|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №9**

по дисциплине «Проектирование и разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы ИКБО-21-23 | Книшенко Д.М. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025г.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc197129114)

[1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ 4](#_Toc197129115)

[1.1 Сохранение состояния приложения 4](#_Toc197129116)

[1.2 Хранение данных в Android 6](#_Toc197129117)

[1.3 Внутреннее хранилище 6](#_Toc197129118)

[1.4 Внешнее хранилище 11](#_Toc197129119)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 14](#_Toc197129120)

[2.1 Внутреннее хранилище 14](#_Toc197129121)

[2.2 Сохранение состояния приложения 23](#_Toc197129122)

[2.3 Внешнее хранилище 24](#_Toc197129123)

[ЗАКЛЮЧЕНИЕ 32](#_Toc197129124)

# ВВЕДЕНИЕ

Данная работа посвящена практической реализации работы с файловой системой Android и сохранения состояния интерфейса. Основные задачи включали: создание, чтение, изменение и удаление файлов во внутреннем и внешнем хранилищах, а также сохранение состояния UI при пересоздании Activity (например, при повороте экрана). Интерфейс создавался с помощью Jetpack Compose, а сохранение состояния реализовано через rememberSaveable. В работе рассмотрены особенности работы с хранилищами, обработка разрешений и показаны результаты выполнения файловых операций и управления состоянием UI — ключевых аспектов для разработки надёжных мобильных приложений.

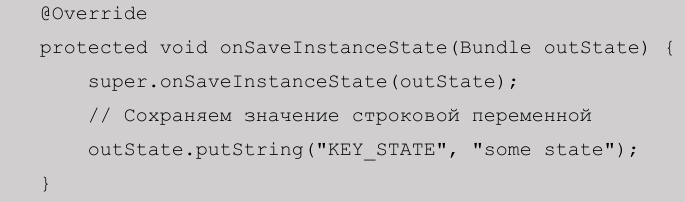
# 1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## 1.1 Сохранение состояния приложения

Сохранение и восстановление состояния приложения являются важными аспектами разработки Android-приложений, особенно при обработке изменений конфигурации, таких как поворот экрана, изменение языка и других сценариев, которые приводят к пересозданию активности.

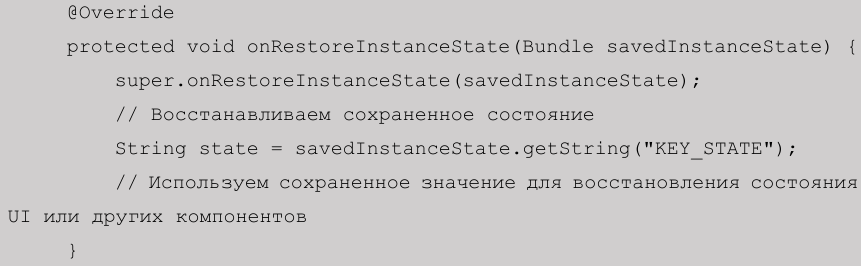
Android предоставляет несколько механизмов для управления состоянием приложения, включая использование методов onSaveInstanceState() и onRestoreInstanceState(). Эти методы позволяют сохранять и восстанавливать данные о состоянии пользовательского интерфейса, обеспечивая бесперебойное взаимодействие пользователя с приложением.

Метод onSaveInstanceState() (Рисунок 1) вызывается системой перед тем, как активность будет уничтожена, чтобы дать возможность сохранить состояние пользовательского интерфейса в объект Bundle. В этот объект можно добавлять различные типы данных, такие как строки, числа, сериализуемые объекты и др.

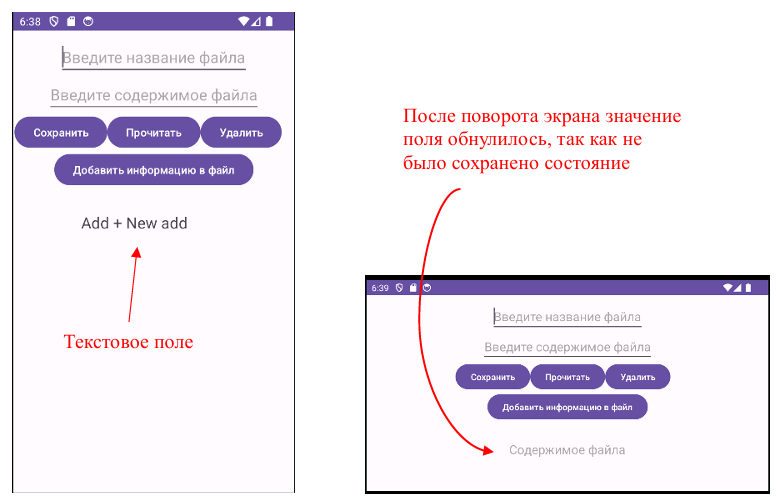
  
Рисунок 1 – Метод onSaveInstanceState

В методе onSaveInstanceState() сохраняем состояние. Для этого вызываем у параметра Bundle метод putString(key, value), первый параметр которого - ключ, а второй - значение сохраняемых данных.

Метод onRestoreInstanceState() вызывается после метода onStart(), когда активность воссоздается после пересоздания. Этот метод получает объект Bundle, содержащий данные о состоянии, которые были сохранены в onSaveInstanceState() (Рисунок 2).

  
Рисунок 2 – Метод onRestoreInstanceState

Таким образом, например, при смене ориентации экрана все записанные данные не пропадут, как в примере на Рисунке 3 при смене ориентации с вертикальной на горизонтальную – описание фрагмента останется свернутым или развернутым, как это было до поворота экрана.

  
Рисунок 3 – Сохранения состояния при смене ориентации

## 1.2 Хранение данных в Android

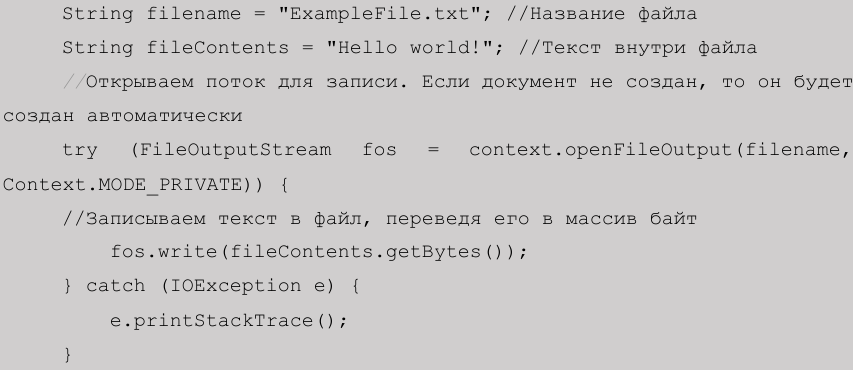
Более удобным способом сохранять все состояния приложения является их запись в отдельный файл, который не будет затронут при смене метода жизненного цикла активности.

Работа с файловой системой в Android является ключевым элементом в процессе разработки мобильных приложений. Это необходимость обусловлена разнообразными задачами, с которыми сталкиваются разработчики: от простого сохранения настроек пользователя и состояний приложения до сложных операций с медиафайлами, документами и другими данными, требующими постоянного хранения между сессиями. Кроме того, эффективное использование файловой системы позволяет реализовывать функции загрузки ресурсов из сети и их последующего кэширования, что существенно улучшает производительность приложения и удобство его использования.

В Android предусмотрены два основных типа хранилища для работы с файлами: внутреннее и внешнее. Внутреннее хранилище гарантирует безопасность данных, делая их доступными только для вашего приложения, и обеспечивает их удаление при деинсталляции приложения. Внешнее хранилище, в свою очередь, предлагает больший объем памяти и возможность обмена файлами между приложениями и даже передачу данных между устройствами.

## 1.3 Внутреннее хранилище

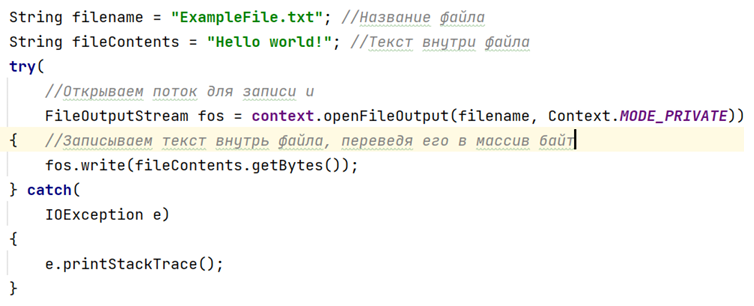
Внутреннее хранилище предназначено для индивидуальных данных приложения, к которым не предполагается доступ из других приложений. Данные, сохраненные во внутреннем хранилище, удаляются при удалении приложения. Для работы с внутренним хранилищем используются следующие методы: 1) Создание и запись файла (Рисунок 4).

  
Рисунок 4 – Создание и запись файла

Система позволяет создавать файлы с двумя разными режимами:

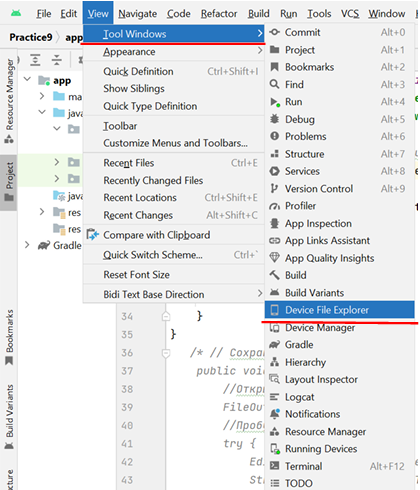
* MODE\_PRIVATE: файлы могут быть доступны только владельцу приложения (режим по умолчанию);
* MODE\_APPEND: данные могут быть добавлены в конец файла. Если файл уже существует, то он будет перезаписан.

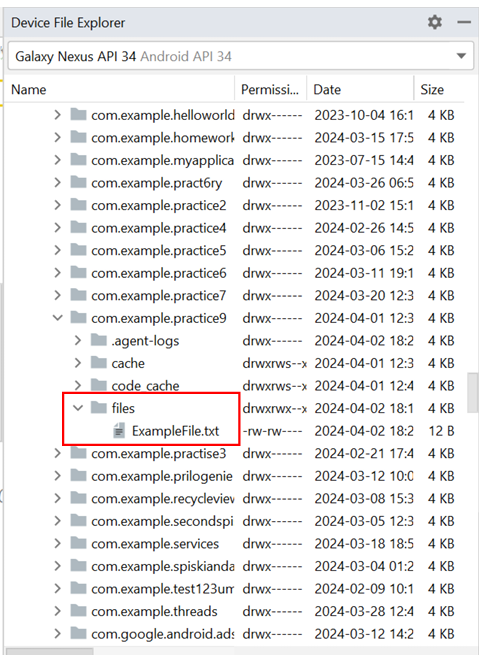
Если же нам надо дописать файл, тогда надо использовать режим MODE\_APPEND. Для автоматического закрытия файла и освобождения ресурса объект FileOutputStream создается с помощью конструктции try...catch.

  
Рисунок 5 – Использование конструкции try...catch

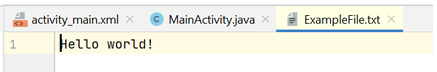
В итоге весь текст будет /data/data/название\_пакета/files/ExampleFile.txt сохранен в файле.

Чтобы увидеть его на подключенном Device→ Tool Windows → устройстве перейдем в Android Stud в меню к пункту View File Explorer (Рисунки 6-7).

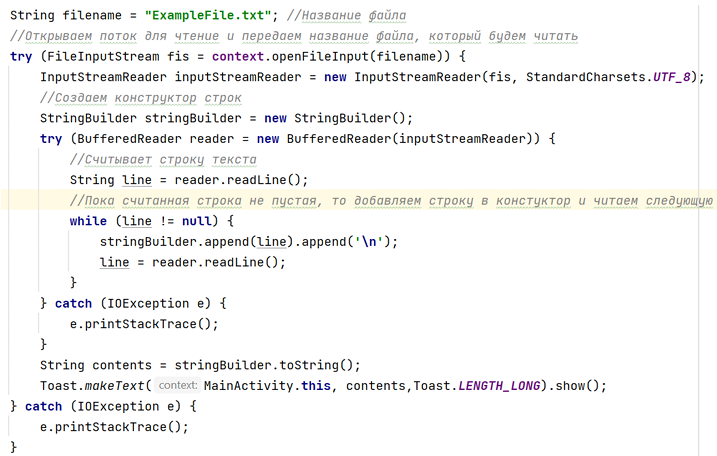
  
Рисунок 6 – Отображение файла на устройстве (часть1 )

  
Рисунок 7 – Отображение файла на устройстве (часть 2 )

И если дважды нажать на файл, то можно увидеть его содержимое (Рисунок 8).

  
Рисунок 8 – Отображение содержимого файла

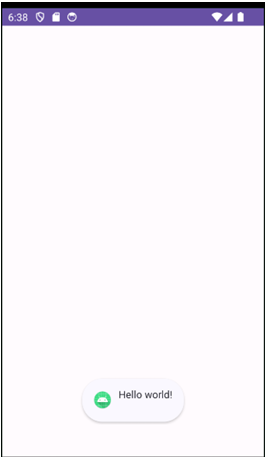
2) Чтение файла (Рисунок 9):

  
Рисунок 9 – Чтение файла

Класс FileInputStream создаёт объект класса InputStream, который можно использовать для чтения байтов из файла.

InputStreamReader – это переход от потоков байтов к потокам символов: он считывает байты и декодирует их в символы, используя указанный Charset. Используемая им кодировка может быть указана по имени или может быть задана явно, или может быть принята кодировка платформы default charset.

На Рисунке 10 представлен результат чтения.

  
Рисунок 10 – Результат чтения файла

3) Удаление файла (Рисунок 11):

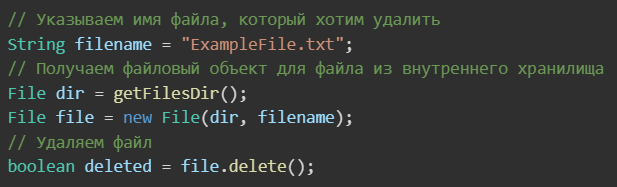
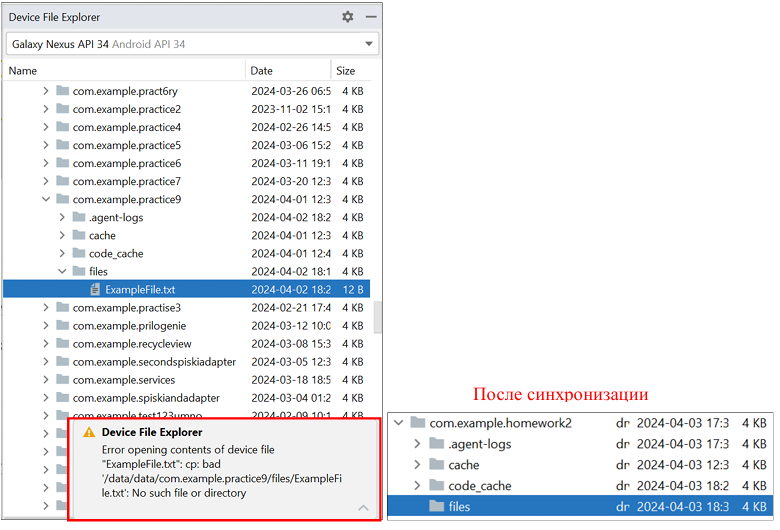


Рисунок 11 – Удаление файла из внутреннего хранилища в Android

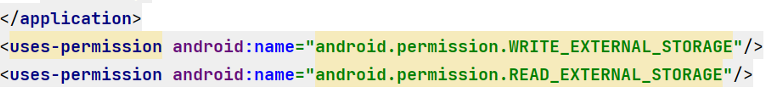
Теперь при попытке открыть файл будет выводиться ошибка. После синхронизации папка с файлами будет пуста. Для синхронизации нажмите правой Synchronize.→кнопкой мыши на папку files (Рисунок 12).

  
Рисунок 12 – Результат удаления файла

## 1.4 Внешнее хранилище

В прошлой теме было рассмотрено сохранение и чтение файлов из каталога приложения. По умолчанию такие файлы доступны только самому приложения.

Внешнее хранилище используется для сохранения файлов, которые могут быть общими для нескольких приложений или требуют сохранения даже после удаления вашего приложения. Прежде чем создавать файлы, убедитесь, что у вашего приложения есть необходимые разрешения для работы с внешним хранилищем. Начиная с Android 6.0 (API уровня 23), требуется запросить эти разрешения во время выполнения (для этого добавьте в файл Манифеста разрешения READ\_EXTERNAL\_STORAGE и WRITE\_EXTERNAL\_STORAGE) (Рисунок 13).

Рисунок 13 – Запрос разрешений в AndroidManifest

Для Android 10 и выше рекомендуется использовать механизм Scoped Storage, который не требует явных разрешений для доступа к определенным типам файлов, таким как фотографии и видео, через MediaStore API.

1) Создание и запись файла (Рисунок 14):



Рисунок 14 – Создание и запись в текстовый файл в публичной директории "Documents"

На Рисунке 14 метод определяет путь к общедоступной папке "Downloads" во внешнем хранилище, создает эту папку, если она еще не существует, а затем проверяет наличие файла "ExampleFile.txt" в ней. Если файл отсутствует, метод создает его и записывает туда строку "Hello world!", обрабатывая при этом возможные ошибки файлового ввода-вывода.

2) Чтение файла (Рисунок 15):

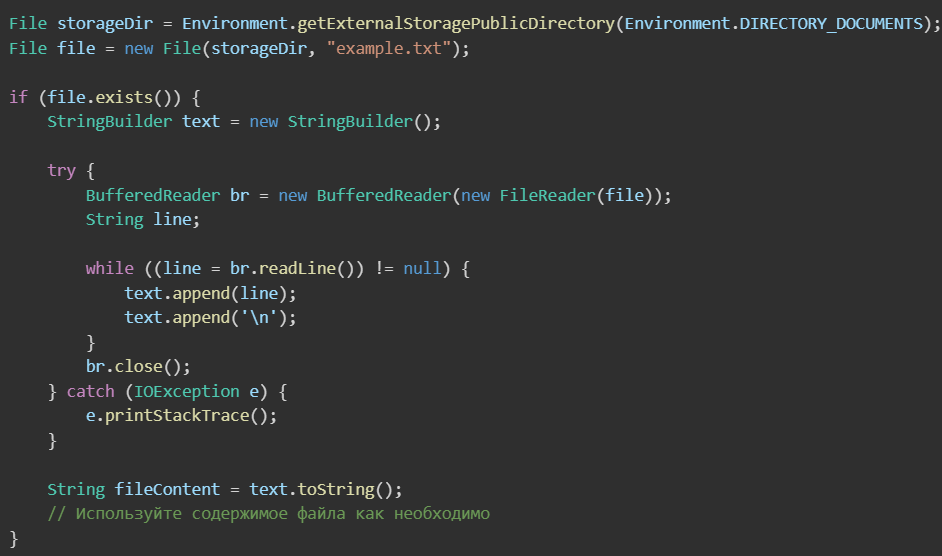


Рисунок 15 – Чтение данных из текстового файла в публичной директории "Documents"

Метод на Рисунке 15 сначала проверяет, существует ли файл "ExampleFile.txt" в общедоступной папке "Downloads" во внешнем хранилище. Если файл найден, метод открывает его для чтения, считывает содержимое построчно, собирает весь текст в единую строку (которая в данном фрагменте кода далее не используется) и обрабатывает возможные ошибки, возникающие в процессе чтения.

3) Удаление файла (Рисунок 16):

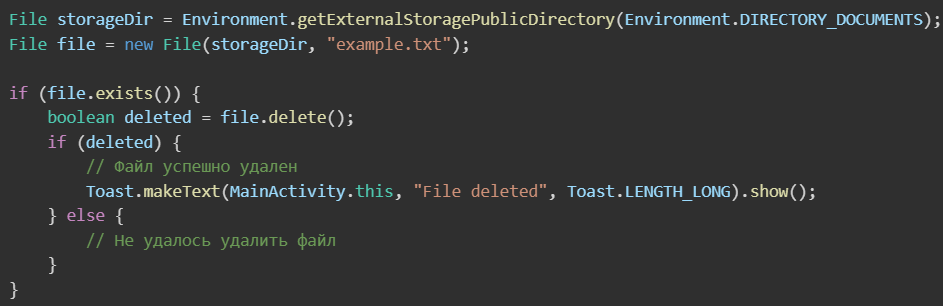


Рисунок 16 – Удаление файла из публичной директории "Documents"

Метод на Рисунке 16 ищет файл "ExampleFile.txt" в общедоступной папке "Downloads" во внешнем хранилище. Если файл существует, метод пытается его удалить; в случае успеха пользователю отображается всплывающее уведомление (Snackbar) с сообщением "File deleted", а если удаление не удалось — соответствующая ошибка записывается в системный лог с тегом "RRR".

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Внутреннее хранилище

Для реализации данного задания создаем файл разметки activity\_main.xml, где расположены кнопки для сохранения, прочтения, добавления и удаления изменений в файле. Также там есть поле для ввода названия файла, для ввода содержимого файла и для отображения содержимого файла. На Рисунках 17-18 показана реализации файла разметки activity\_main.xml.

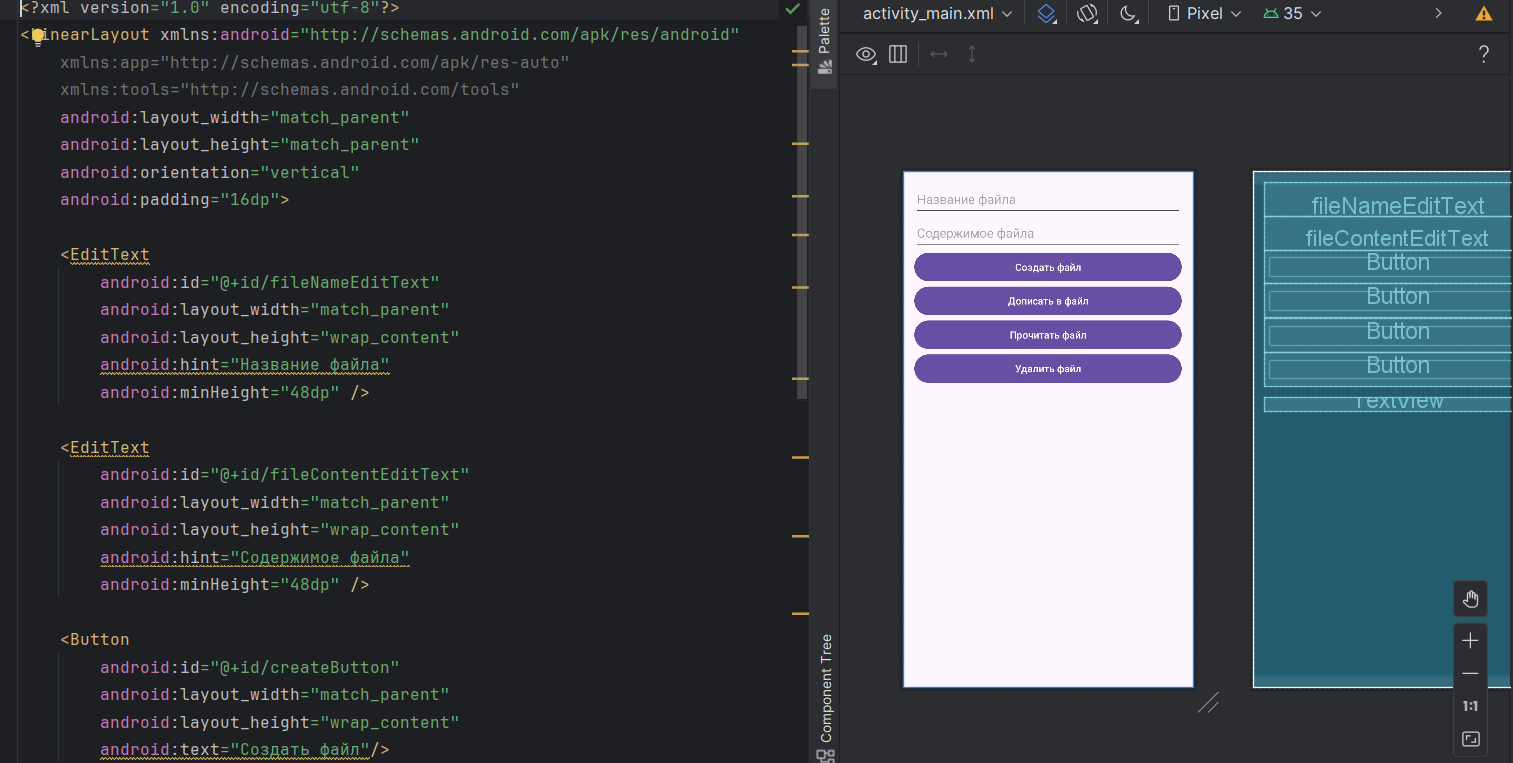


Рисунок 17 – Реализация activity\_main.xml, ч.1

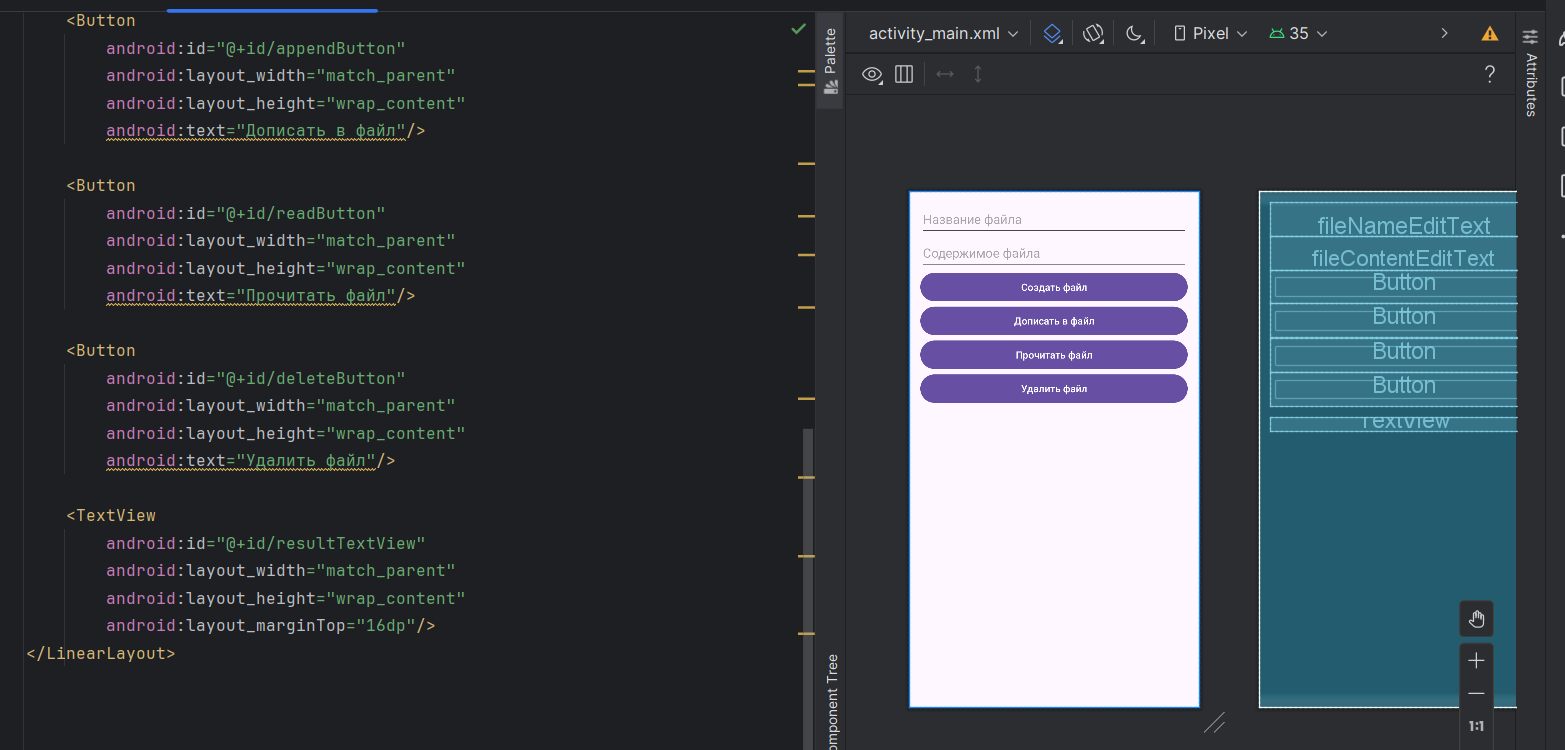


Рисунок 18 – Реализация activity\_main.xml, ч.2

На рисунке 19 покажем первую часть кода файла MainActivity.java.

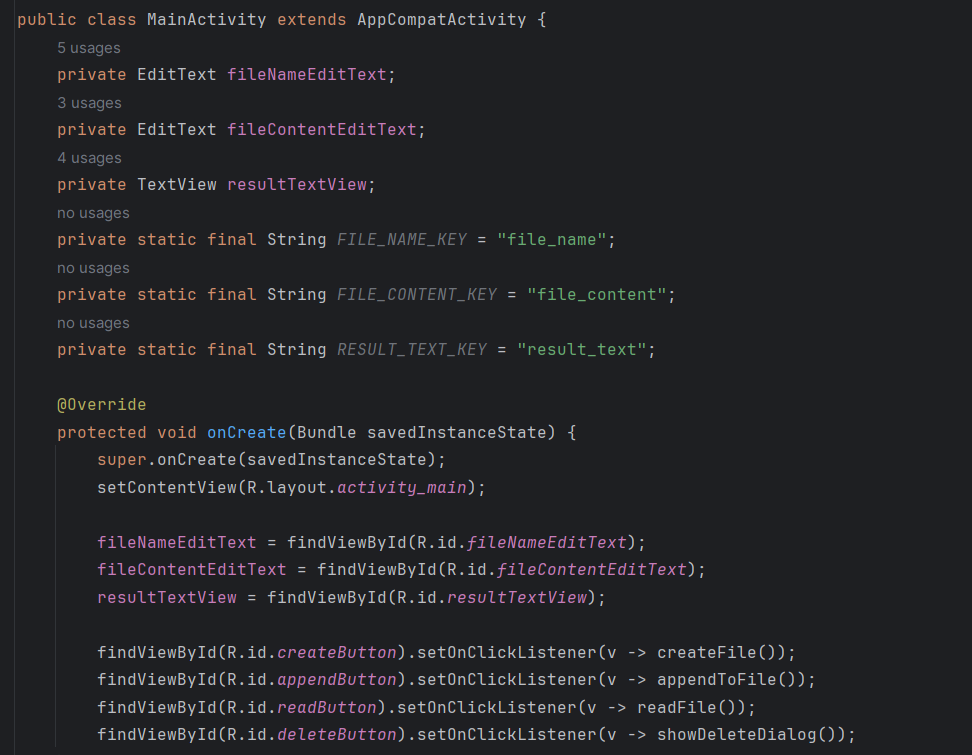


Рисунок 19 – Код файла MainActivity.java, ч.1

На рисунке 20 покажем вторую часть кода файла MainActivity.java.



Рисунок 20 – Код файла MainActivity.java, ч.2

На рисунке 21 покажем третью часть кода файла MainActivity.java.

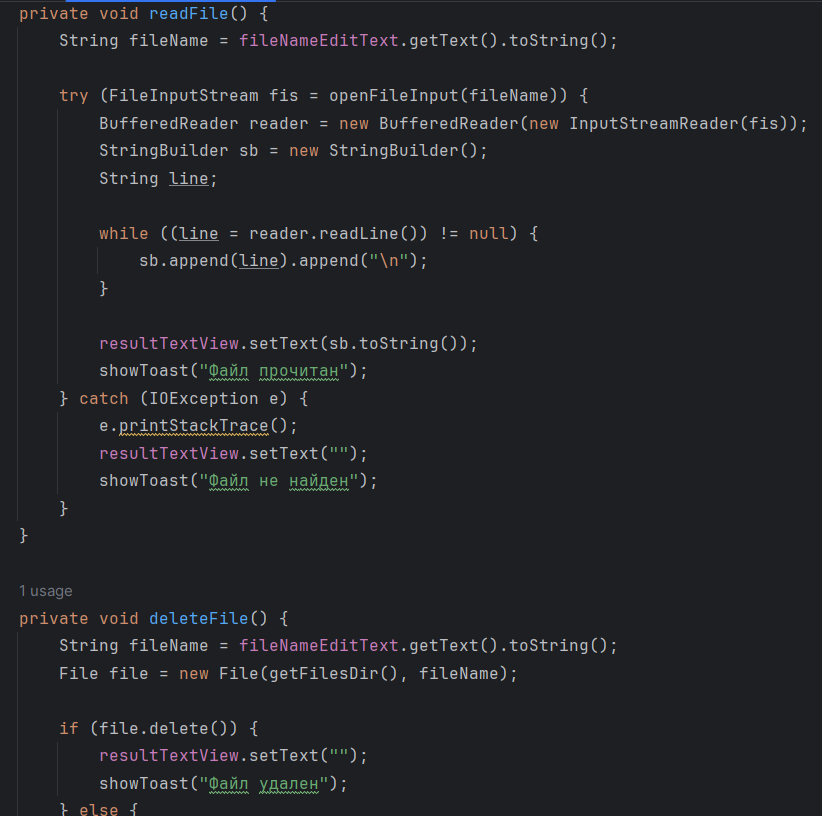


Рисунок 21 – Код файла MainActivity.java, ч.3

На рисунке 22 покажем четвертую часть кода файла MainActivity.java.



Рисунок 22 – Код файла MainActivity.java, ч.4

На рисунке 23 покажем результат запуска проекта. Сразу впишем все необходимое в поля. На рисунке видно, что файл соханился.

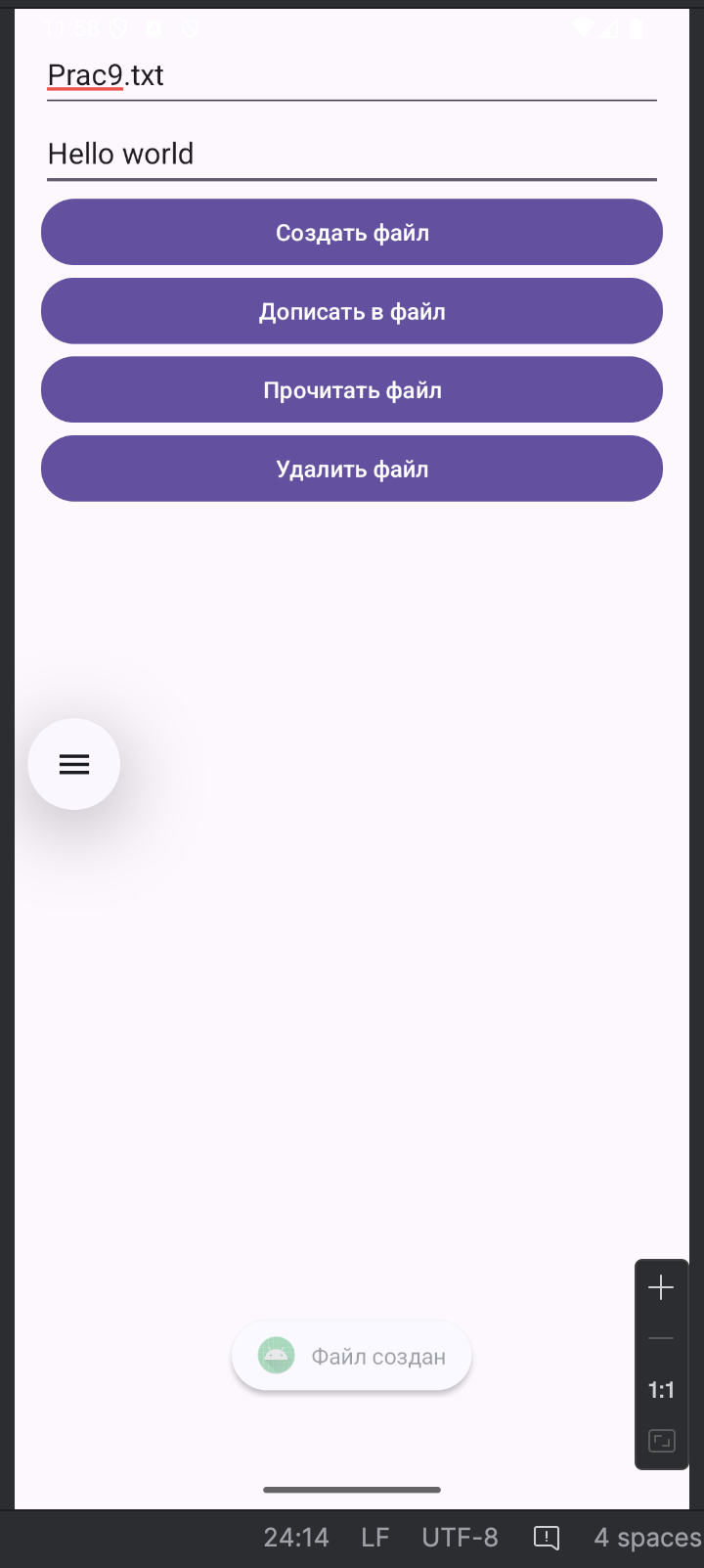


Рисунок 23 – Результат запуска проекта с сохранением файла

Проверим сохранился ли файл это посмотреть через AndroidStudio, что показано на рисунке 24.

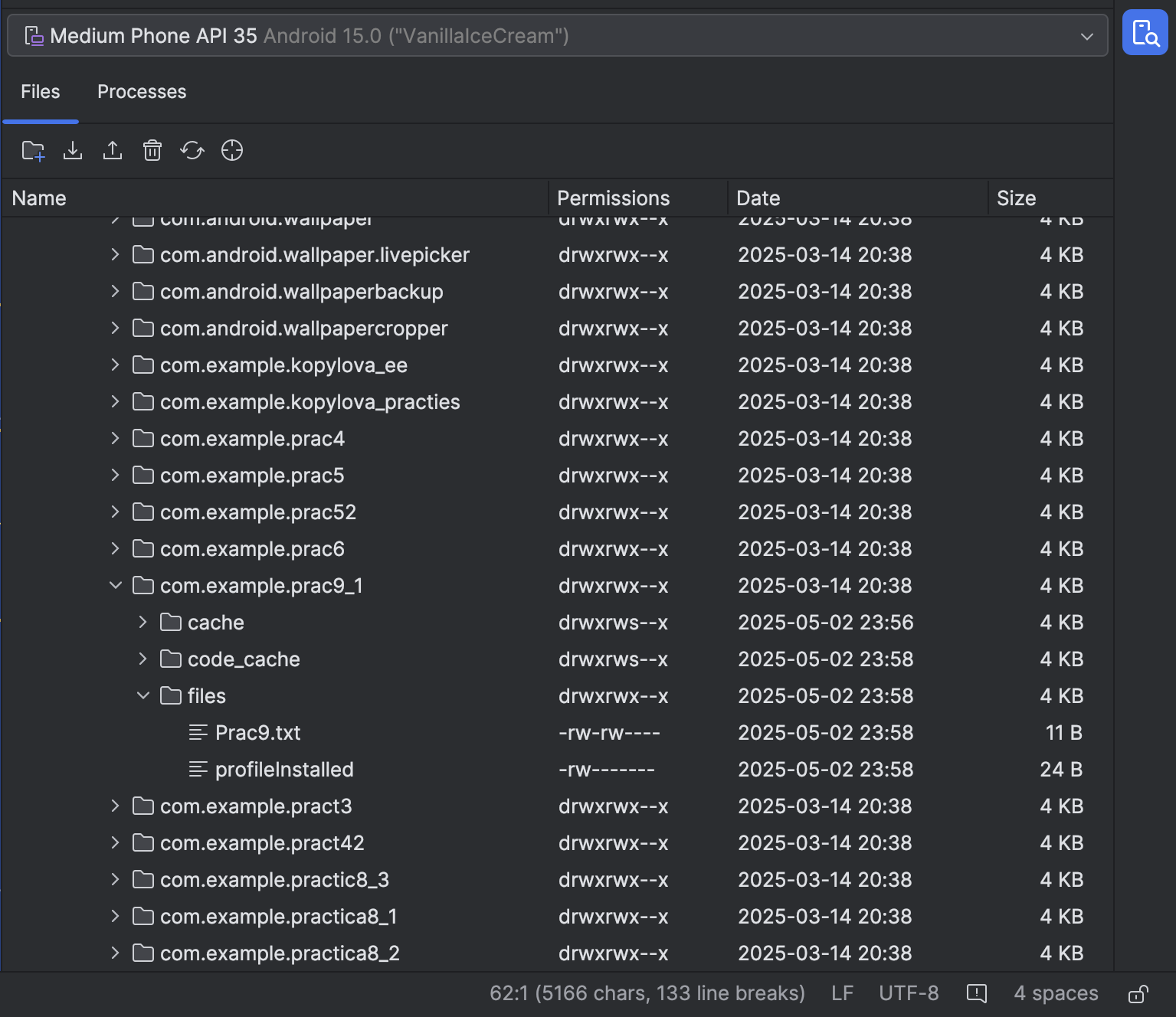


Рисунок 24 – Результат запуска проекта с сохранением файла

На рисунке 25 покажем чтение файла.

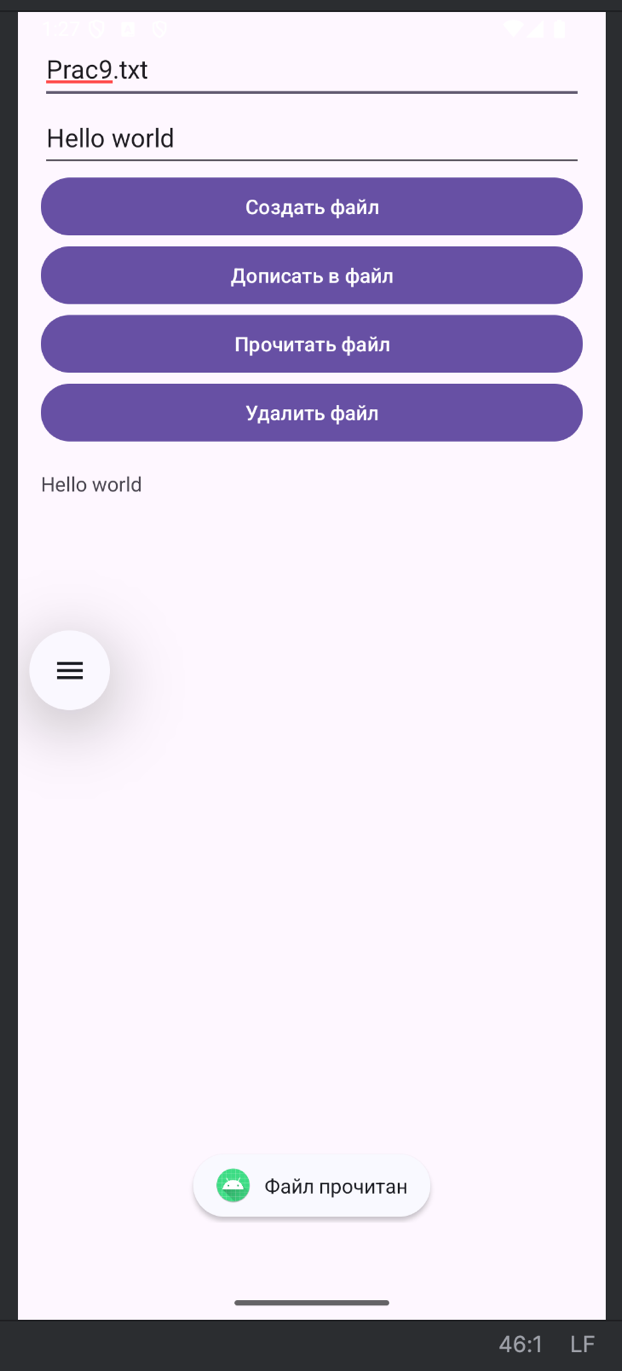


Рисунок 25 – Демонстрация чтения файла

Попробуем добавить информацию и после этого прочитать файл. На рисунке 26 продемонстрируем этот процесс.

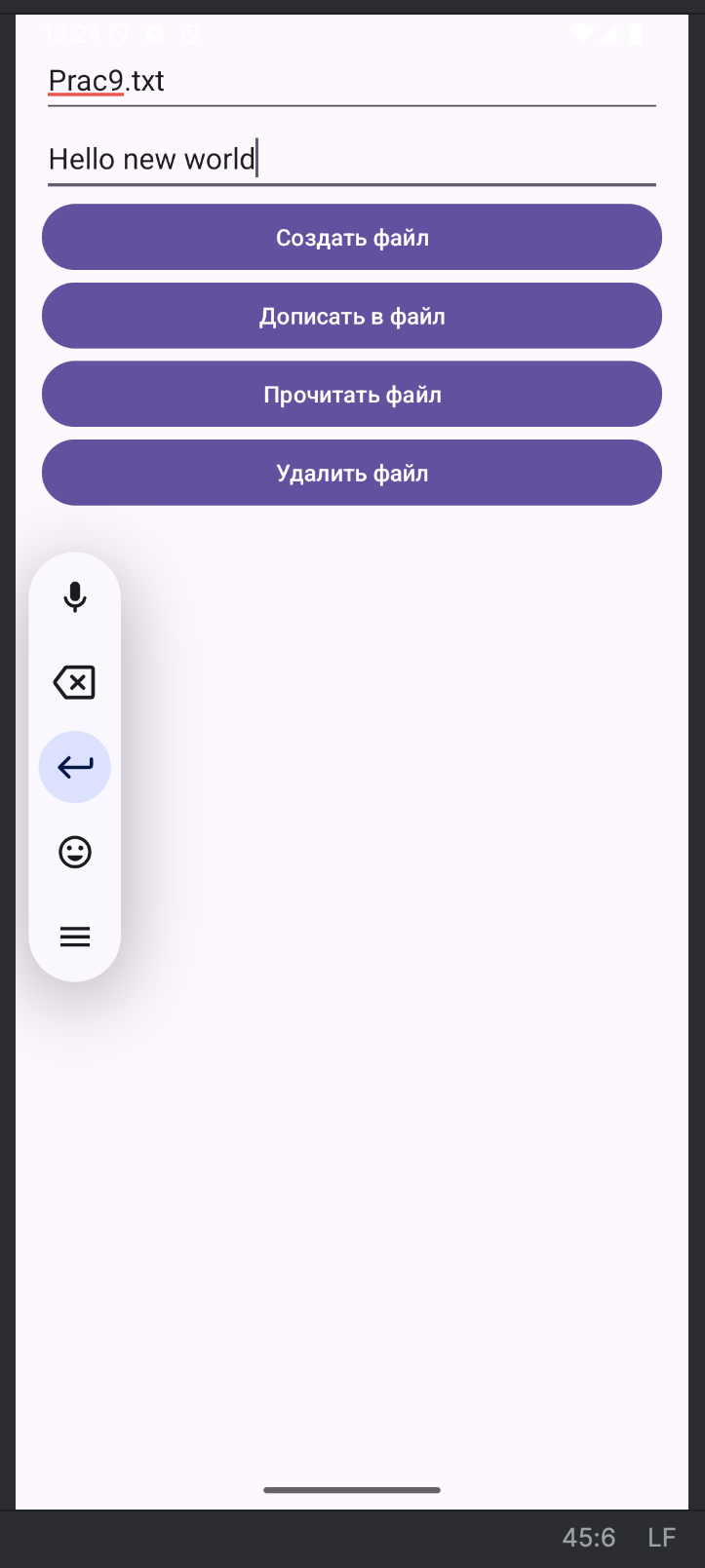


Рисунок 26 – Демонстрация добавления информации в файл

На рисунке 27 покажем отображение добавленной информации.

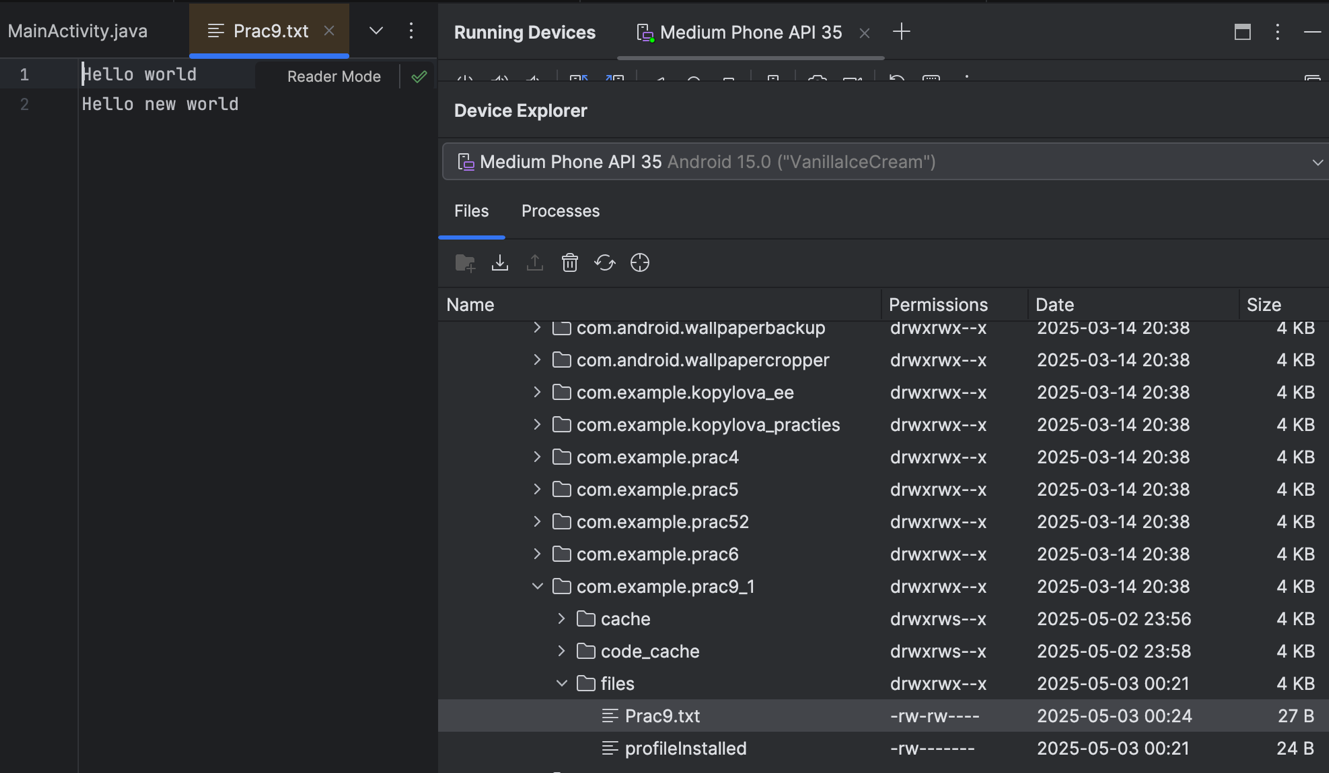


Рисунок 27 – Проверка добавления в файл

На Рисунке 28 продемонстрирована попытка удаления файла.

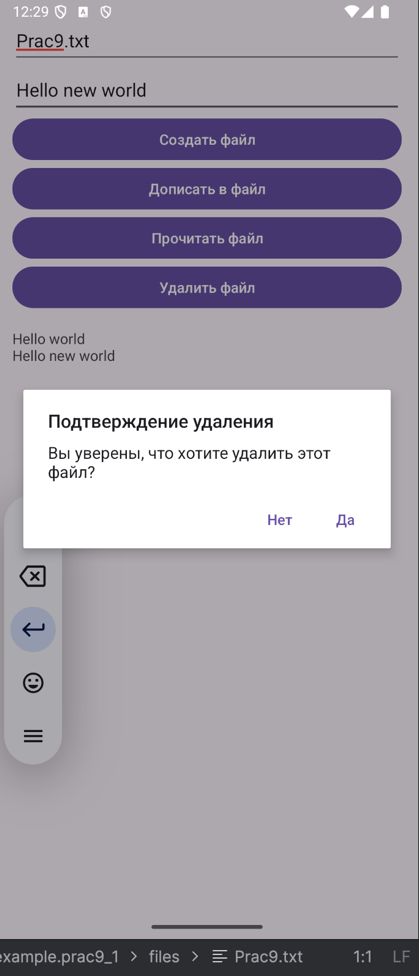


Рисунок 28 – Диалоговое окно подтверждения удаления файла

Чтобы проверить удалился ли файл на самом деле мы воспользуемся AndroidStudio. Как показано на рисунке 29, файл действительно удалился.

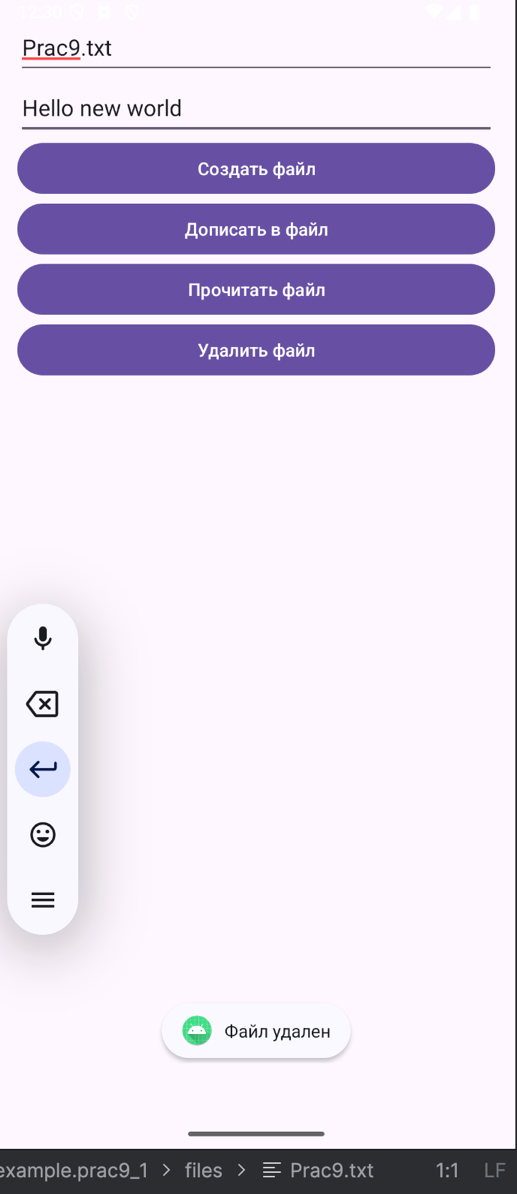


Рисунок 29 – Демонстрация, что файл удалился

## 2.2 Сохранение состояния приложения

Для сохранения информации в горизонтальном положении нам не нужно менять файл разметки. Нам достаточно внести небольшое изменения файл MainActivity.java. На рисунке 30 показан блок, который мы немного изменили в файл.

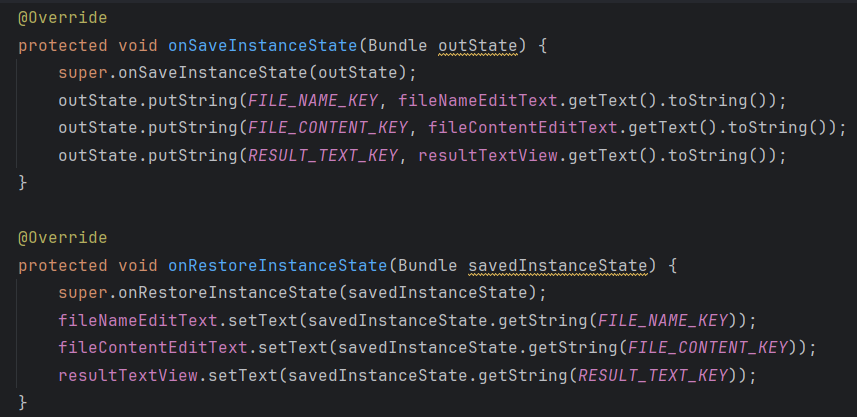


Рисунок 30 – Измененная часть файла MainActivity.java

На рисунке 31 продемонстрируем, что при повороте экрана все сохраняется.

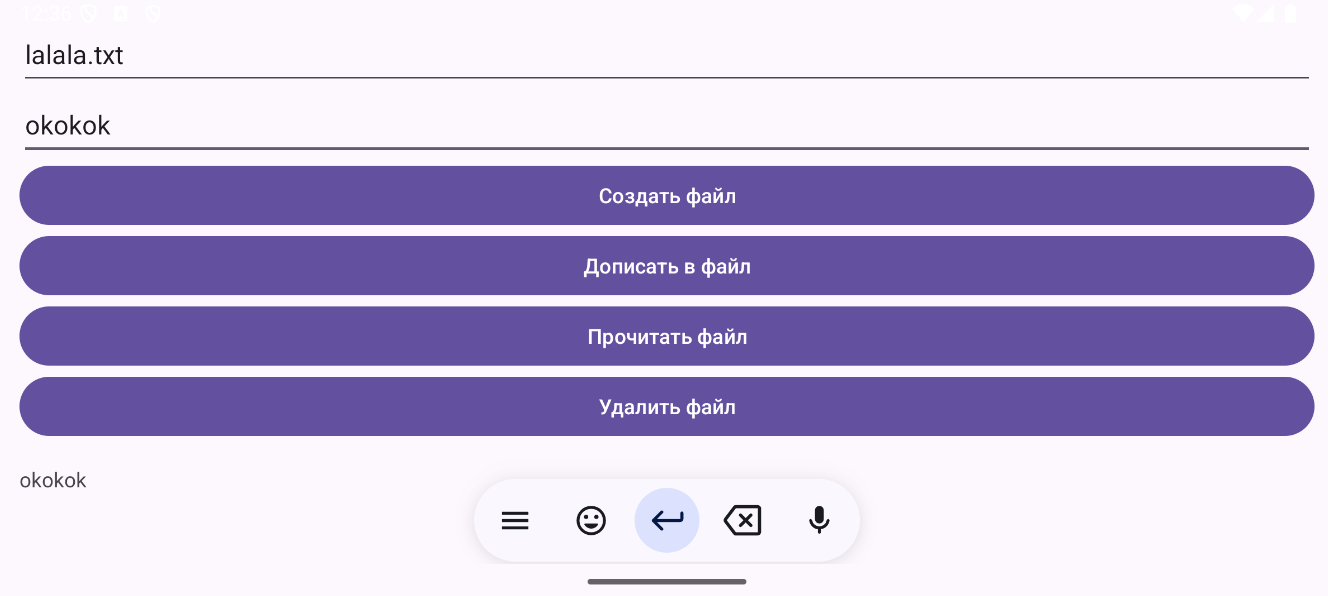


Рисунок 31 – Демонстрация поворота экрана

## 2.3 Внешнее хранилище

Для реализации данного задания создаем 2 приложения с немного разными файлами разметки файлами разметки activity\_main.xml. Одно приложение для записи файла, а именно FileWriter. Другое для чтения – FileReader.

На рисунке 32 покажем реализацию и результат работы файла разметки activity\_main.xml из приложения FileWriter.

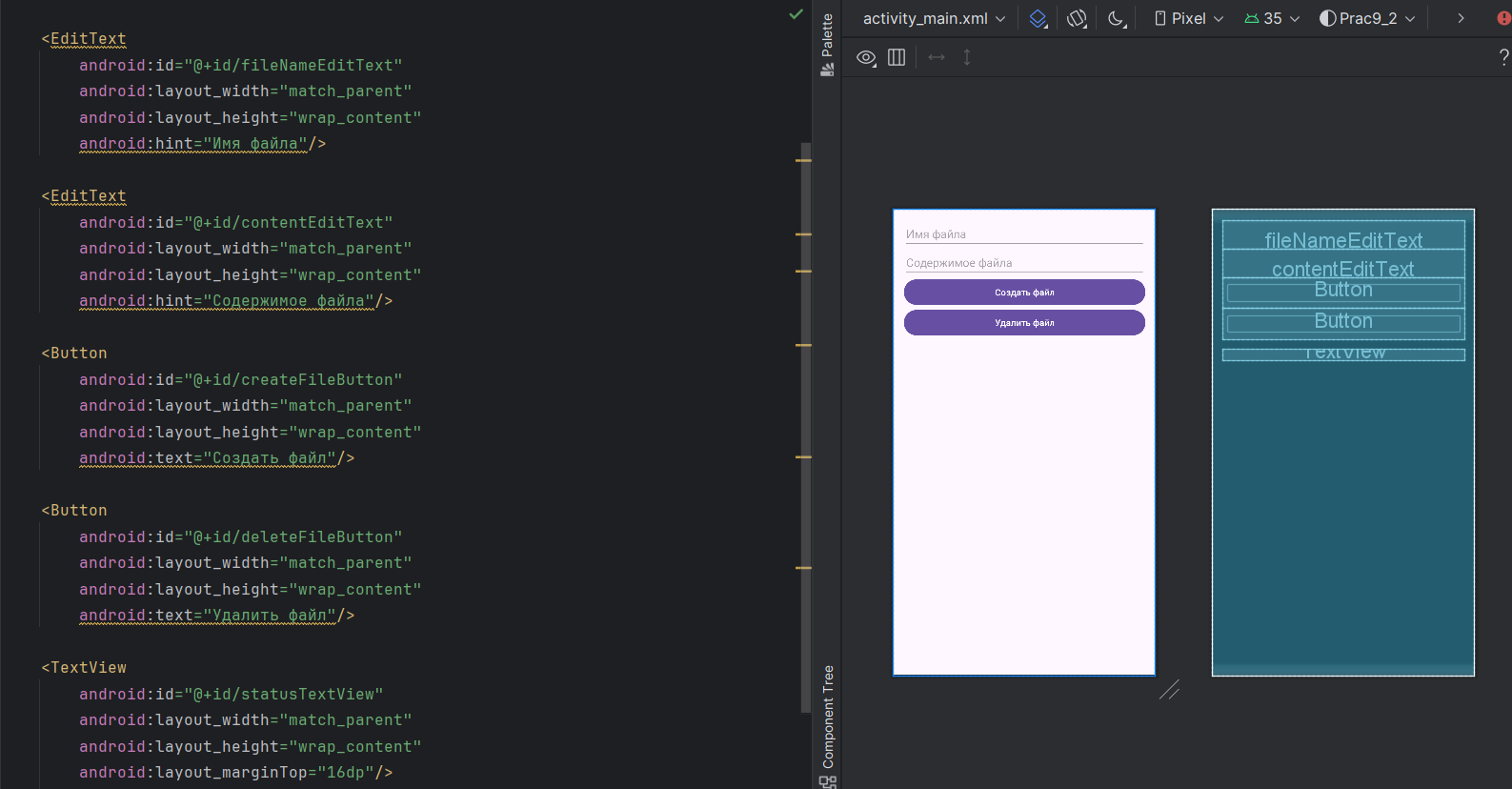


Рисунок 32 – Реализация и результат работы файла разметки activity\_main.xml

На рисунке 33 покажем первую часть кода файла MainActivity.java из FileWriter.



Рисунок 33 – Код файла MainActivity.java, ч.1

На рисунке 34 покажем вторую часть кода файла MainActivity.java из FileWriter.

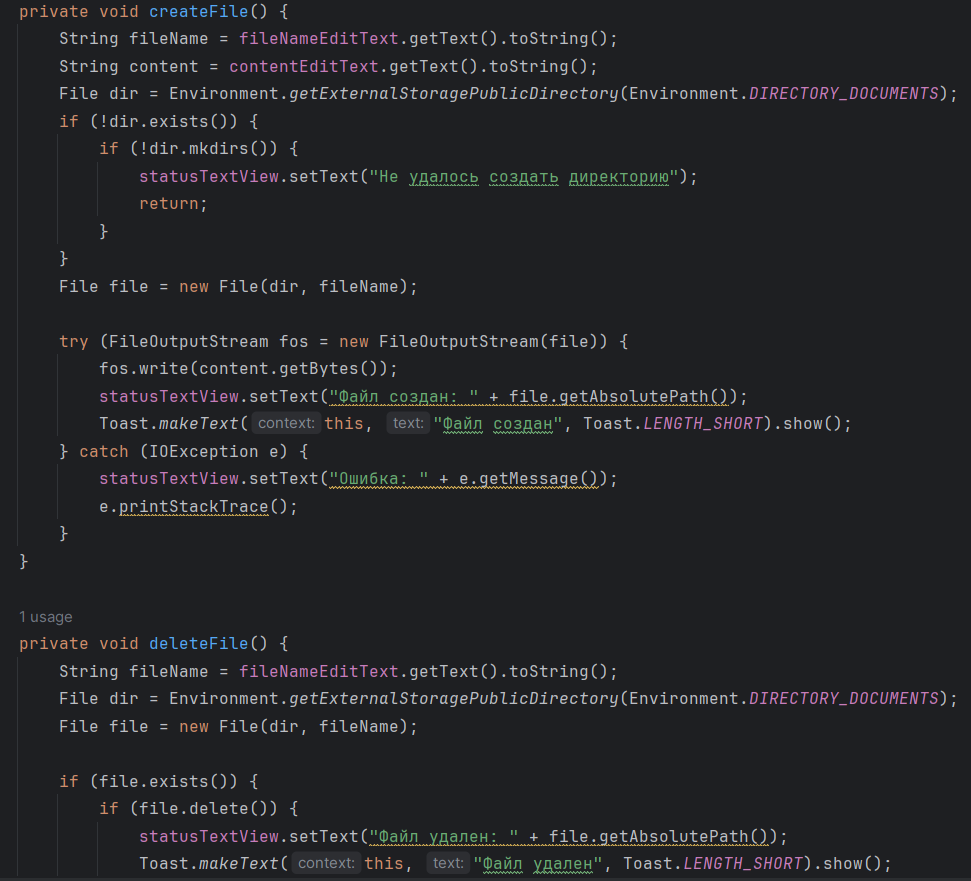


Рисунок 34 – Код файла MainActivity.java, ч.2

На рисунке 35 покажем реализацию и результат работы файла разметки activity\_main.xml из приложения FileReader.

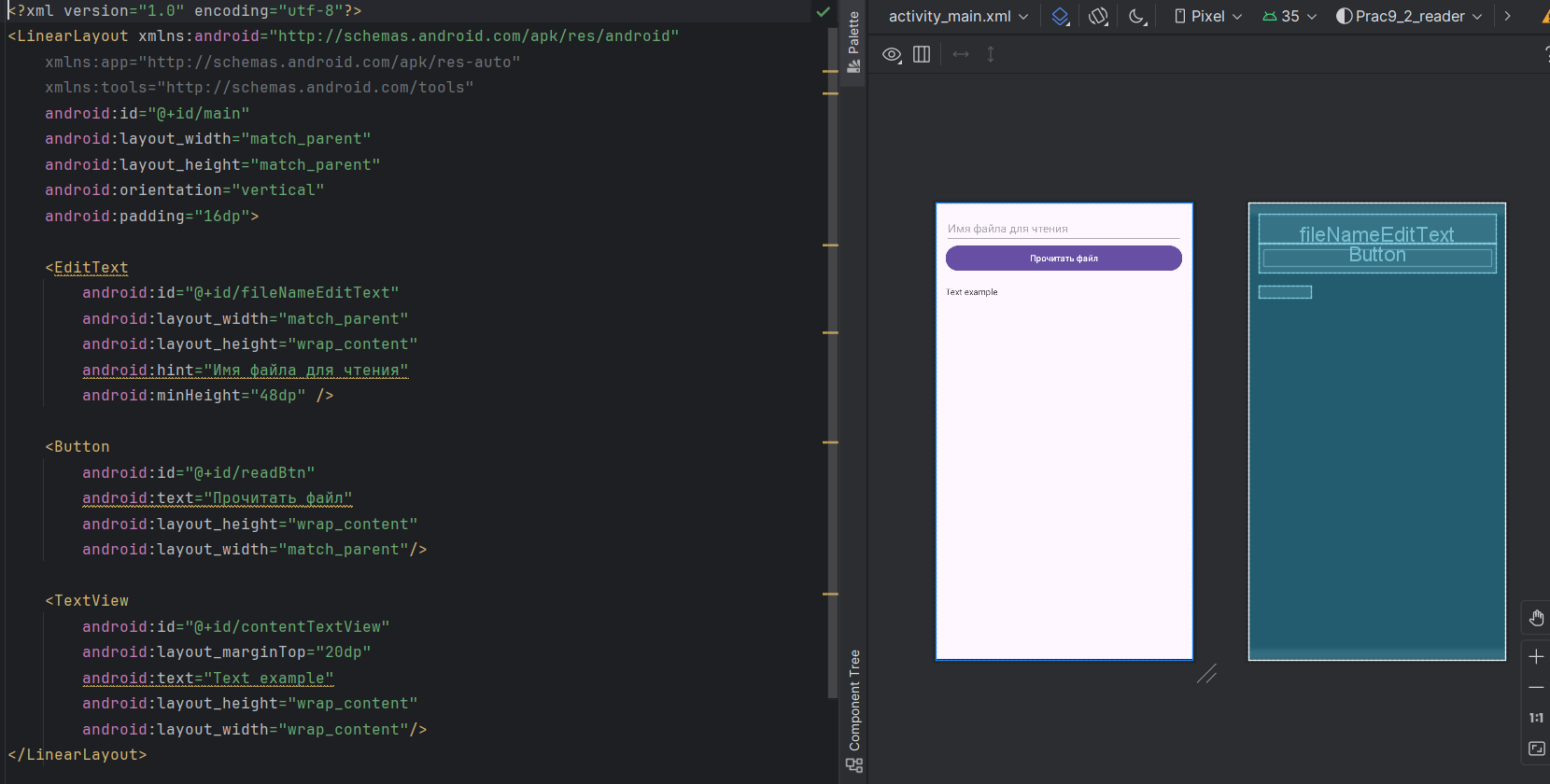


Рисунок 35 – Реализация и результат работы файла разметки activity\_main.xml

На рисунке 36 покажем первую часть кода файла MainActivity.java из FileReader.

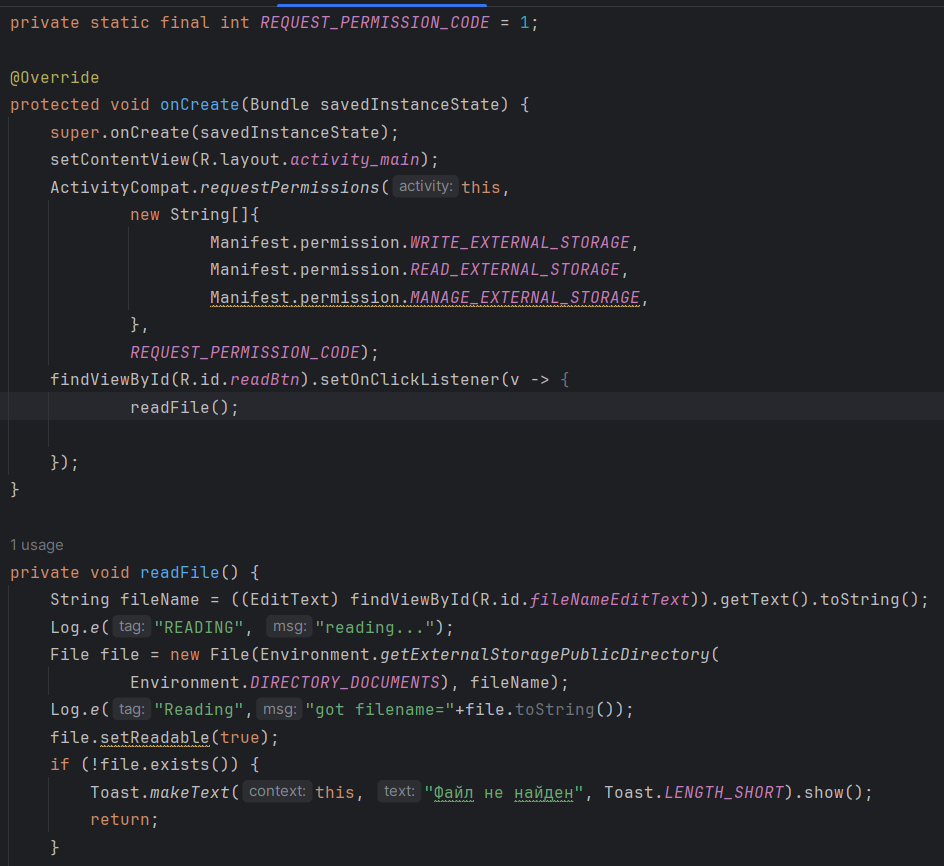


Рисунок 36 – Реализация и результат работы файла разметки activity\_main.xml

На рисунке 37 покажем вторую часть кода файла MainActivity.java из FileReader.

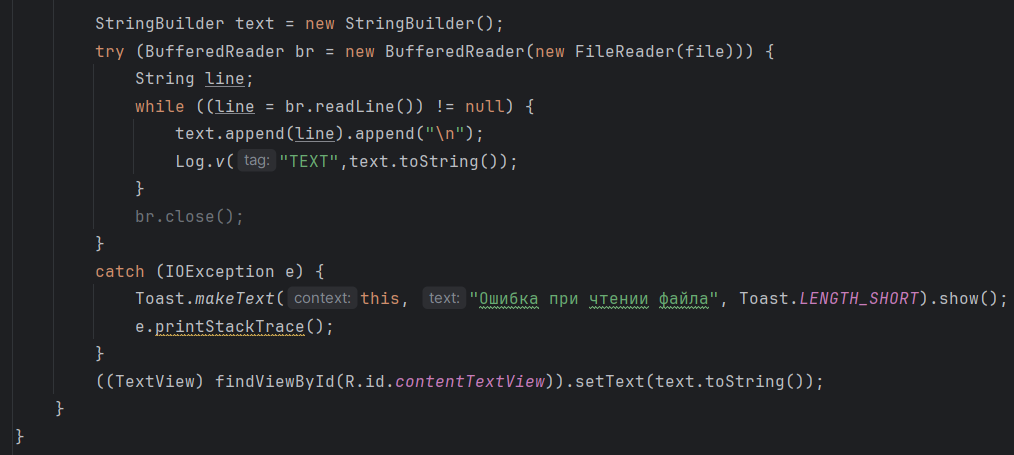


Рисунок 37 – Реализация и результат работы файла разметки activity\_main.xml

На рисунке 38 покажем демонстрацию работы проекта FileWriter.



Рисунок 38 – Демонстрация работы проекта FileWriter

На рисунке 39 покажем, что файл действительно был успешно создан.

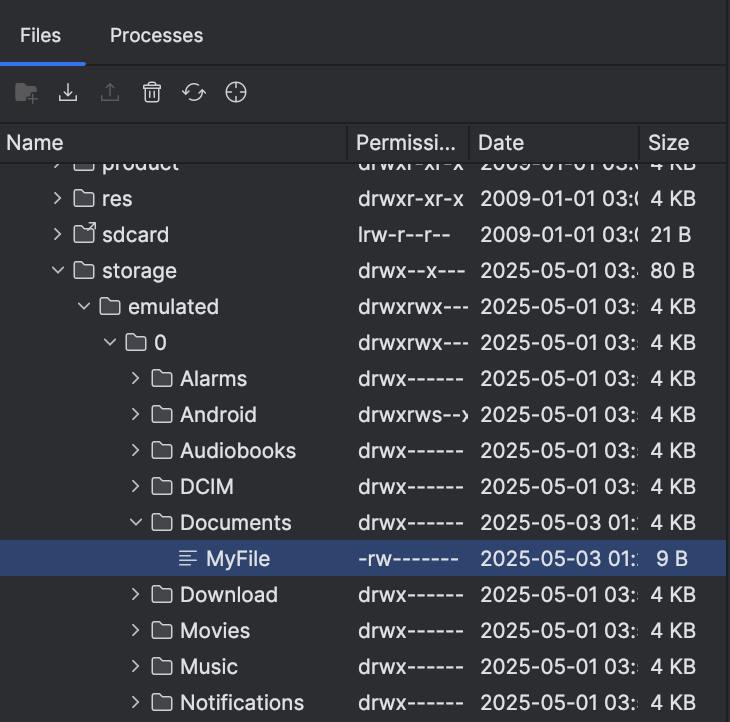


Рисунок 39 – Проверка создания файла

На рисунке 40 покажем демонстрацию работы проекта FileReader.



Рисунок 39 – Демонстрация работы проекта FileReader

# ЗАКЛЮЧЕНИЕ

В процессе практической работы были получены навыки взаимодействия с файловой системой Android-приложений, включая использование внутреннего и внешнего хранилищ, а также сохранение состояния при изменении конфигурации устройства. Эти умения были эффективно использованы при создании мобильного приложения, ориентированного на устойчивую работу и удобство пользователя.