|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| **Институт кибербезопасности и цифровых технологий** |

Кафедра КБ-14 «Цифровые технологии обработки данных»

**ОТЧЕТ**

**Практическая работа 2**

**по дисциплине «Разработка мобильных компонент анализа безопасности информационно-аналитических систем»**

Выполнил:

Студент группы БСБО-07-22

Сладкина Анастасия Андреевна

**Москва, 2025 г**

Изучим класс MainActivity из Android-приложения, который демонстрирует работу с жизненным циклом активности (Activity Lifecycle) и сохранением состояния при повороте экрана или других изменениях конфигурации. Для начала вызывается родительский onCreate(), где устанавливается макет activity\_main.xml (setContentView()). Далее находится EditText по ID (findViewById). Если savedInstanceState != null (например, после поворота экрана) — восстанавливается текст из Bundle (используя ключ KEY\_TEXT). Текст устанавливается в EditText. При повороте экрана Android уничтожает и пересоздаёт активность. Для того чтобы текст в EditText не пропадал, он сохраняется в onSaveInstanceState() и восстанавливается здесь (см. рис. 1).

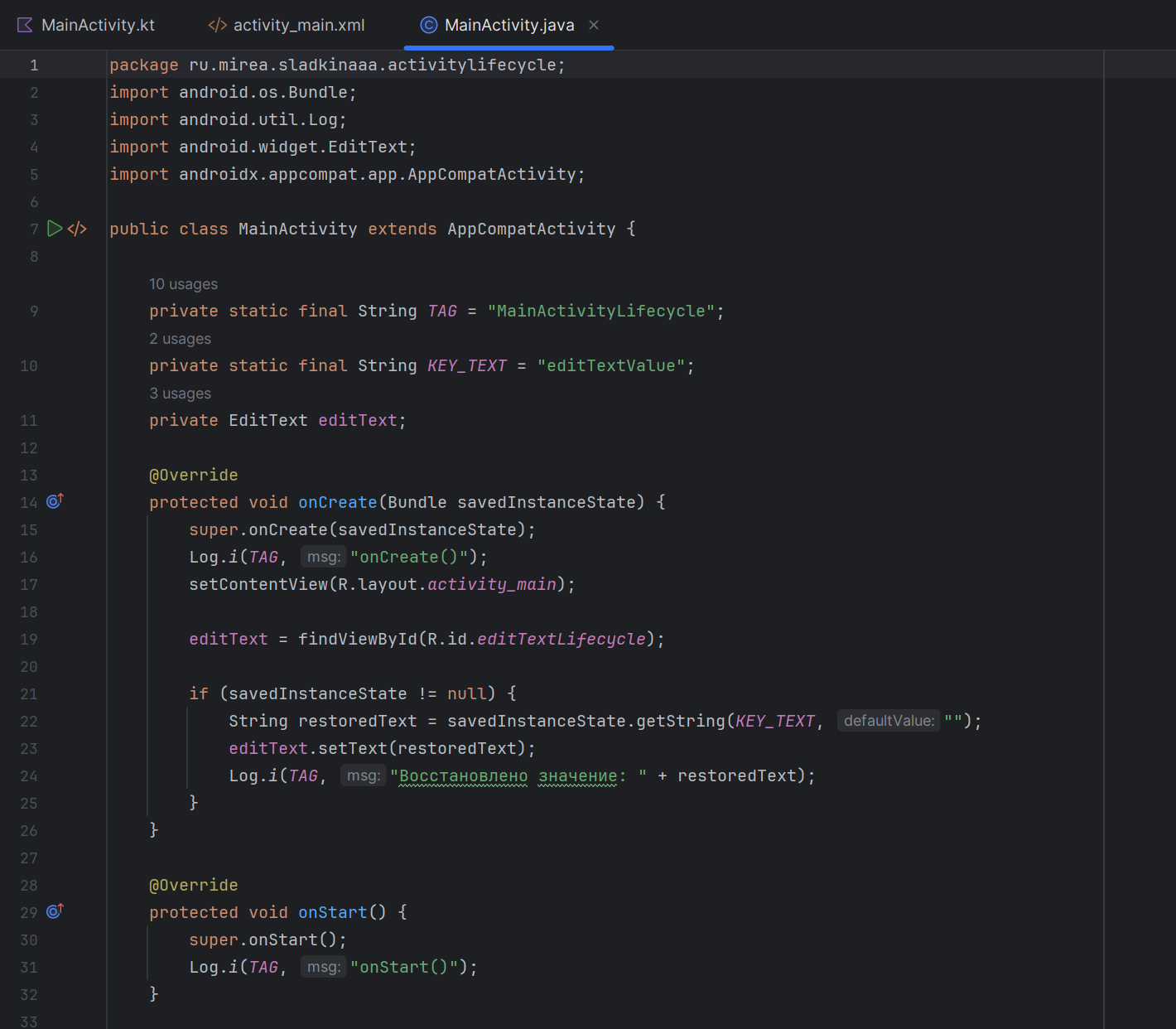


Рисунок 1. Методы onCreate() и onStart()

Метод onRestoreInstanceState() отвечает за восстановление состояния, вызывается после onStart(), если активность была пересоздана (например, после поворота экрана), и есть сохранённое состояние (savedInstanceState). Метод восстанавливает данные, сохранённые в onSaveInstanceState(). onRestart() отвечает за перезапуск активности, когда активность возвращается на экран после остановки (onStop()), но ещё не стала видимой. Метод onResume() следит за активностью в фокусе, когда активность становится интерактивной (пользователь может с ней взаимодействовать).

Выше упомянутый метод onSaveInstanceState() сохраняет состояния перед уничтожением активности. Сохраняет текст из EditText в Bundle под ключом \_KEY\_TEXT\_. Позже это значение можно восстановить в onCreate() или onRestoreInstanceState().

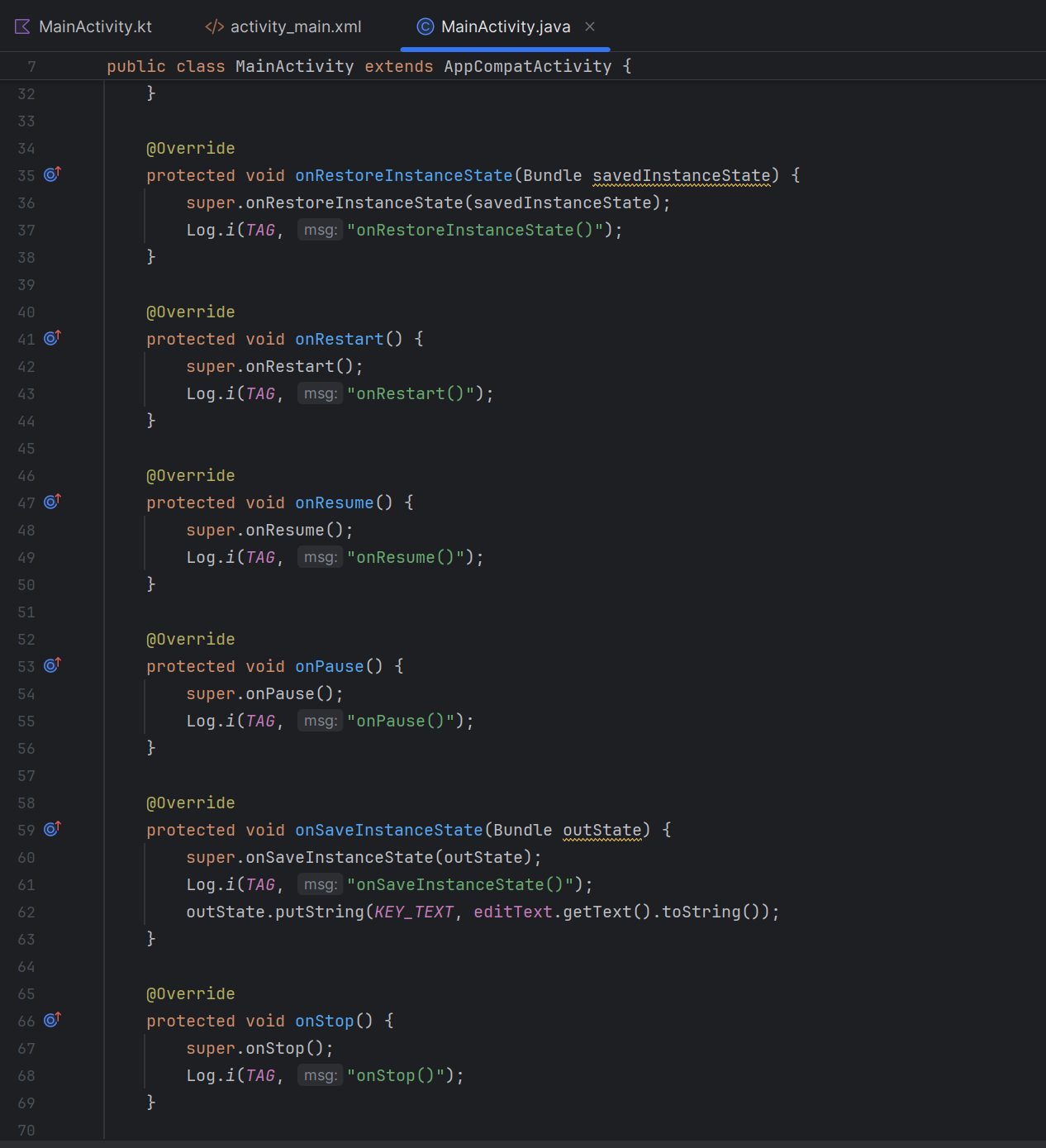


Рисунок 2. Оставшиеся методы класса MainActivity

Далее добавляем в разметку «activity\_main.xml» поле текстового ввода «EditText» и реализуем отображение состояния «activity» используя класс «Log» (см. рис. 3).

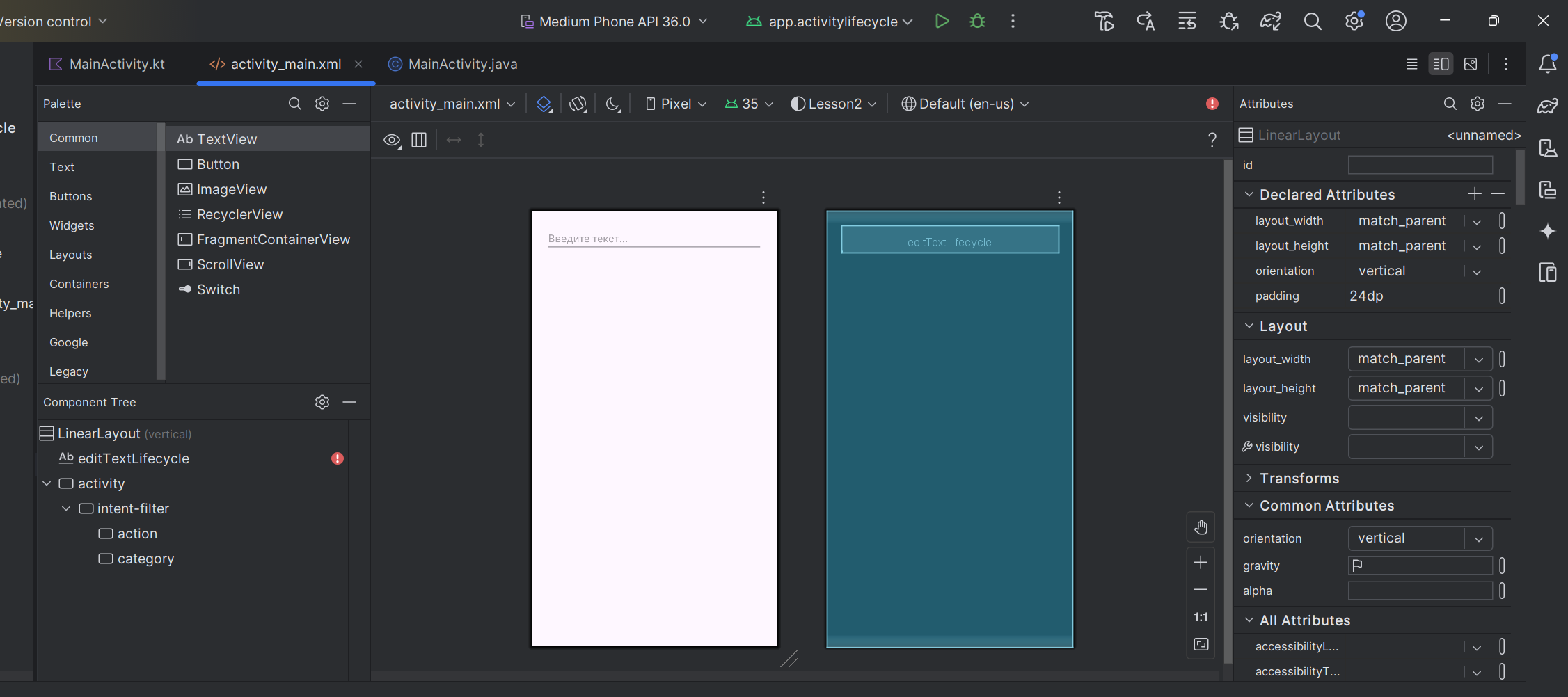


Рисунок 3. Добавление поля текстового ввода «EditText»

1. *Будет ли вызван метод onCreate после нажатия на кнопку «Home» и возврата в приложение?* Нет, метод onCreate() не будет вызван после нажатия кнопки «Home» и возврата в приложение. При нажатии на кнопку «Home» приложение переходит в фоновый режим, но активность не уничтожается — она просто переходит в состояние onStop(). При возвращении в приложение вызываются методы onRestart(), onStart() и onResume(), но не onCreate(), потому что Activity не пересоздаётся, а просто возобновляется.

2. *Изменится ли значение поля EditText после нажатия на кнопку «Home» и возврата в приложение?* Значение не изменится, так как активность не пересоздаётся, а только переходит в фон и возвращается, состояние элементов UI, таких как EditText, сохраняется автоматически в памяти процесса. Поэтому при возврате отображается то же значение, которое вводилось до нажатия «Home».

3. *Изменится ли значение поля EditText после нажатия на кнопку «Back» и возврата в приложение?* Да, изменится — поле будет пустым (или будет значение по умолчанию). Кнопка «Back» закрывает текущую активность (вызывает finish()), что приводит к её уничтожению. При следующем запуске приложения (если нажать на иконку снова) активность создаётся заново — вызывается onCreate(). При этом, если приложение было полностью закрыто, поле EditText загрузится из ресурсов (разметки), и если не было сохранено состояние (например, через onSaveInstanceState()), то текст будет пустым.

Для следующего задания в разметке activity\_main.xml была добавлена кнопка и реализован обработчик нажатия на кнопку — при нажатии должен вызываться метод nClickNewActivity(View) в MainActivity (см. рис. 4)

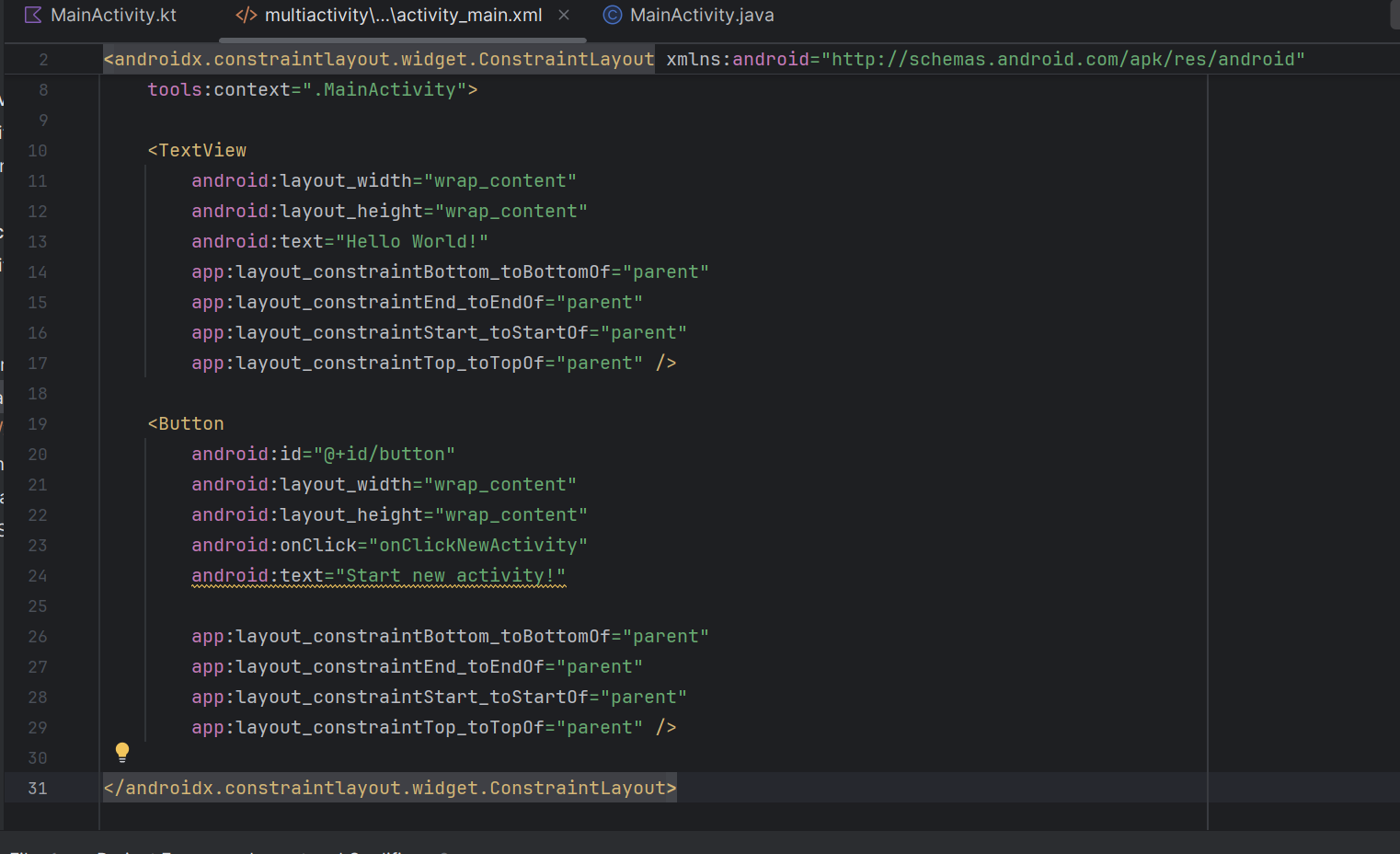


Рисунок 4. Добавление и настройка новой кнопки

Создаем новую активность через менеджер создания компонента (см. рис. 5). 

Рисунок 5. Менеджер создания Activity

В файл разметки «activity\_second.xml», расположенный в ресурсах, был добавлен «TextView». Далее был инициализирован вызов данного «activity» из «MainActivity» (см. рис. 6).

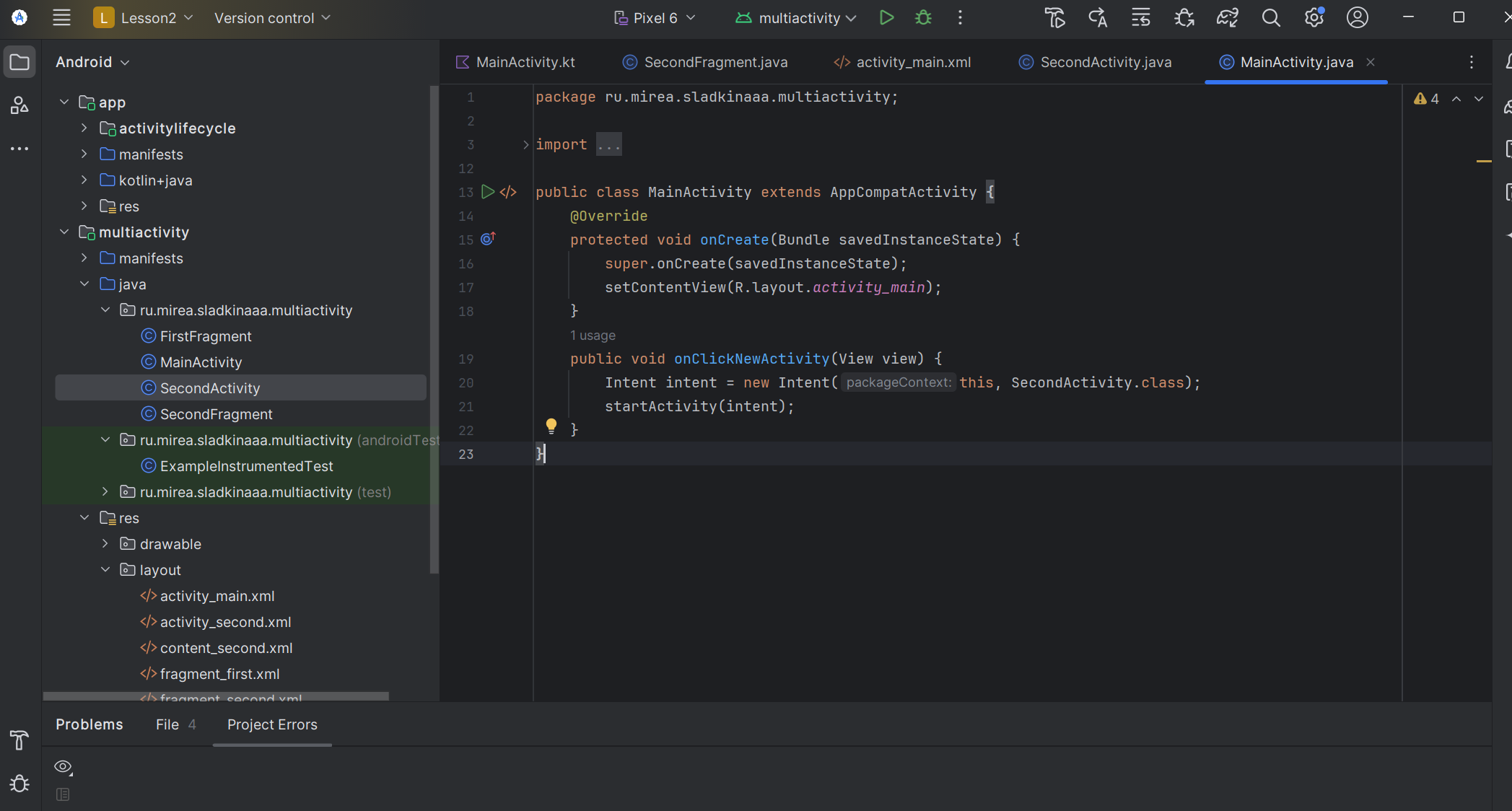


Рисунок 6. Менеджер создания Activity



Рисунок 7. Визуальный экран созданной активности

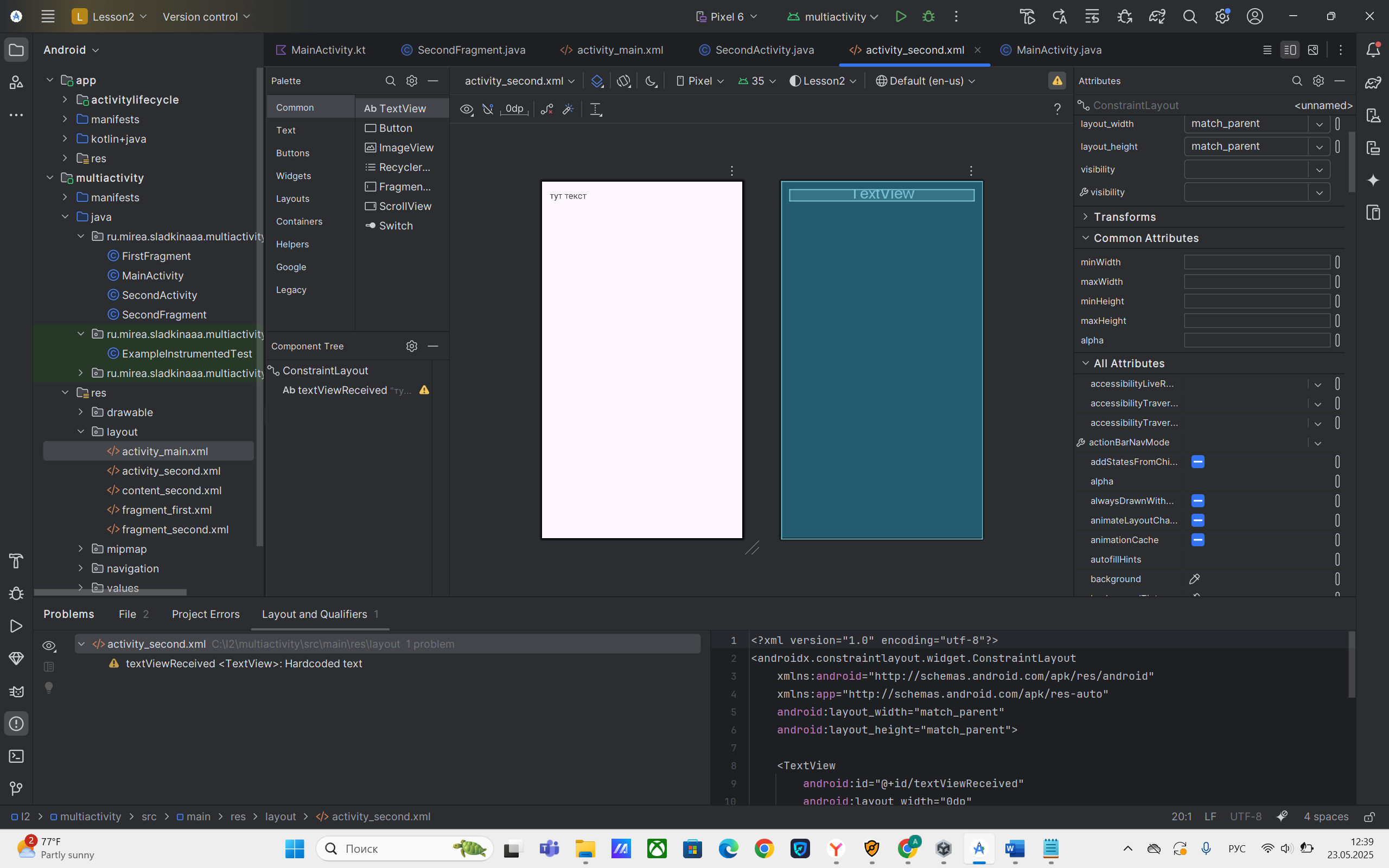


Рисунок 8. Результат добавления текста

Рассмотрим код активности SecondActivity, которая принимает текстовые данные из другой активности и отображает их (см. рис. 9). Метод onCreate() устанавливает макет activity\_second.xml (содержит TextView с ID textViewReceived). Он находит TextView через findViewById() и извлекает текст из Intent: getIntent() — получает Intent, который запустил активность, getStringExtra("key") — извлекает строку по ключу "key". Далее текст отображается в TextView, если он не null.

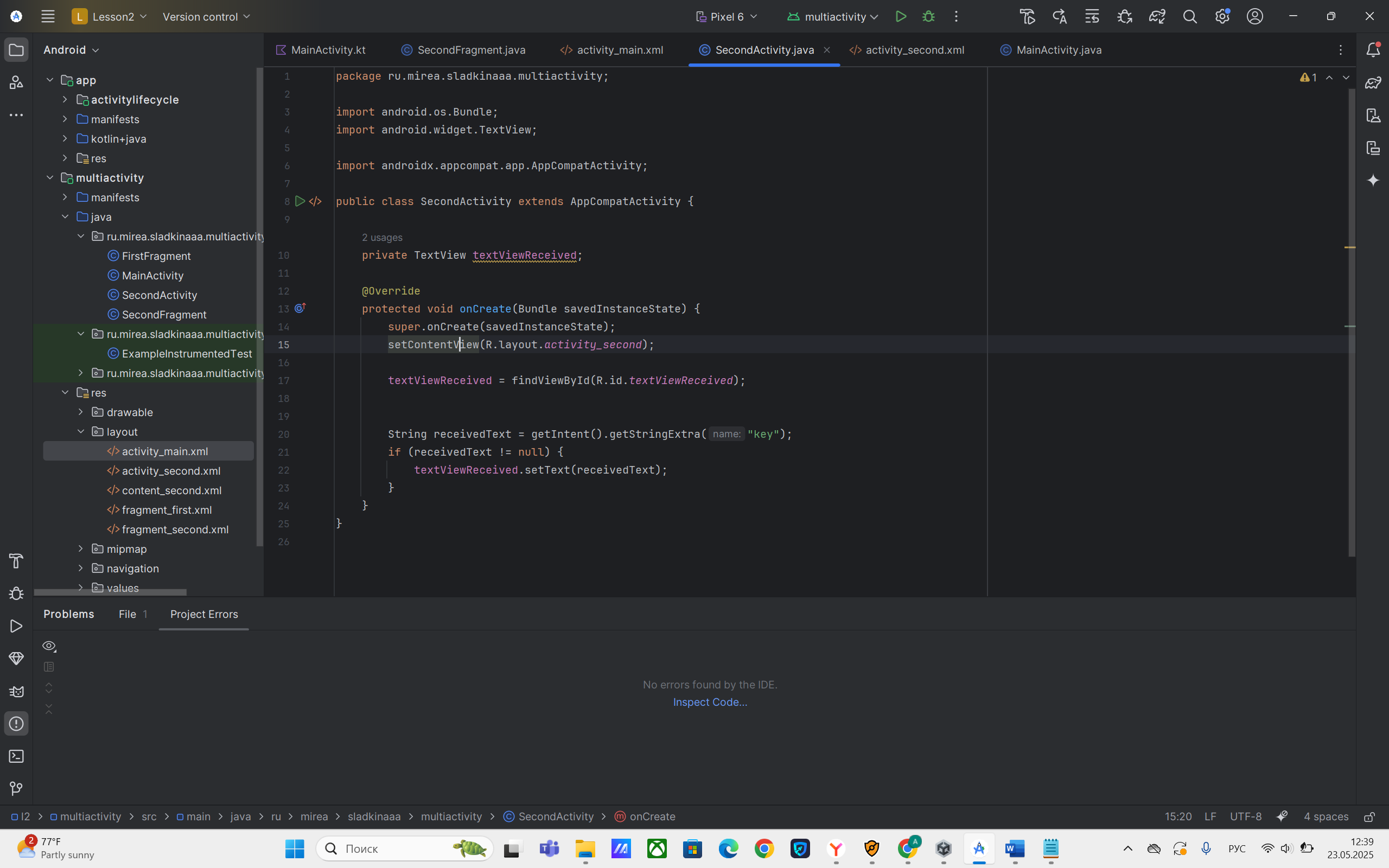


Рисунок 9. Активность SecondActivity

Теперь рассмотрим код главной активности MainActivity, которая отправляет текст из EditText в другую активность SecondActivity (см. рис. 10). Метод onCreate() загружает макет activity\_main.xml и находит поле EditText по ID editTextInput из макета. onClickNewActivity() из EditText извлекает введённый текст, создаёт Intent для перехода в SecondActivity. Текст упаковывается в Intent с ключом "key".

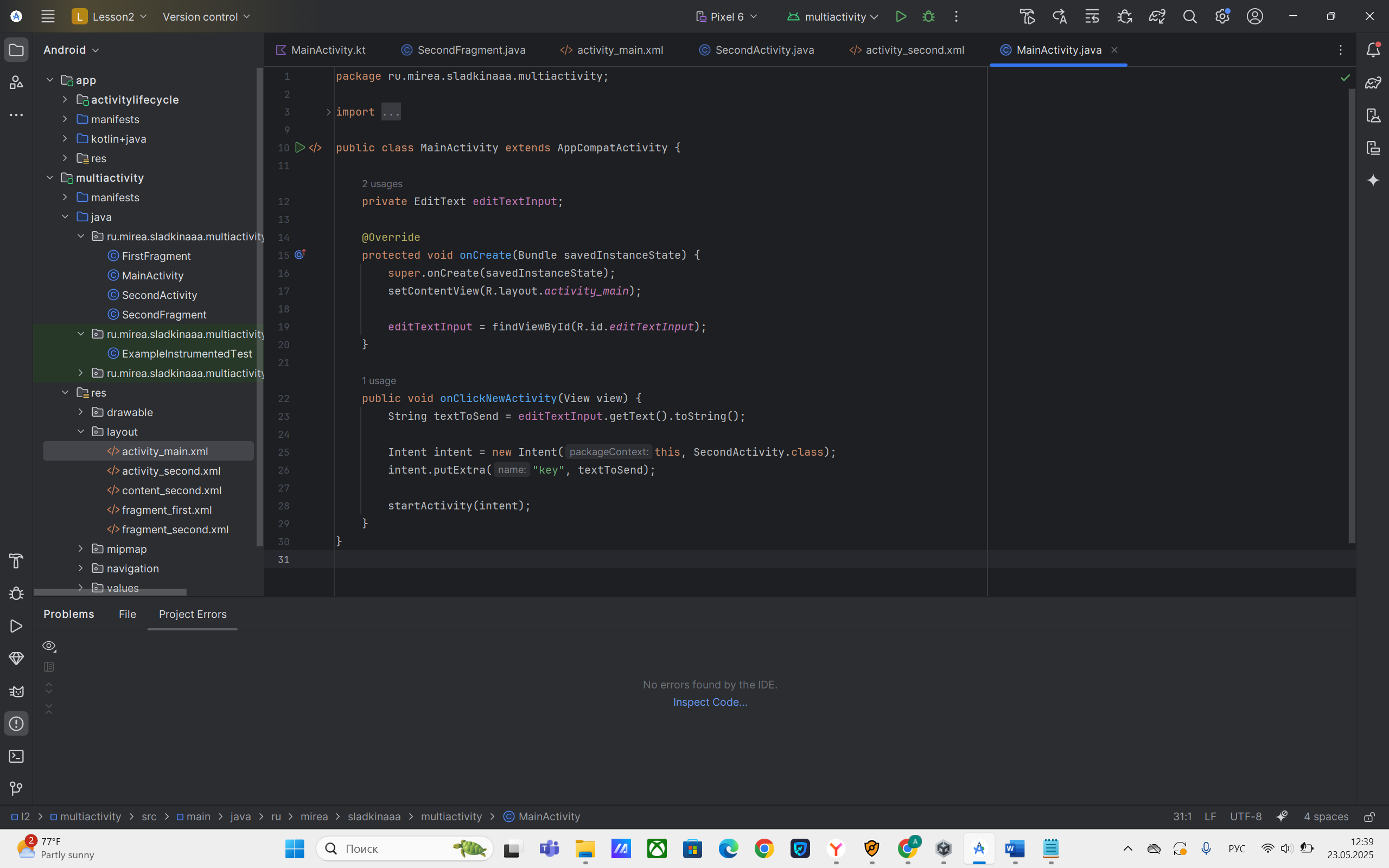


Рисунок 10. Активность MainActivity

Далее требуется переопределить основные методы жизненного цикла у обеих «Activity». Изучить жизненный цикл двух активностей. Рассмотрим жизненный цикл SecondActivity (см. рис. 11):

onCreate() – создание активности (инициализация UI, данных).

onStart() – активность становится видимой.

onResume() – активность получает фокус (пользователь может взаимодействовать).

onPause() – активность теряет фокус (например, при открытии диалога).

onStop() – активность больше не видна (её перекрыла другая активность).

onRestart() – активность возвращается из stopped-состояния.

onDestroy() – завершение работы активности.

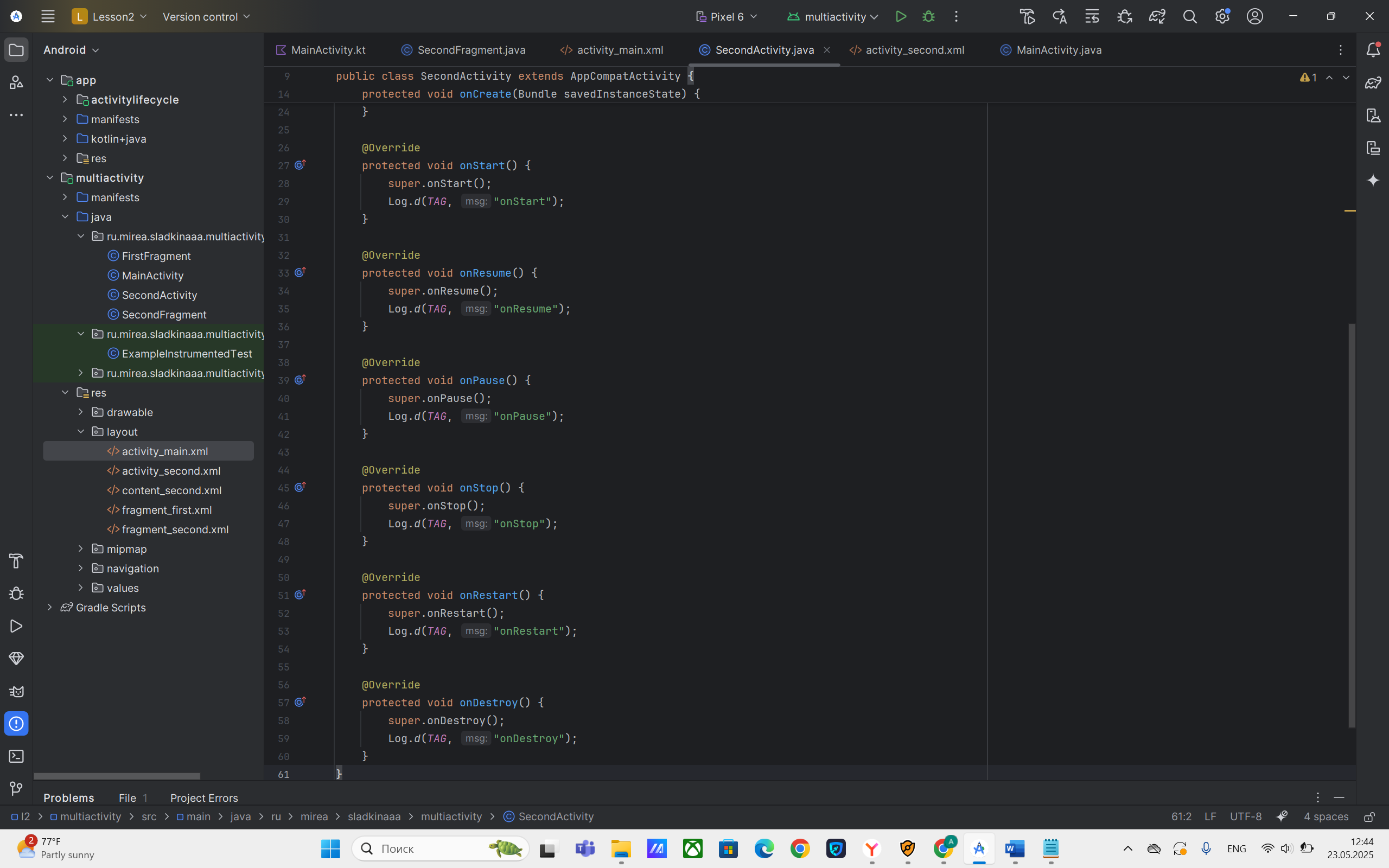


Рисунок 11. Жизненный цикл SecondActivity

Жизненный цикл MainActivity (см. рис. 12):

onCreate() – создание активности (Обязательно должен содержать setContentView()).

onStart() – активность становится видимой (Активность видна, но не взаимодействует с пользователем).

onResume() – активность получает фокус (Пользователь может взаимодействовать с активностью).

onPause() – активность теряет фокус (Сохраняет данные, останавливает ресурсоёмкие операции).

onStop() – активность больше не видна (Освобождает системные ресурсы).

onRestart() – перезапуск активности(Вызывается перед onStart() при возврате к активности).

onDestroy() – завершение работы активности (Финальная очистка ресурсов).

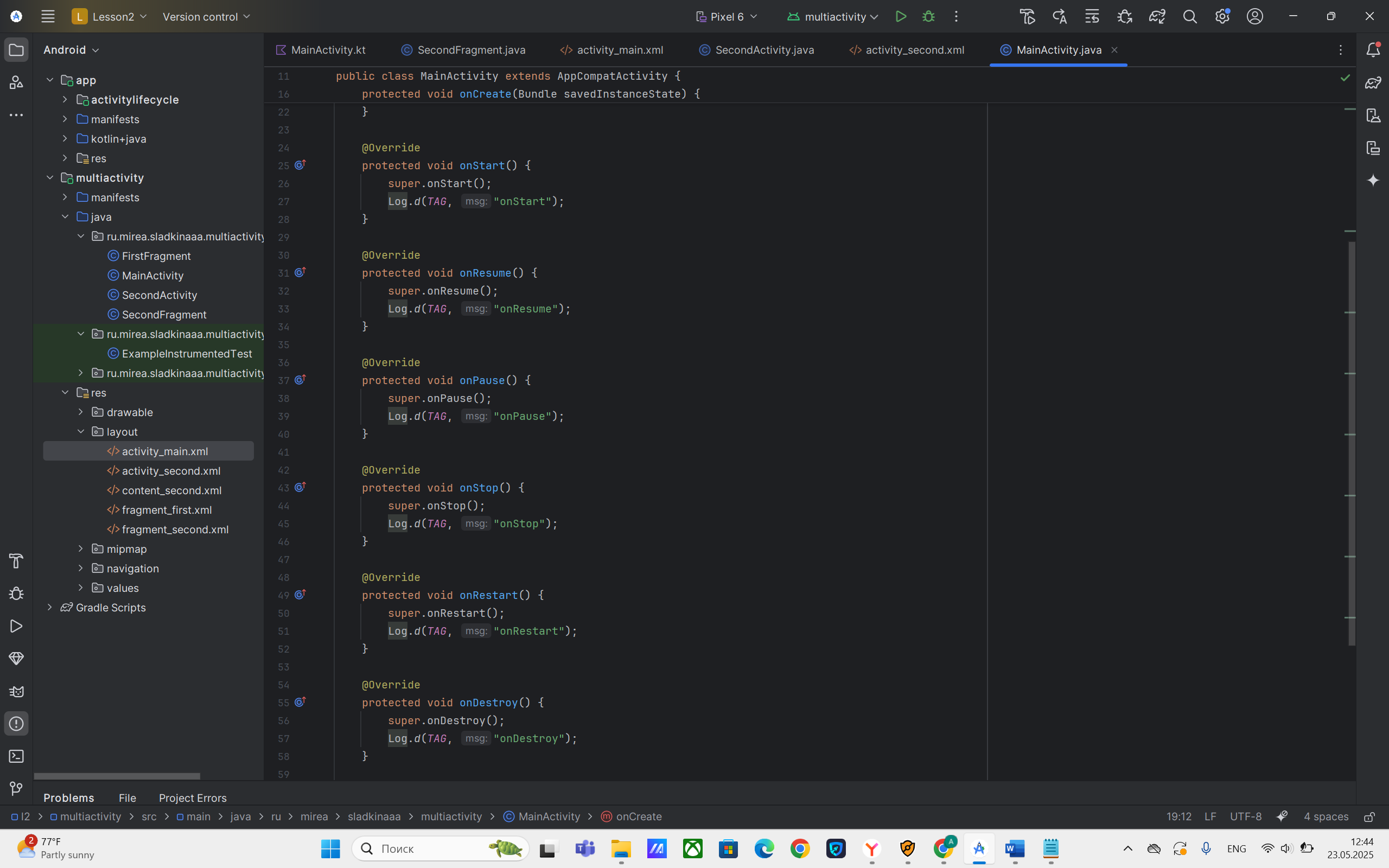


Рисунок 12. Жизненный цикл MainActivity

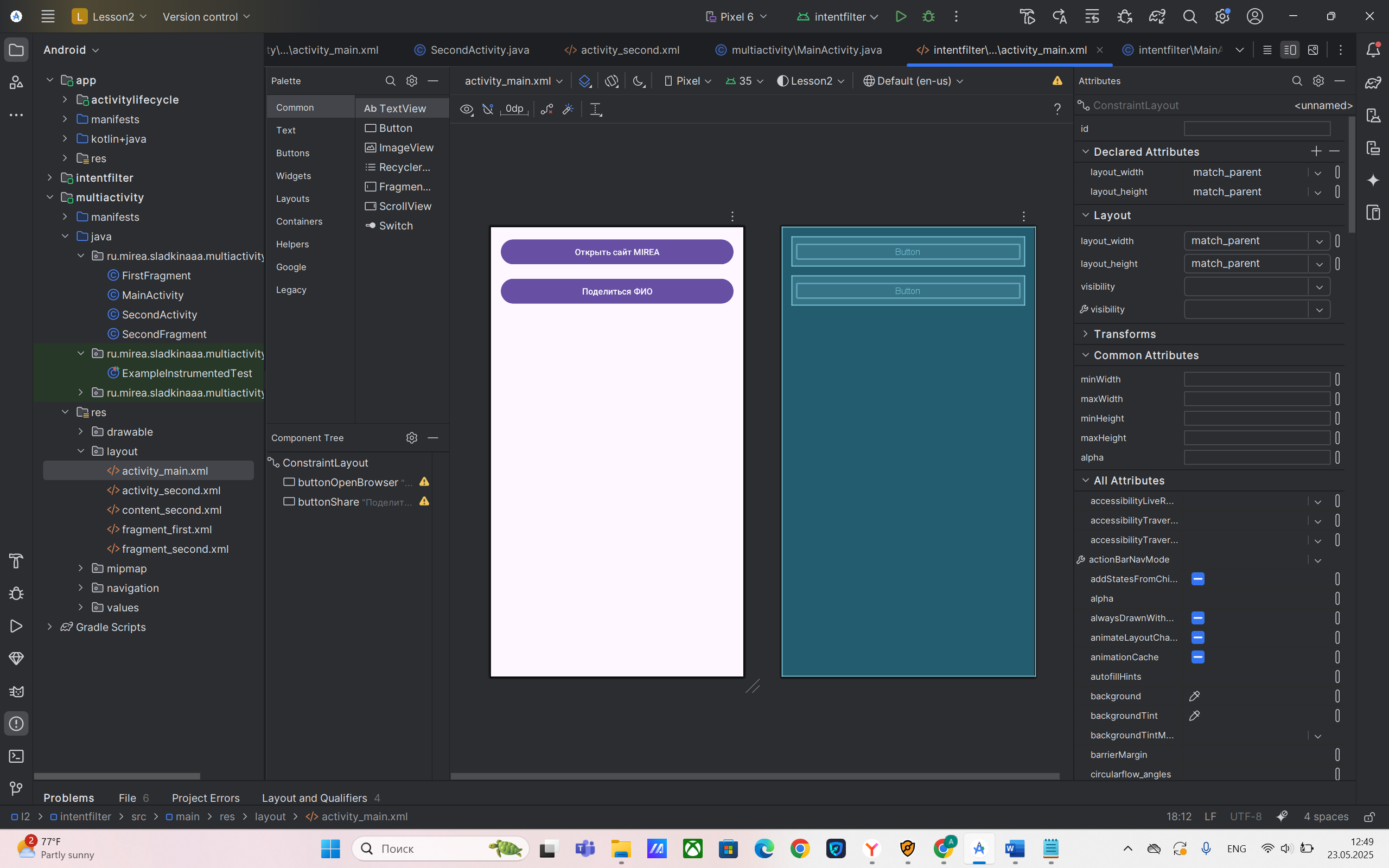
Далее были добавлены в разметку «activity\_main.xml» одна кнопка и обработчик нажатия для вызова веб-браузера (см. рис. 13). 

Рисунок 13. Изменение разметки «activity\_main.xml»

Добавляем дополнительную кнопку для передачи ФИО студента и университета в другое приложение (см. рис. 14). Имеющиеся кнопки: buttonOpenBrowser - открывает сайт в браузере, buttonShare - передаёт данные в другие приложения. Кнопки связываются с элементами из макета activity\_main.xml.

Передача данных реализуется через создание Intent.ACTION\_SEND, который указывает на действие "отправить данные", где setType("text/plain") определяет тип данных (текст). Рассмотрим добавление данных: EXTRA\_SUBJECT - тема сообщения ("HIREA"), EXTRA\_TEXT - основной текст с ФИО и университетом. Далее createChooser() показывает системный диалог с выбором приложений (мессенджеры, почта и т.д.).

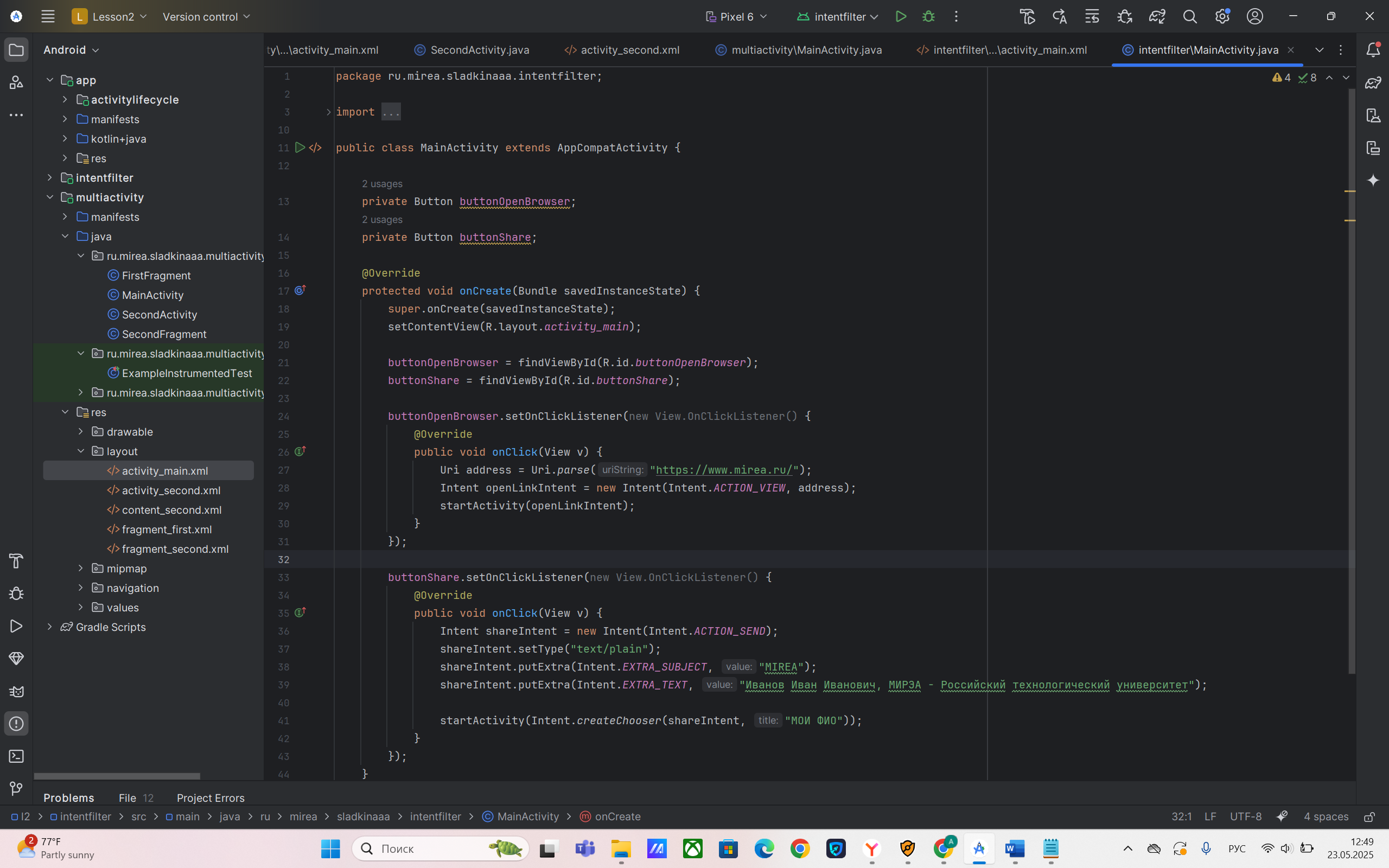


Рисунок 14. Реализация дополнительной кнопки передачи данных

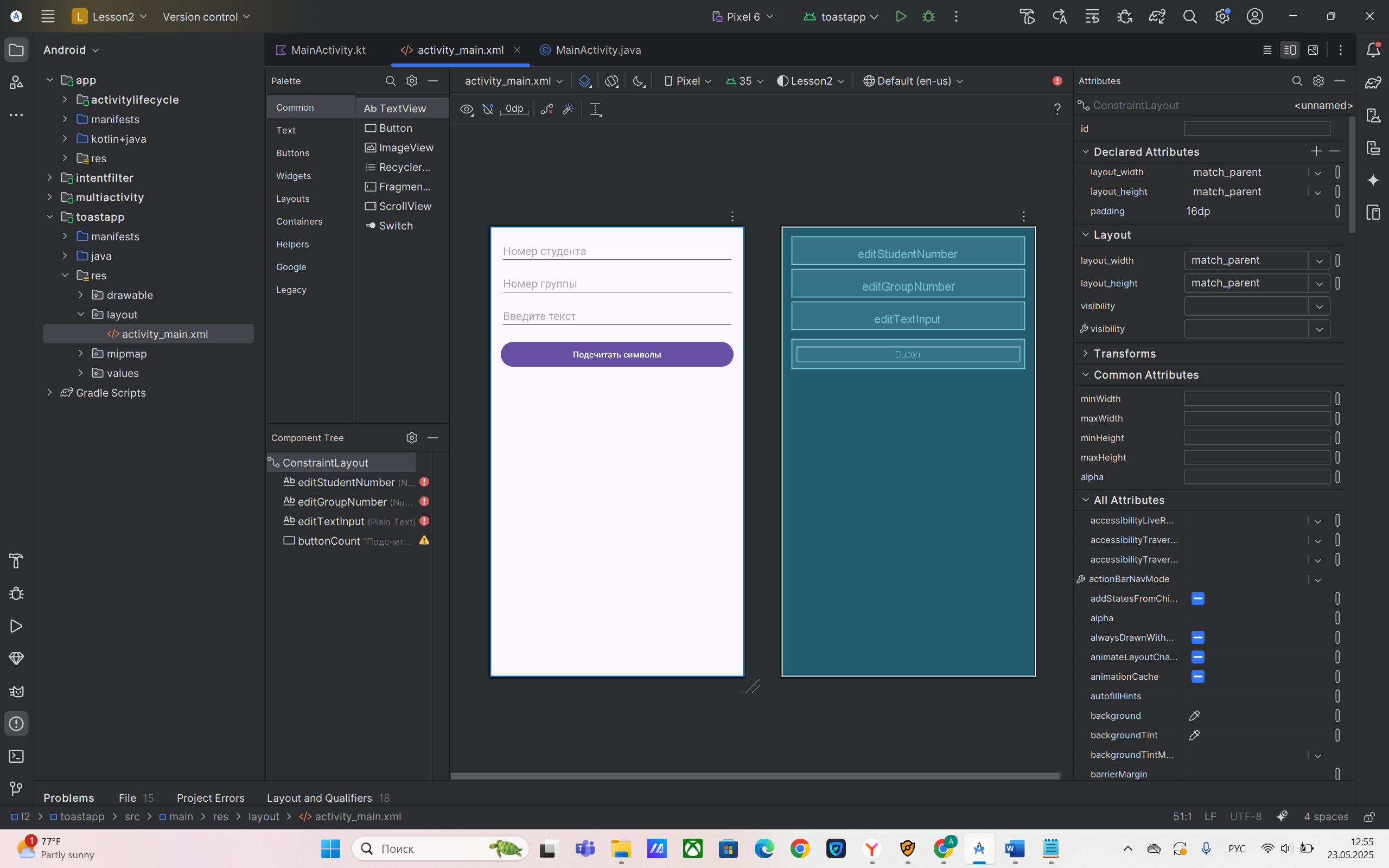
Перед реализацией всплывающих окон были добавлены поле ввода и кнопка для подсчета количества символов в поле ввода и отображения сообщения (см. рис. 15).

Рисунок 15. Добавление полей ввода и текста

Экземпляр «NotificationManager» создается при помощи вызова метода «getSystemService», а затем, когда требуется показать уведомление пользователю, вызывается метод «notify», либо метод from() (см. рис. 16). Пользователь вводит данные в три поля номер студента (editStudentNumber), номер группы (editGroupNumber), произвольный текст (editTextInput). При нажатии кнопки "buttonCount" проверяется заполнение обязательных полей (номер студента и группы). Если поля пустые - показывается Toast с ошибкой, если поля заполнены, то считается количество символов в editTextInput, далее формируется сообщение с данными студента и результатом подсчёта и показывается информационный Toast.

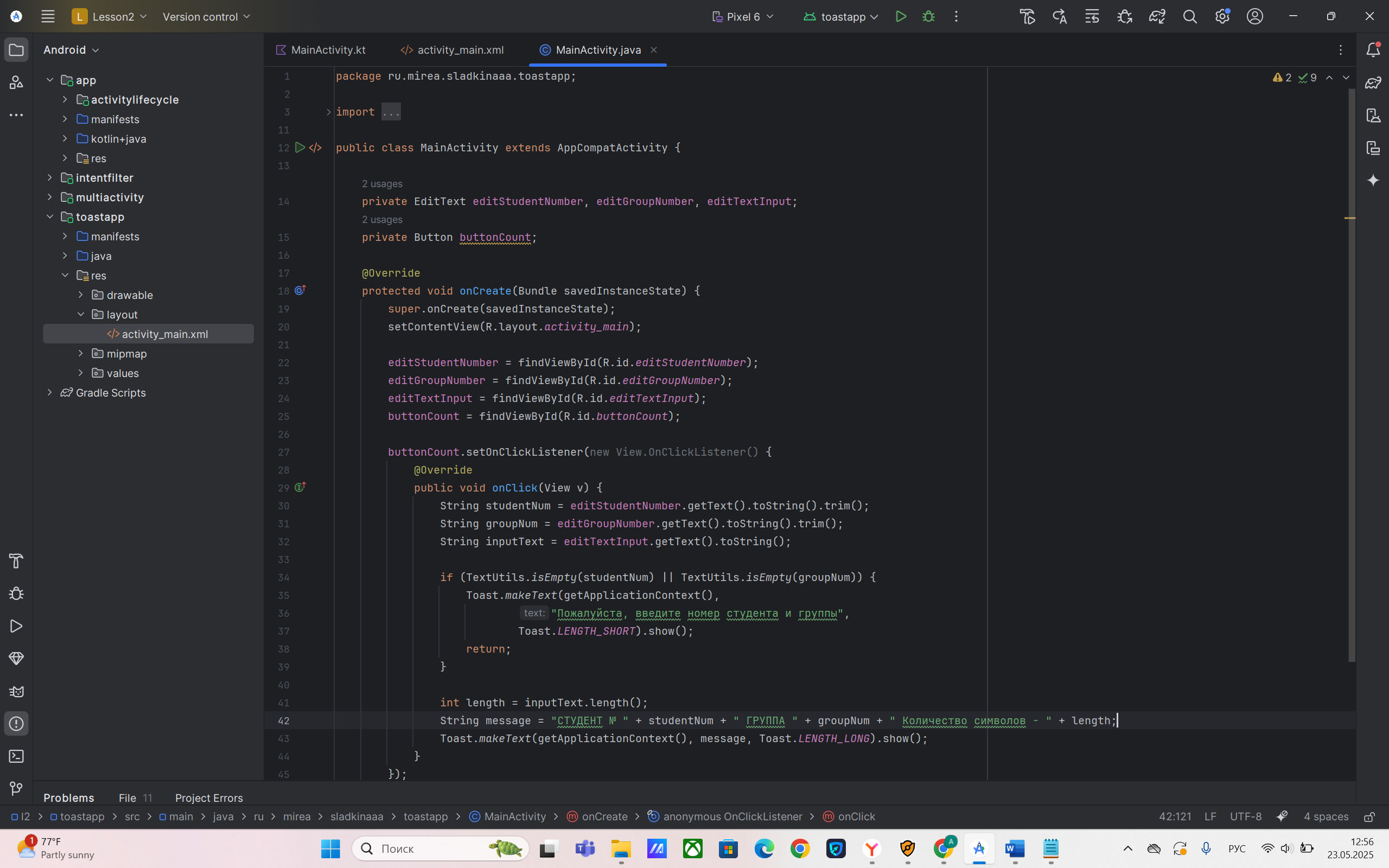


Рисунок 16. Реализация подсчета

Далее в новом модуле в файле разметки activity\_main.xml размещаем button и присваиваем новое значение (см. рис. 17).

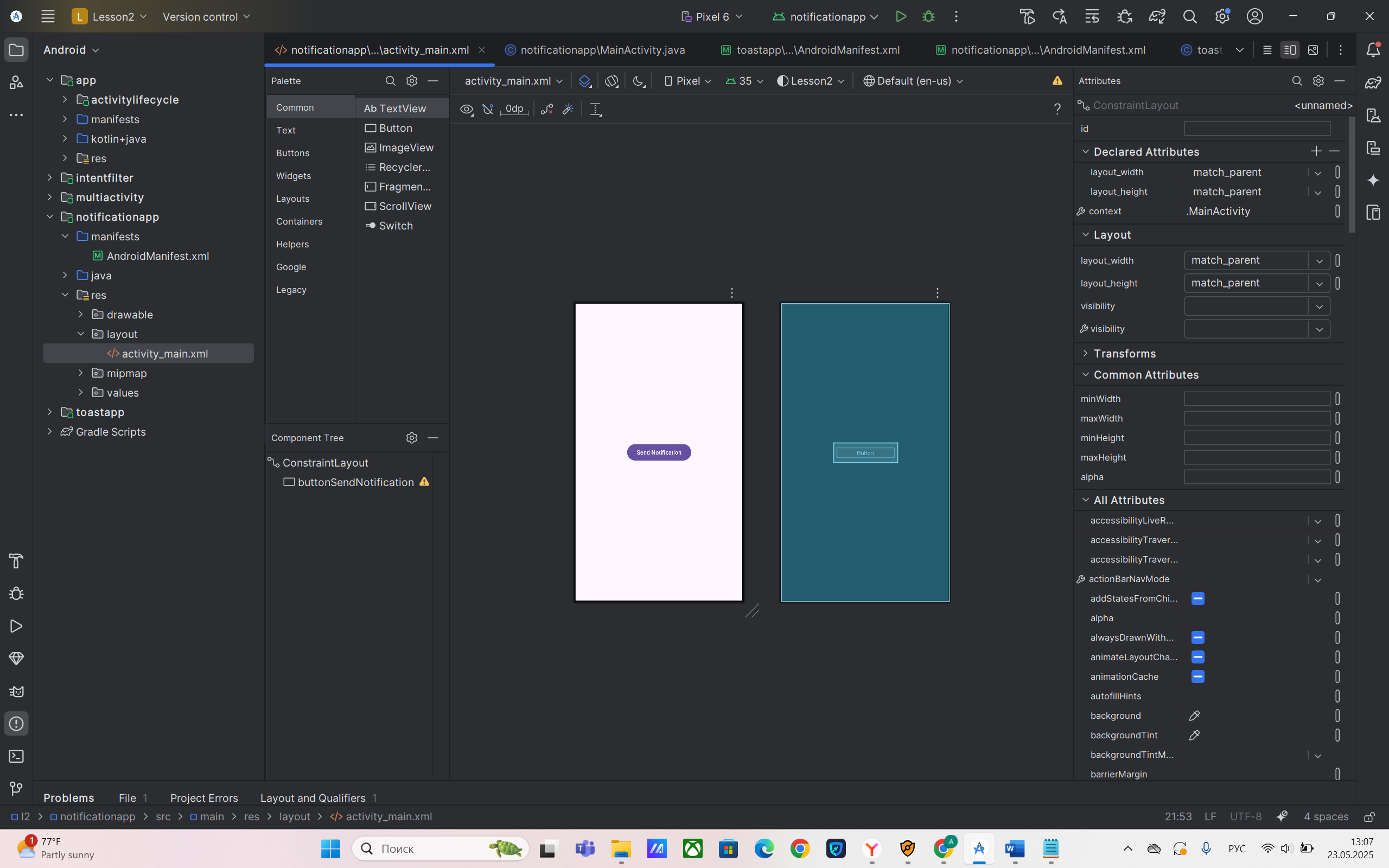


Рисунок 17. Создание новой кнопки

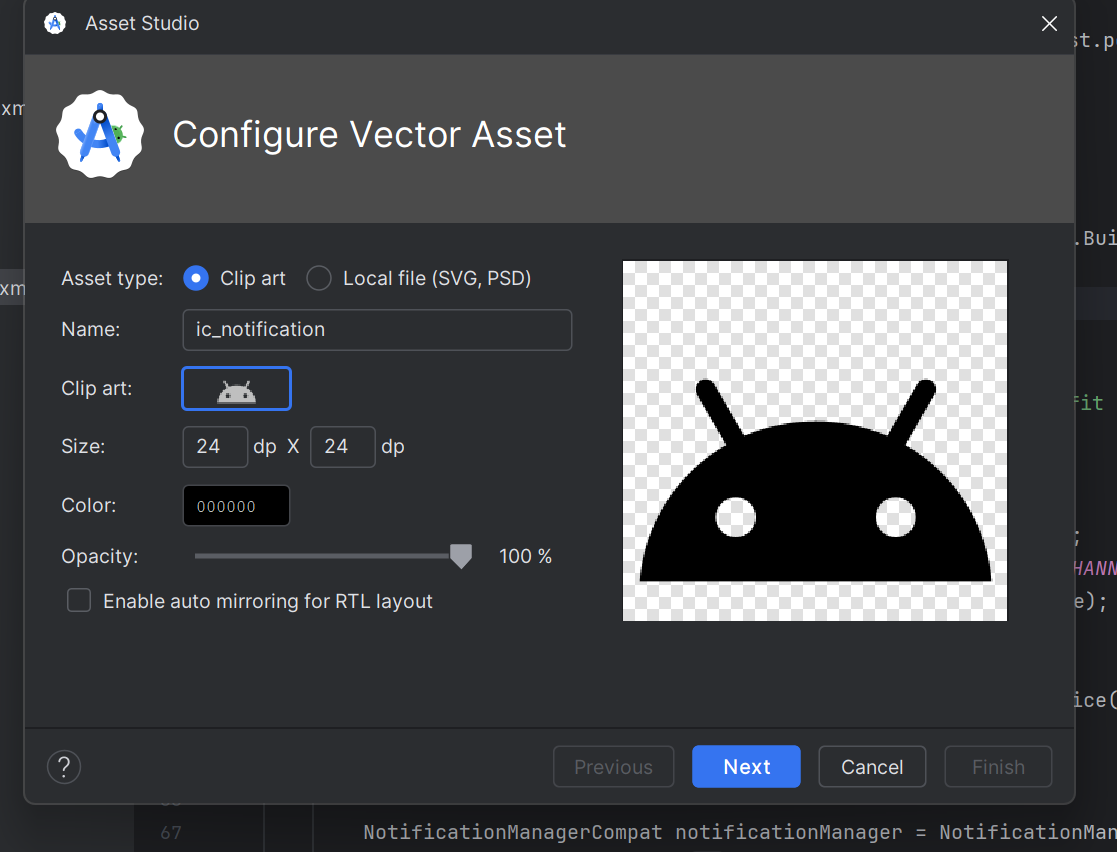


Рисунок 18. Создание иконки

Далее была реализована проверка и запрос разрешения на отправку уведомлений (см. рис. 19).

Проверка разрешений в onCreate() происходит через checkSelfPermission(), который проверяет, есть ли у приложения право на отправку уведомлений. Для Android 13+ (API 33/Tiramisu) требуется явное разрешение POST\_NOTIFICATIONS. Если разрешение есть, то выводится "Разрешения получены". После requestPermissions() запускает системный диалог запроса разрешения. При первом запуске показывается системный диалог "Разрешить приложению отправлять уведомления?", где пользователь выбирает "Разрешить" или "Отклонить"

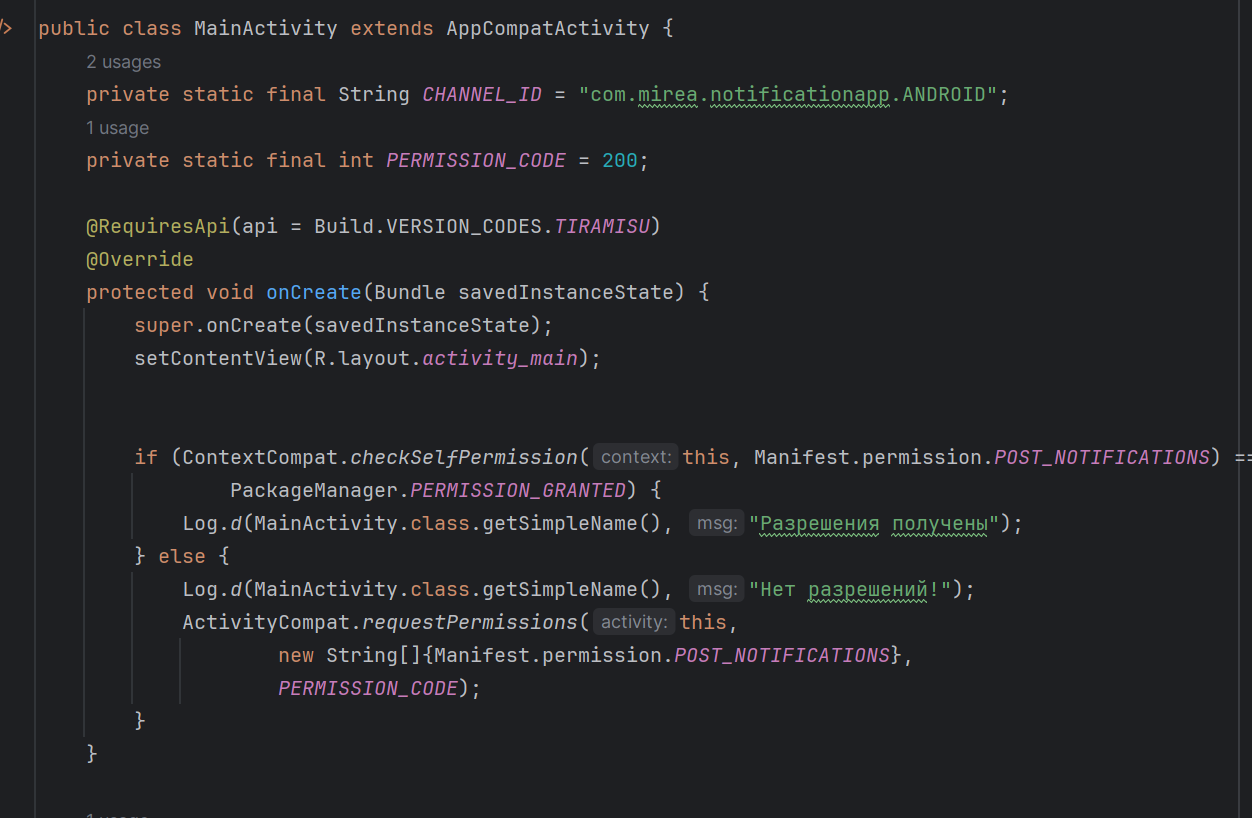


Рисунок 19. Проверка и запрос разрешения на отправку уведомлений

Метод onClickSendNotification вызывается при нажатии на кнопку (через атрибут android:onClick в XML) и выполняет проверку разрешения на отправку уведомлений (для Android 13+), создает уведомления с заголовком, текстом и иконкой. Также происходит создание канала уведомлений и отправка уведомления (см. рис. 20).



Рисунок 20. Создание уведомления

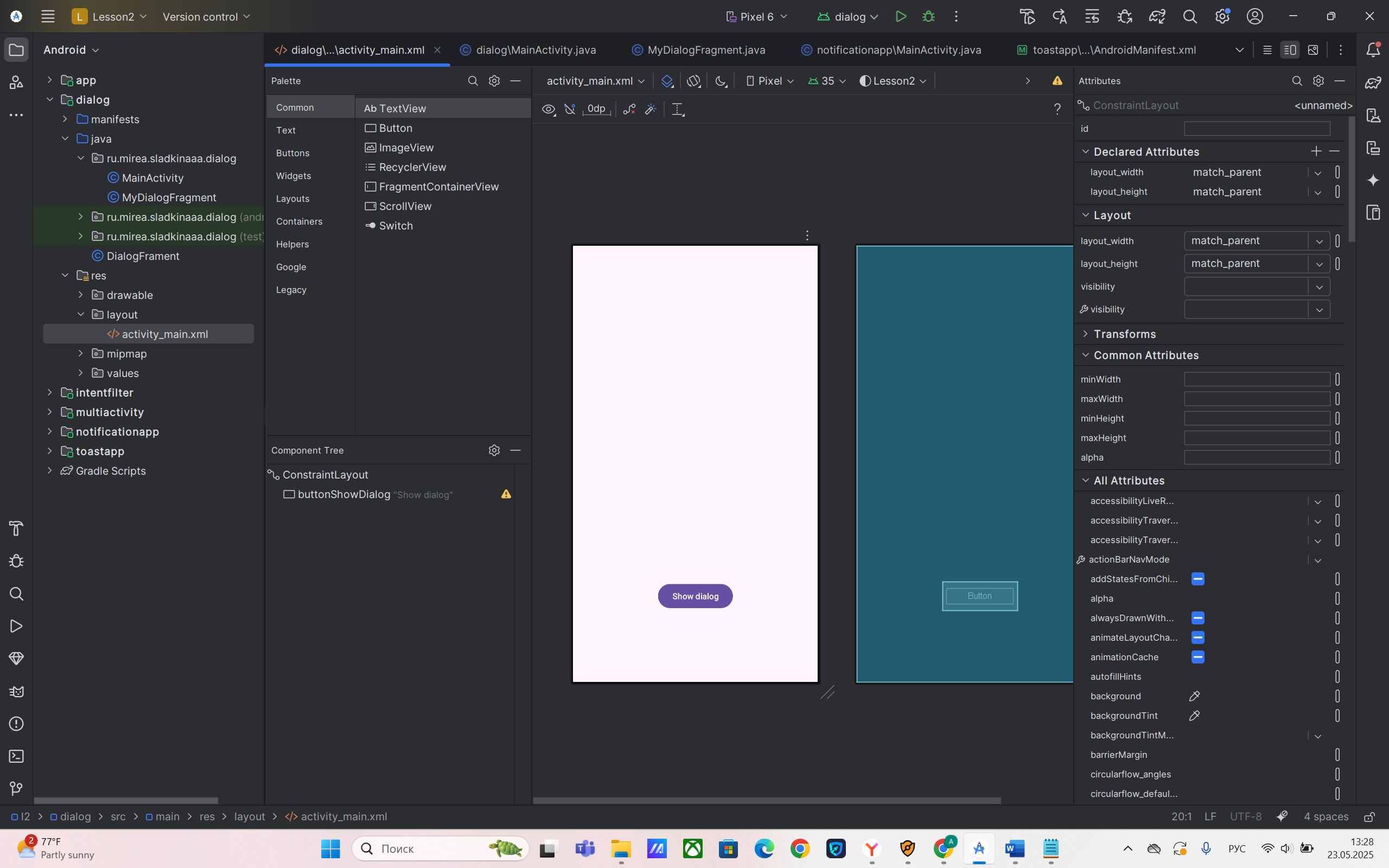
Диалоговое окно занимает часть экрана и обычно используется в модальном режиме. Это означает, что работа приложения приостанавливается до момента, пока пользователь не закроет диалоговое окно. При этом, возможно, потребуется ввести какие-либо данные или просто выбрать один из вариантов ответа. 

Рисунок 21. Создание кнопки для диалогового окна

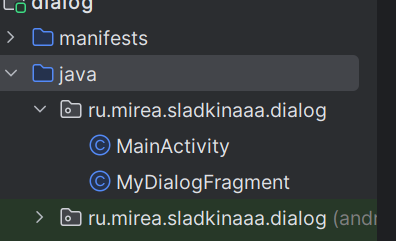


Рисунок 22. Создание нового класса

Класс MyDialogFragment наследуется от DialogFragment, что позволяет создать диалоговое окно, которое управляется фрагментом (см. рис. 23). Метод onCreateDialog переопределяется для создания и настройки диалогового окна. Далее создается AlertDialog.Builder, который используется для конфигурации диалога. Также настраиваются основные элементы диалога setTitle("Заголовок диалога"), setMessage("Это текст диалогового окна."), setIcon(android.R.drawable.ic\_dialog\_info). Далее добавляются три кнопки с обработчиками кликов:

Положительная кнопка "Да" - при нажатии показывает Toast "Вы нажали: Да"

Отрицательная кнопка "Нет" - при нажатии показывает Toast "Вы нажали: Нет"

Нейтральная кнопка "Отмена" - при нажатии показывает Toast "Вы нажали: Отмена".

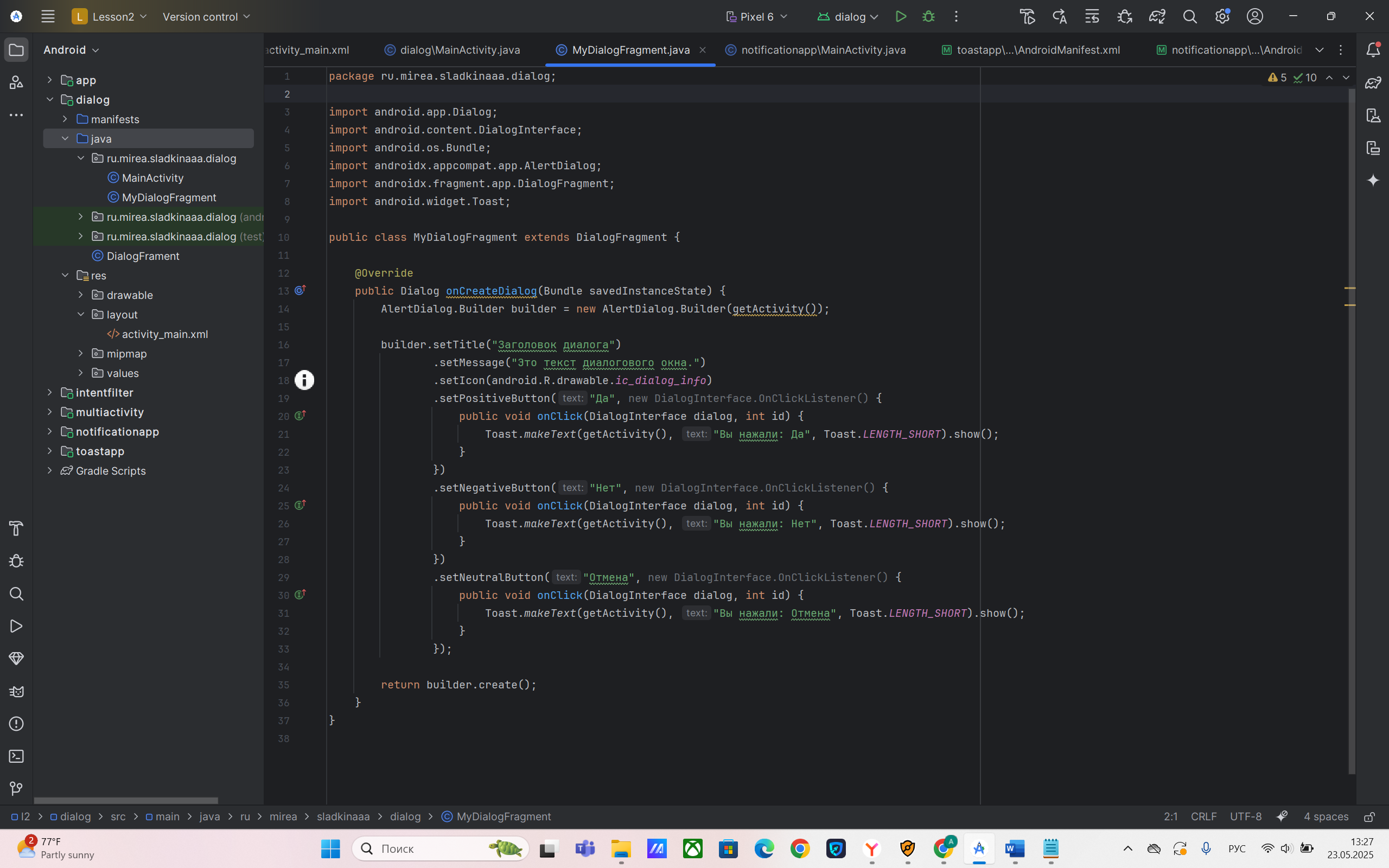


Рисунок 23. Реализация класса MyDialogFragment

Далее рассмотрим класс MainActivity (см. рис. 24). В методе onCreate вызывается родительский метод super.onCreate() и устанавливается макет из ресурсов setContentView(R.layout.activity\_main). Метод onClickShowDialog создает экземпляр MyDialogFragment, отображает диалог с помощью show() и использует getSupportFragmentManager() для управления фрагментами.

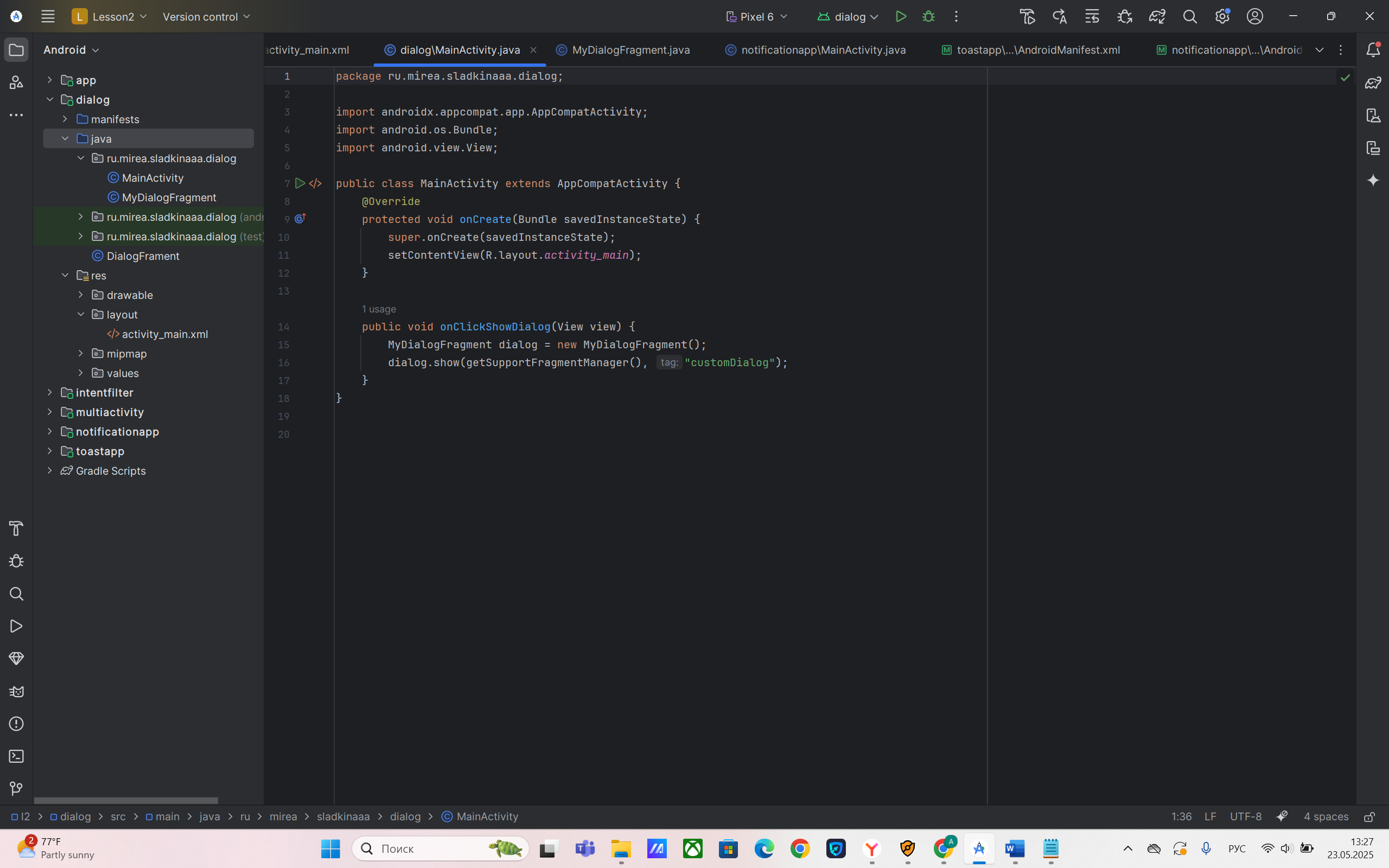


Рисунок 24. Реализация класса MainActivity