|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №2**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-06-21 | Гейдаров Э. З. |
| **Принял:** | доцент Синицин И.В. |

Москва 2023

**Содержание**

[Ход работы 3](#_Toc128684636)

[1. Поддержка различных языков 3](#_Toc128684637)

[2. Поддержка устройств с различными экранами 5](#_Toc128684638)

[3. Поддержка различных версий Android 7](#_Toc128684639)

[4. Использование внутренних тем и стилей 8](#_Toc128684640)

[5. Жизненный цикл явлений 9](#_Toc128684641)

[Тестирование приложения 13](#_Toc128684642)

[Вывод 15](#_Toc128684643)

# Ход работы

## Поддержка различных языков

В новом проекте создадим директорию и в ней файл со строковыми ресурсами для английского языка:

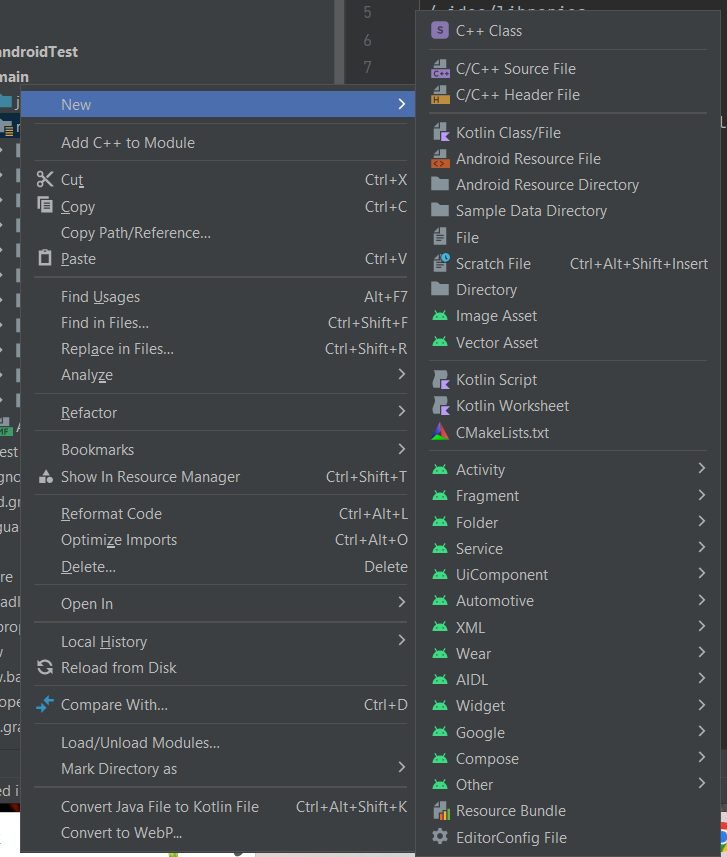


Рисунок 1. Создание директории английского языка

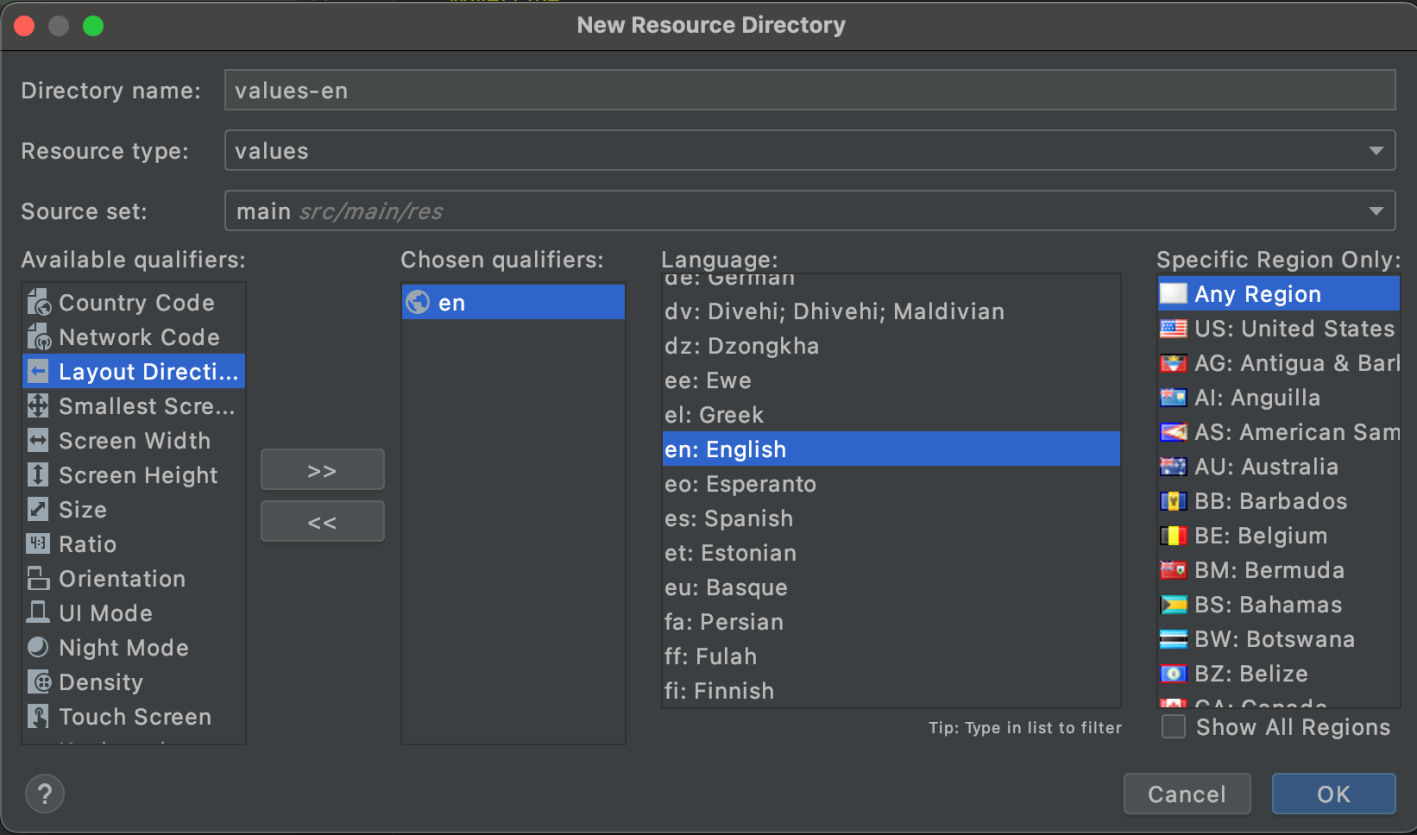


Рисунок 2. Создание файла со строковыми ресурсами для английского языка

****

Рисунок 3. Строковые ресурсы



Рисунок 5. Использование строковых ресурсов

* 1. **Поддержка устройств с различными экранами**

В качестве примера использования различной разметки возьмём разметки для landscape ориентации и больших экранов.

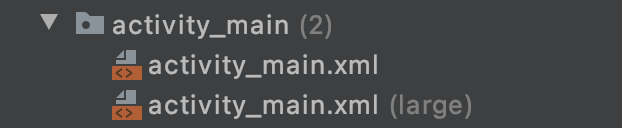


Рисунок 6. Разметки для landscape ориентации и больших экранов

Далее добавим в проект изображение с различными разрешениями, для этого выберем создание Image Asset. В нём выберем изображение и добавим, как показано ниже:

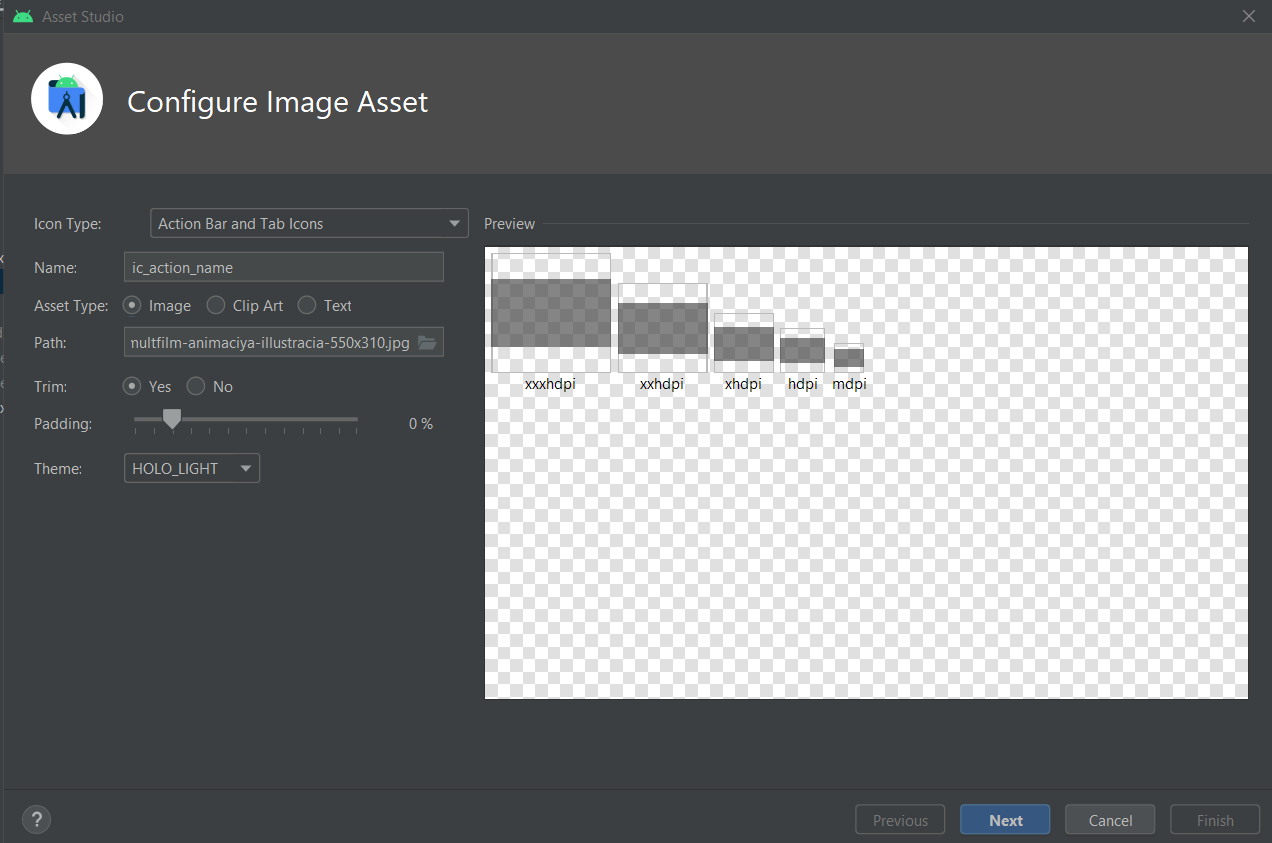


Рисунок 7. Добавление изображения

После в файле activity\_main.xml добавим виджет ImageView, в котором укажем наше изображение.

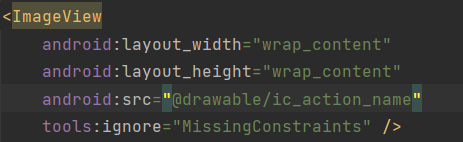


Рисунок 8. Добавление изображения в разметке

## Поддержка различных версий Android

Установим минимальный и целевой уровень API, на котором приложение будет использоваться. Для этого изменим файл AndroidManifest.xml:

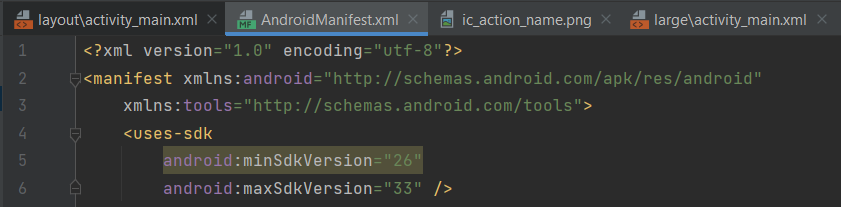


Рисунок 9. Установка минимального и целевого уровня API

Так же проведём проверку уровня API в коде:



Рисунок 10. Проверка уровня API

## Использование внутренних тем и стилей

Изначально приложение использует стиль созданный автоматически, который можно редактировать. Его применение прописано в манифесте.

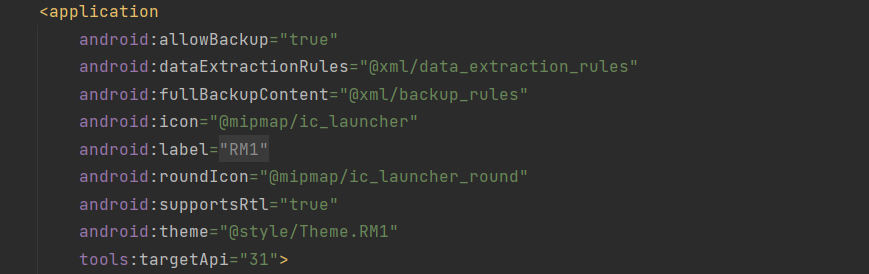


Рисунок 11. Применение стиля

Попробуем поменять стиль на какой-нибудь из стандартных стилей, как показано ниже:



Рисунок 12. Применение встроенного стиля

## Жизненный цикл явлений

Убедимся, что наше явление является главным при запуске, для этого проверим файл AndroidManifest.xml и найдём строки

<action android:name="android.intent.action.MAIN" /> <categoryandroid:name="android.intent.category.LAUNCHER" />

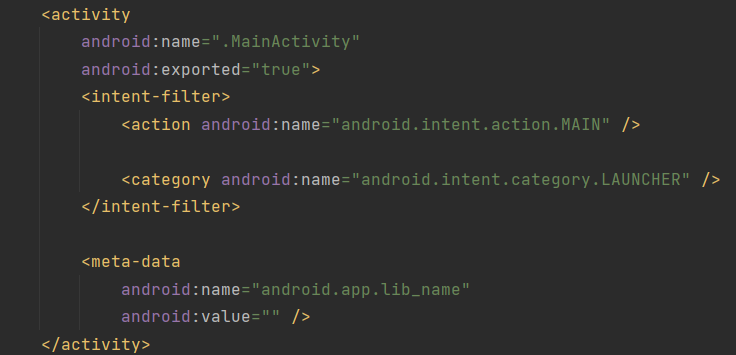


Рисунок 13. Проверка явления

В коде явления добавим проверку метода onCreate для этого изменим значение текстового поля:

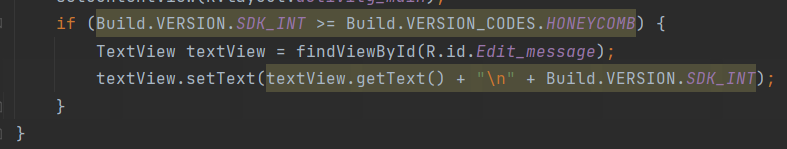


Рисунок 14. Изменение текстового поля

Так же добавим обработку метода onDestroy, который выполняется при уничтожении приложения:

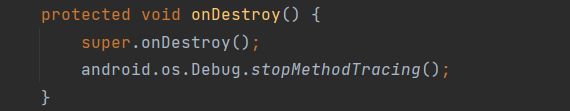


Рисунок 15. Метод onDestroy

Далее добавим обработку метода onPause, который вызывается, когда пользователь только закрыл приложение и оно ещё может быть частично видимо:

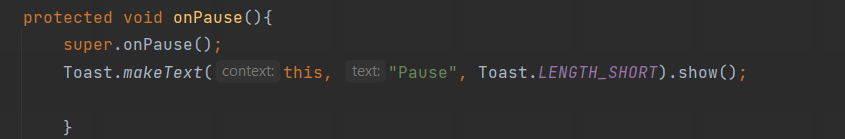


Рисунок 16. Метод onPause

Теперь добавим обработку метода onResume, который вызывается при возвращении фокуса явлению, как показано ниже:

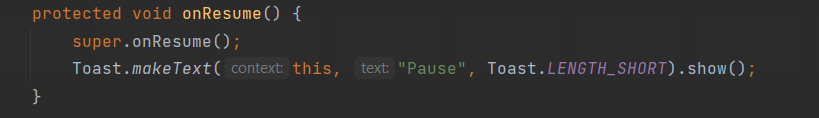


Рисунок 17. Метод onResume

Когда вызывается метод onStop, оно уже невидимо пользователю, и вы должны освободить все неиспользуемые ресурсы.



Рисунок 18. Метод onStop

Метод onStart вызывается всегда приложение становится видимым, а метод onRestart вызывается, когда приложение возвращается из остановленного состояния. Добавим реализацию методов, как показано ниже:

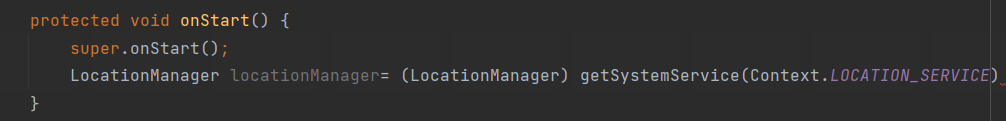


Рисунок 19. Методы onStart и onRestart

Реализуем так же метод onSaveInstanceState, который помогает сохранить состояние явление при его пересоздании, как показано ниже:

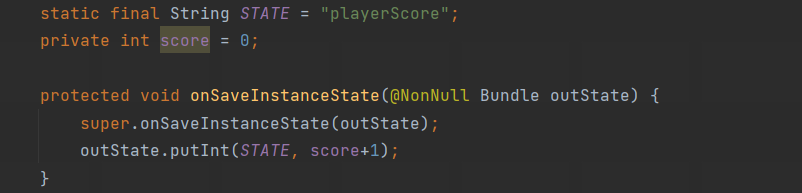


Рисунок 20. Метод onSaveInstanceState

Теперь добавим в методе onCreate обработку восстановления состояния:

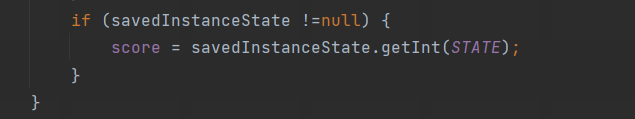


Рисунок 21. Восстановление состояния

Так же для восстановления состояния можно использовать метод onRestoreInstanceState, который вызывается после метода onStart, если есть данные о предыдущем состоянии явления и нет необходимости проверять их наличие внутри метода, как показано ниже:

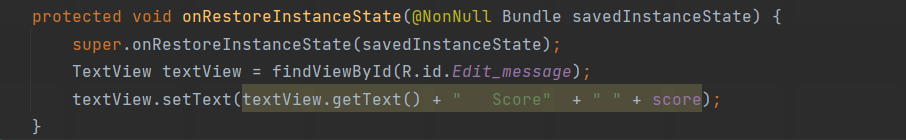


Рисунок 22. Метод onRestoreInstanceState

# Вывод

В ходе работы были получены навыки разработки Android приложения, поддержка различных языков, устройств с различными экранами, различных версий Android, использование внутренних тем и стилей, жизненный цикл явлений. Был составлен отчёт со скриншотами полученных результатов.