Log.*d*(***TAG***, **"debug in onResume"**);  Log.*v*(***TAG***, **"verbose in onResume"**);  }   @Override  **protected void** onPause() {  **super**.onPause();  Context context = getApplicationContext();  CharSequence text = **"Hello onPause!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onPause"**);  Log.*w*(***TAG***, **"warning in onPause"**);  Log.*i*(***TAG***, **"info in onPause"**);  Log.*d*(***TAG***, **"debug in onPause"**);  Log.*v*(***TAG***, **"verbose in onPause"**);  }   @Override  **protected void** onStop() {  **super**.onStop();  Context context = getApplicationContext();  CharSequence text = **"Hello onStop!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onStop"**);  Log.*w*(***TAG***, **"warning in onStop"**);  Log.*i*(***TAG***, **"info in onStop"**);  Log.*d*(***TAG***, **"debug in onStop"**);  Log.*v*(***TAG***, **"verbose in onStop"**);  }   @Override  **protected void** onDestroy() {  **super**.onDestroy();  Context context = getApplicationContext();  CharSequence text = **"Hello onDestroy!"**;  **int** duration = Toast.***LENGTH\_SHORT***;  Toast.*makeText*(context, text, duration).show();  Log.*e*(***TAG***, **"error in onDestroy"**);  Log.*w*(***TAG***, **"warning in onDestroy"**);  Log.*i*(***TAG***, **"info in onDestroy"**);  Log.*d*(***TAG***, **"debug in onDestroy"**);  Log.*v*(***TAG***, **"verbose in onDestroy"**);  } } |

# **ВЫВОД**

В результате выполнения практической работы создано первое Android приложение. Мы научились создавать и запускать приложение. Разобрались как выглядят и происходят жизненные циклы приложения. Получены навыки по созданию всплывающих сообщений. Изучили работу логов в Android Studio.