|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-21-23 | Муравьев А. О. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc197711011)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc197711012)

[1.1 Сохранение состояния приложения 4](#_Toc197711013)

[1.2 Хранение данных в Android 6](#_Toc197711014)

[1.3 Внутреннее хранилище 6](#_Toc197711015)

[1.4 Внешнее хранилище 12](#_Toc197711016)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 16](#_Toc197711017)

[2.1 Внутреннее хранилище 16](#_Toc197711018)

[2.2 Сохранения состояния приложения 25](#_Toc197711019)

[2.3 Внешнее хранилище 27](#_Toc197711020)

[ЗАКЛЮЧЕНИЕ 34](#_Toc197711021)

# ВВЕД**ЕНИЕ**

В данной практической работе необходимо реализовать три ключевые задачи, связанные с работой с файловой системой Android-приложения.  
Первая задача направлена на освоение операций создания, записи, чтения и удаления файлов во внутреннем хранилище, включая организацию пользовательского интерфейса и подтверждение удаления через диалоговое окно.

Во второй задаче реализуется сохранение состояния приложения при изменении конфигурации, например, при повороте экрана, с использованием механизмов onSaveInstanceState и onRestoreInstanceState.

Третья задача предполагает работу с внешним хранилищем: создание, запись и удаление файлов в одном приложении, а также чтение этих файлов в другом. Задачи позволяют закрепить понимание работы с внутренним и внешним хранилищем, жизненным циклом активности и взаимодействием с файловой системой Android

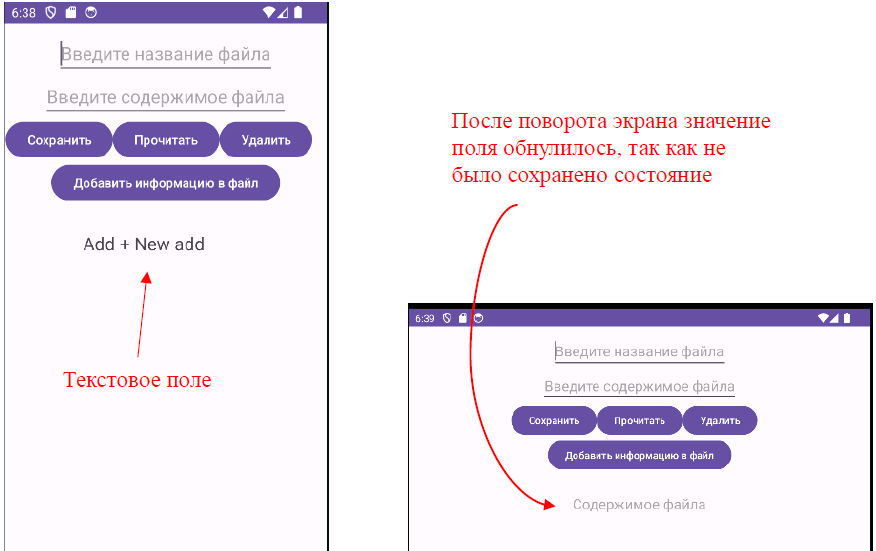
.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 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.

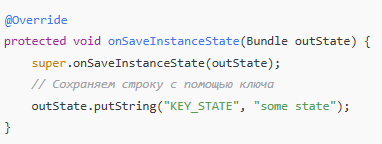
На рисунке 1 показан пример несохраненного состояния.

  
Рисунок 1 – Сброс текстового поля при повороте экрана

Android предоставляет несколько способов управления состоянием приложения, в том числе через методы onSaveInstanceState() и onRestoreInstanceState(). Они используются для сохранения и восстановления данных пользовательского интерфейса, чтобы обеспечить непрерывную работу приложения при изменениях, таких как поворот экрана.

Метод onSaveInstanceState() вызывается системой перед уничтожением активности, чтобы дать возможность сохранить её текущее состояние в объекте Bundle. В этот объект можно помещать данные разных типов — строки, числа, сериализуемые объекты и др.

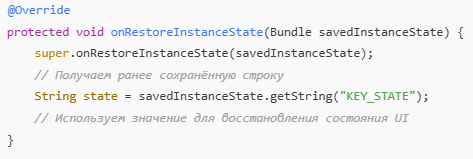
Пример использования onSaveInstanceState() показан на рисунке 2.

  
Рисунок 2 – Пример использования onSaveInstanceState()

Если, например, текстовое поле очищается после смены ориентации экрана, это происходит из-за того, что его значение не было сохранено. Чтобы избежать потери данных, используйте метод putString() объекта Bundle, передавая ключ и значение, которое необходимо сохранить.

Метод onRestoreInstanceState() вызывается после onStart(), если активность была пересоздана. Он принимает Bundle, содержащий ранее сохранённые данные, и позволяет восстановить интерфейс и другие компоненты.

На рисунке 3 показан пример использования onRestoreInstanceState().

  
Рисунок 3 – Пример использования onRestoreInstanceState()

Таким образом, даже при изменении конфигурации, например, при повороте экрана, введённые данные сохраняются и восстанавливаются автоматически.

## Хранение данных в Android

Более надёжным и удобным способом сохранения состояния приложения является его запись в отдельный файл. Такой подход позволяет сохранить данные независимо от смены состояния активности и её жизненного цикла.

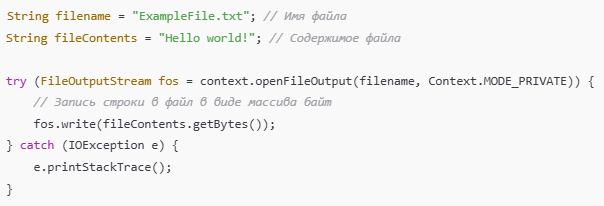
Работа с файловой системой — важная часть разработки Android-приложений. Разработчики часто сталкиваются с задачами, требующими долговременного хранения данных: от сохранения пользовательских настроек и состояний интерфейса до обработки медиафайлов, документов и другой информации, которая должна сохраняться между сессиями. Кроме того, грамотное использование файловой системы позволяет реализовать загрузку и кэширование данных из сети, что способствует повышению производительности и улучшению пользовательского опыта.

Android предоставляет два основных типа хранилища: внутреннее и внешнее. Внутреннее хранилище обеспечивает конфиденциальность — доступ к данным имеет только ваше приложение. Эти файлы автоматически удаляются при деинсталляции. Внешнее хранилище предоставляет больше пространства и позволяет обмениваться файлами между приложениями или даже между устройствами.

## Внутреннее хранилище

Внутреннее хранилище предназначено для хранения данных, к которым должно иметь доступ только само приложение. Эти данные изолированы от других приложений и удаляются автоматически при удалении приложения с устройства.

Для работы с внутренним хранилищем в Android используются специальные методы. Пример записи данных в файл показан на рисунке 4.

  
Рисунок 4 – Запись данных в файл

При записи файлов можно использовать два режима:

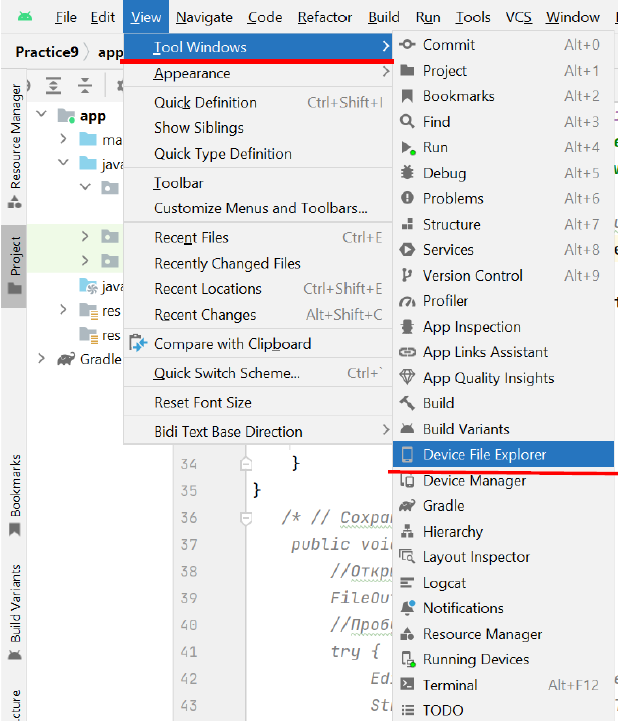
* MODE\_PRIVATE — доступ к файлу есть только у данного приложения. Это режим по умолчанию.
* MODE\_APPEND — новые данные будут добавляться в конец уже существующего файла.

Если файл с указанным именем уже существует, то при использовании MODE\_PRIVATE он будет перезаписан. Чтобы сохранить старые данные и добавить новые, нужно использовать режим MODE\_APPEND.

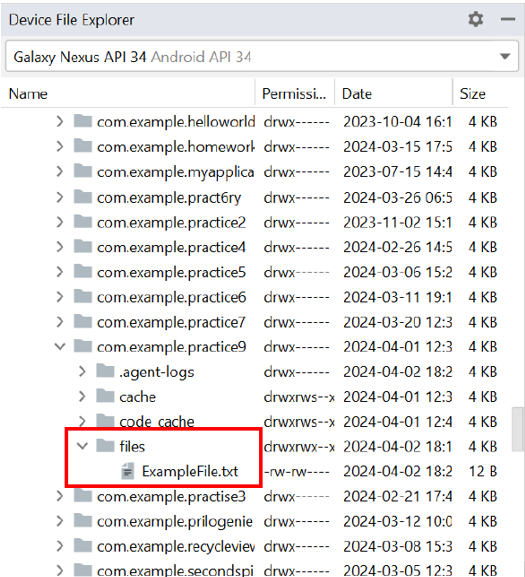
Конструкция try-with-resources обеспечивает автоматическое закрытие потока FileOutputStream после завершения записи, что помогает избежать утечки ресурсов.

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

Где физически находится созданный файл? Чтобы увидеть его на подключенном устройстве перейдем в Android Stud в меню к пункту View, затем Tool Windows и потом Device File Explorer. Это показано на рисунке 5.

  
Рисунок 5 – Поиск созданного файла

На рисунке 6 покажем отображение созданного файла.

  
Рисунок 6 – Отображение созданного файла

И если дважды нажать на файл, то можно увидеть его содержимое.

Для чтения данных из файла, сохранённого во внутреннем хранилище, используется подход, показанный на рисунке 7.

  
Рисунок 7 – Чтение данных из файла

Здесь (рисунок 7) используется FileInputStream, который позволяет читать байты из файла. Для преобразования этих байтов в символы применяется InputStreamReader — он декодирует байты с помощью заданной кодировки (в данном случае UTF-8). Это переход от потоков байтов к потокам символов.

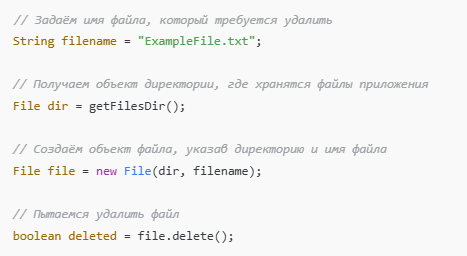
Класс BufferedReader используется для удобного и эффективного чтения текста построчно. Весь прочитанный контент собирается в StringBuilder, после чего преобразуется в строку.

Указание кодировки важно: это может быть явно заданный стандарт (UTF-8, UTF-16 и т. д.), имя кодировки в виде строки или, если не указано, будет использована кодировка по умолчанию для платформы.

Результат работы программы покажем на рисунке 8.

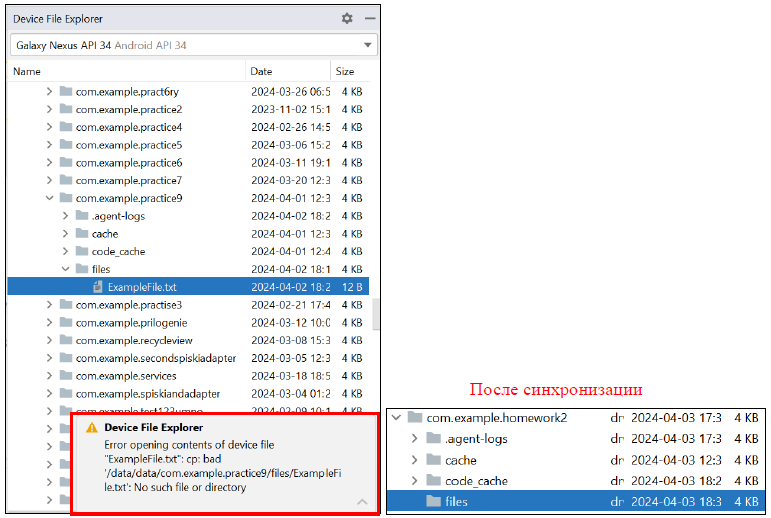
  
Рисунок 8 – Результат работы программы

Удаление файла из внутреннего хранилища показано на рисунке 9.

  
Рисунок 9 – Удаление файла

Если удаление прошло успешно, переменная deleted будет содержать true. После этого доступ к файлу будет невозможен — при попытке его открыть возникнет ошибка.

Чтобы убедиться, что файл действительно удалён, можно обновить содержимое папки files в Android Studio: кликнем правой кнопкой мыши по папке и выберем Synchronize. После синхронизации мы увидим, что файл исчез. Это показано на рисунке 10.

  
Рисунок 10 – Исчезновение файла после синхронизации

## Внешнее хранилище

Ранее мы рассмотрели, как сохранять и читать файлы во внутреннем каталоге приложения. Такие файлы по умолчанию доступны только самому приложению и удаляются вместе с ним.

Если необходимо хранить данные, которые могут быть использованы другими приложениями или должны сохраняться после удаления вашего, следует использовать внешнее хранилище.

Перед тем как записывать файлы во внешнюю память, нужно убедиться, что у приложения есть соответствующие разрешения. Начиная с Android 6.0 (API 23), такие разрешения должны запрашиваться не только в манифесте, но и во время работы приложения. Необходимые разрешения показаны на рисунке 11.

  
Рисунок 11 – Необходимые разрешения

Начиная с Android 10, рекомендуется использовать механизм Scoped Storage, который позволяет безопасно работать с внешними файлами без необходимости запрашивать явные разрешения. Например, доступ к изображениям, видео и другим медиафайлам можно получить через MediaStore API.

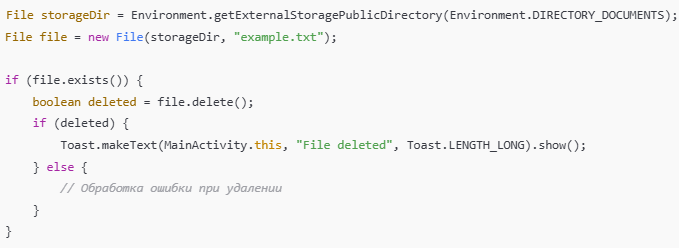
Для создания текстового файла в общей папке "Documents" используется код, показанный на рисунке 12.

  
Рисунок 12 – Создание текстового файла

Для считывания содержимого файла используется код, показанный на рисунке 13.

  
Рисунок 13 – Считывание содержимого файла

Чтобы удалить файл из публичной директории используем код, показанный на рисунке 14.

  
Рисунок 14 – Удаление файла

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Внутреннее хранилище

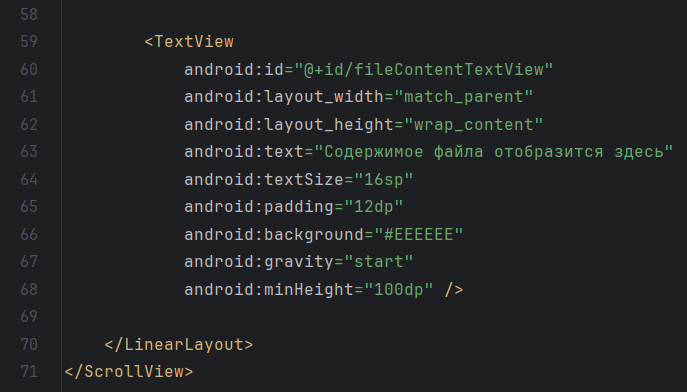
Для реализации данного задания создаем файл разметки activity\_main.xml, где расположены кнопки для сохранения, прочтения, добавления и удаления изменений в файле. Также там есть поле для ввода названия файла, для ввода содержимого файла и для отображения содержимого файла. Первая часть реализации файла activity\_main.xml показаны на рисунке 15.

  
Рисунок 15 – Реализация activity\_main.xml, ч.1

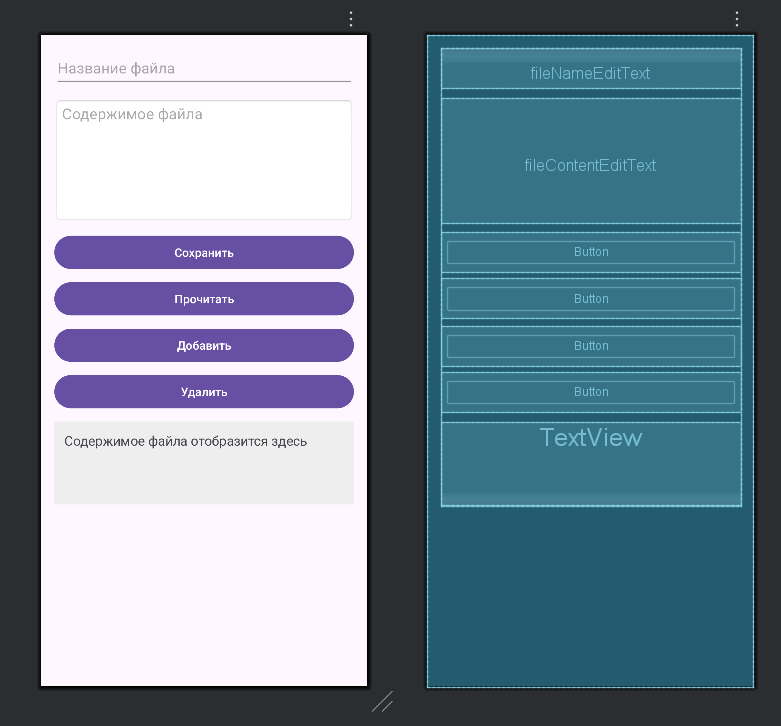
На рисунке 16 показана вторая часть реализации файла разметки.

  
Рисунок 16 – Реализация activity\_main.xml, ч.2

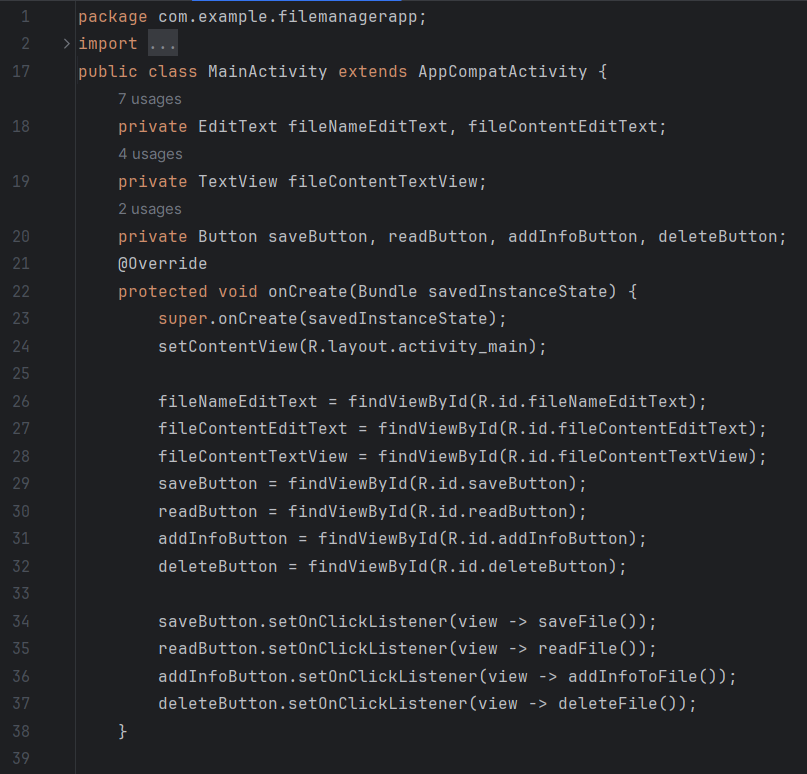
На рисунке 17 показана вторая часть реализации файла разметки.

  
Рисунок 17 – Реализация activity\_main.xml, ч.3

Результат работы activity\_main.xml будет показан на рисунке 18.

  
Рисунок 18 – Результат работы activity\_main.xml

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

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

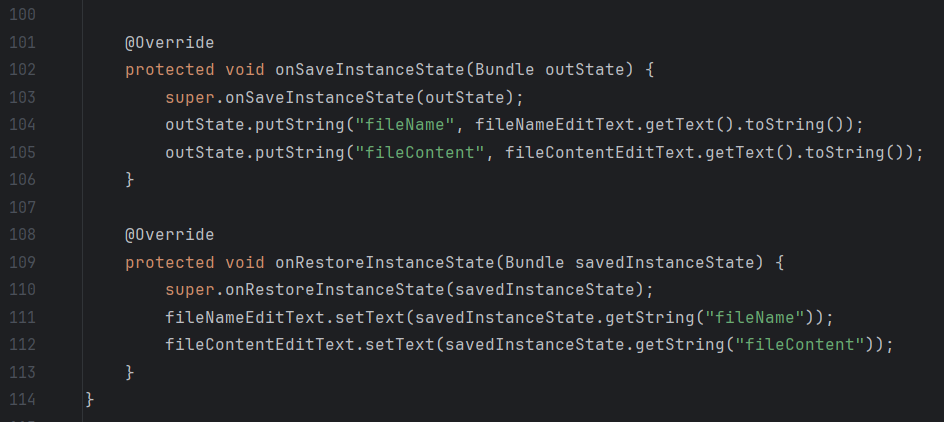
На рисунке 20 покажем вторую часть кода файла MainActivity.java.

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

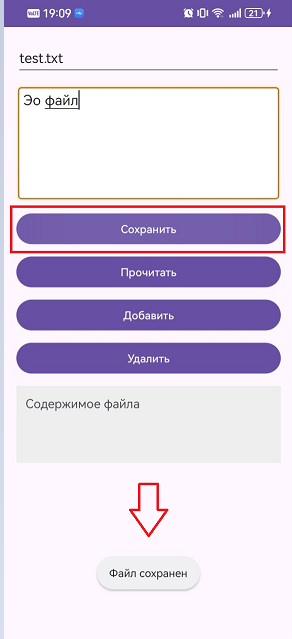
На рисунке 21 покажем третью часть кода файла MainActivity.java.

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

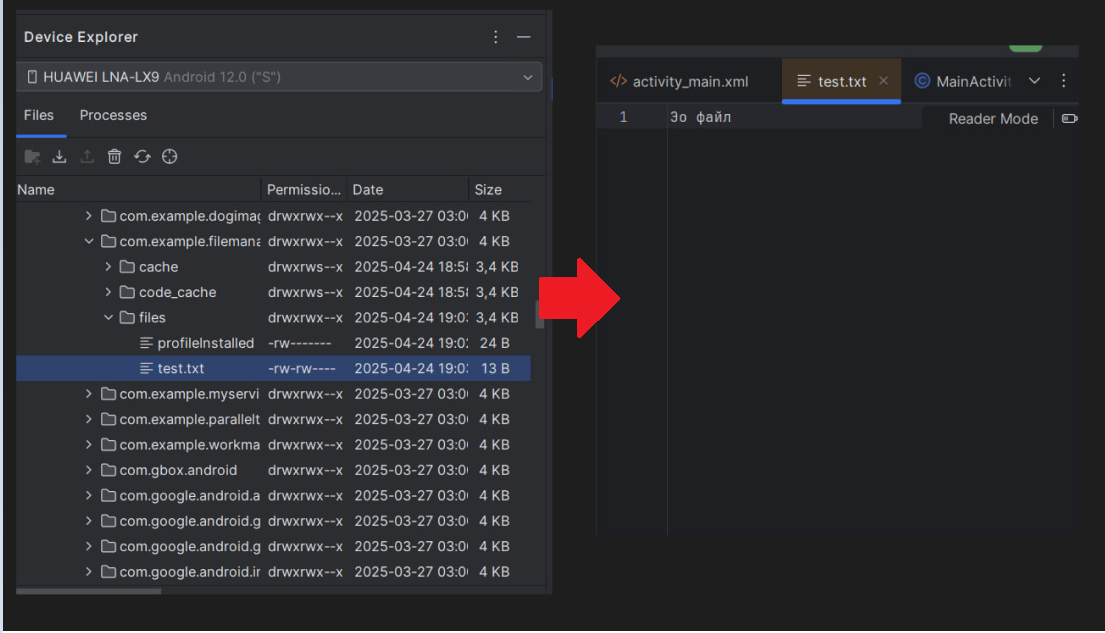
На рисунке 22 покажем четвертую часть кода файла MainActivity.java.

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

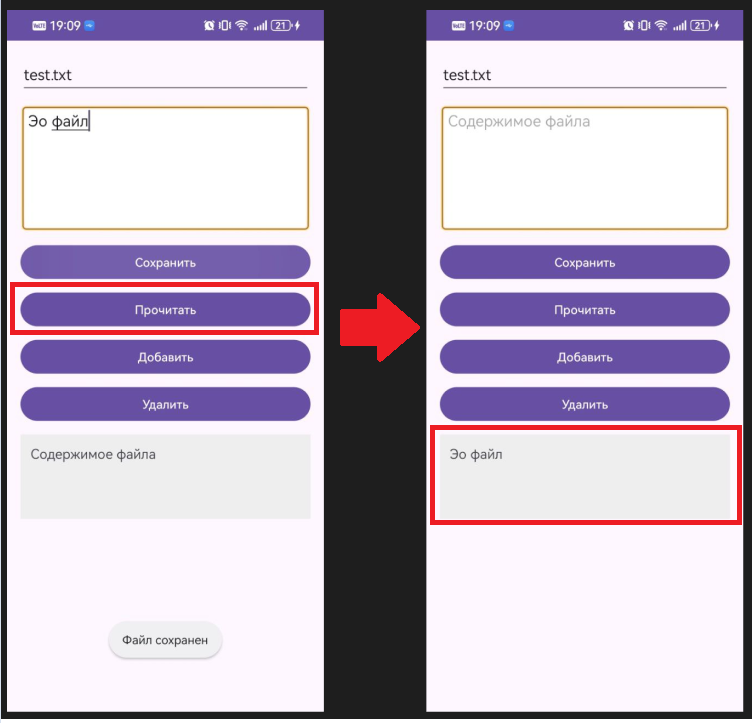
На рисунке 23 покажем результат запуска проекта. Сразу впишем все необходимое в поля. На рисунке видно, что файл соханился.

  
Рисунок 23 – Результат запуска проекта с сохранением файла

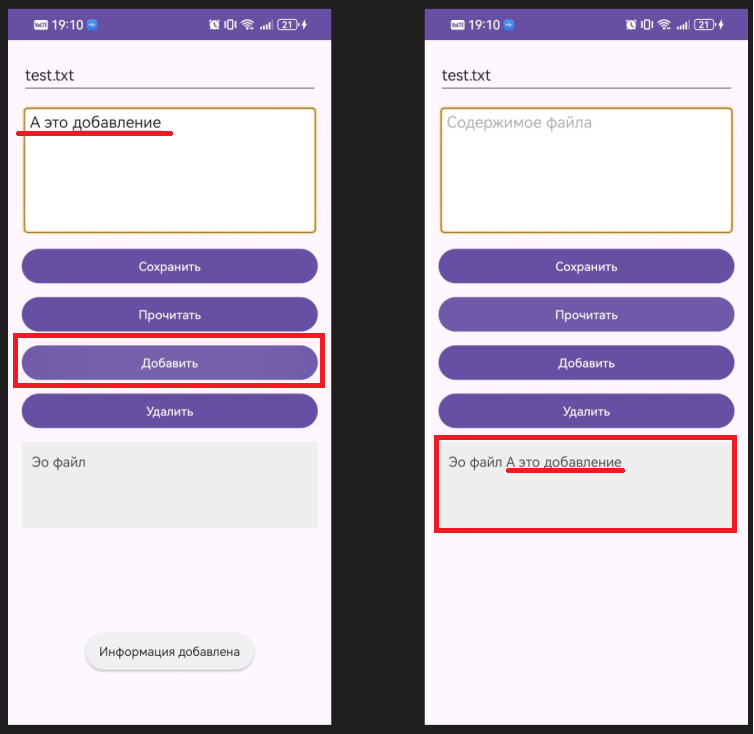
Чтобы посмотреть сохранился ли файл мы можем это посмотреть через AndroidStudio, что показано на рисунке 24.

  
Рисунок 24 – Демонстрация сохраненного файла в AndroidStudio

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 процесс удаления файла.

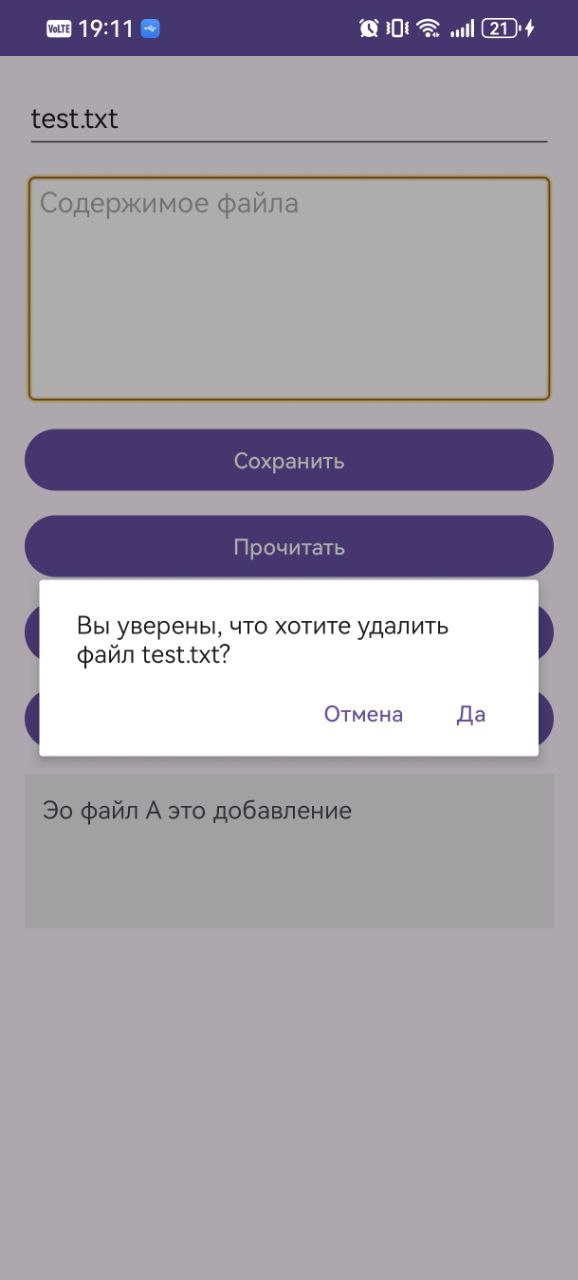
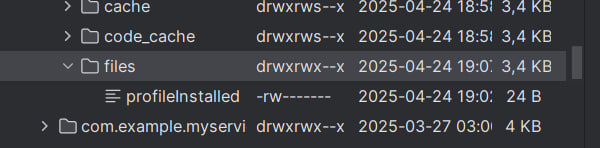


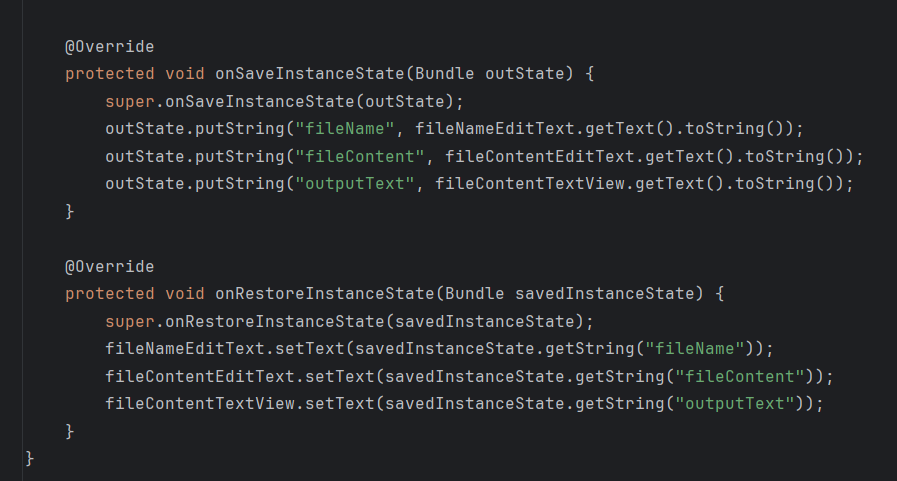
Рисунок 27 – Процесс удаления файла

Чтобы проверить удалился ли файл на самом деле мы воспользуемся AndroidStudio. Как показано на рисунке 26, файл действительно удалился.

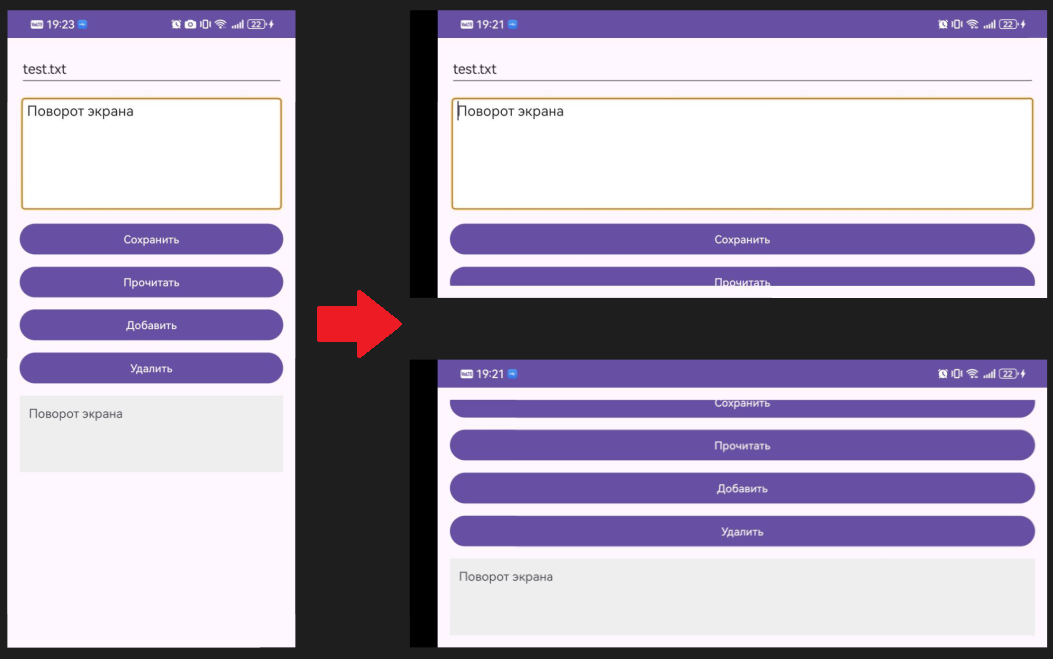
  
Рисунок 28 – Демонстрация, что файл удалился

## Сохранения состояния приложения

Для сохранения информации в горизонтальном положении нам не нужно менять файл разметки. Нам достаточно внести небольшое изменения файл MainActivity.java. На рисунке 26 показан блок, который мы немного изменили в файл.

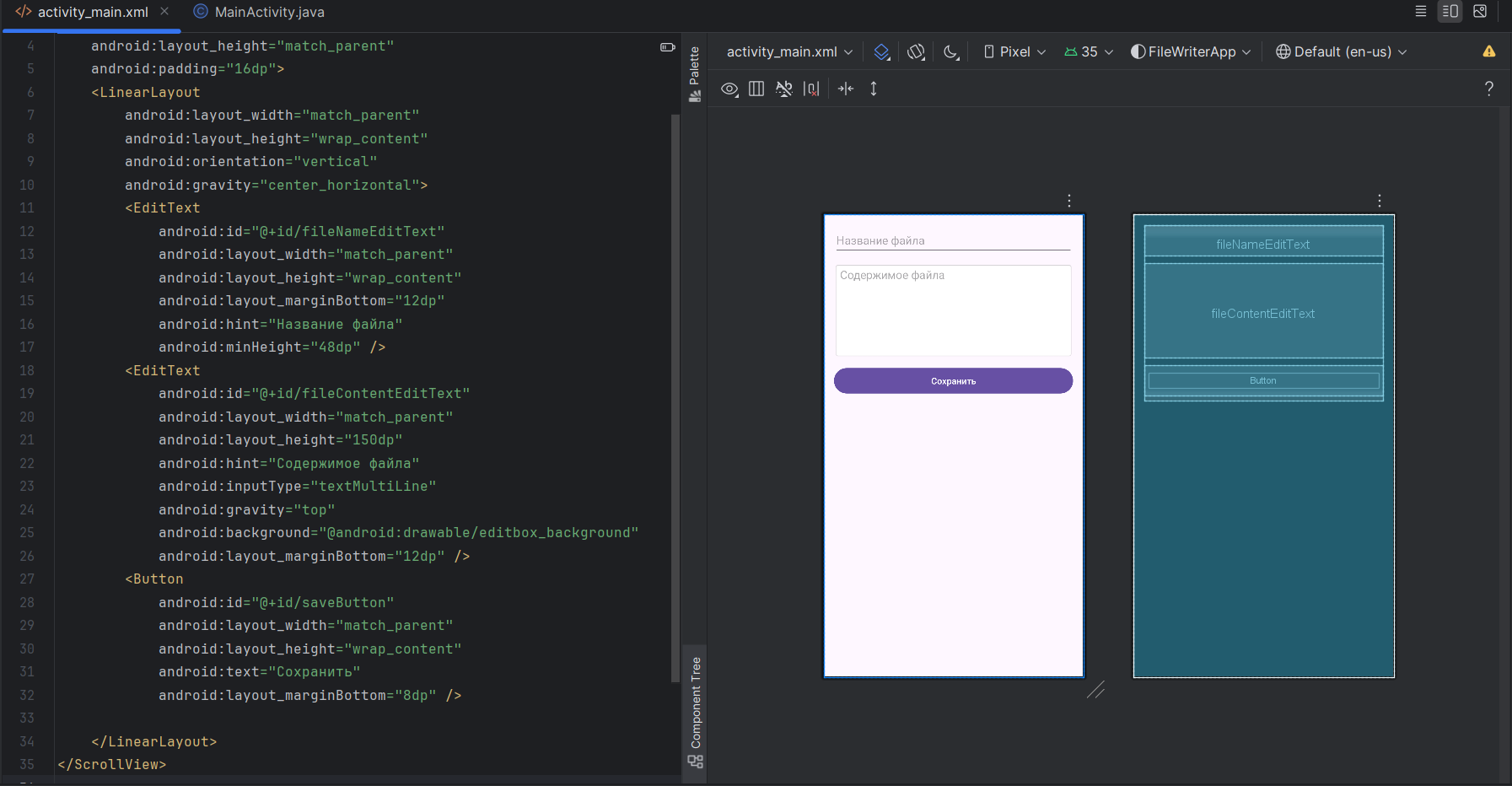
  
Рисунок 28 – Измененная часть файла MainActivity.java

На рисунке 29 продемонстрируем, что при повороте экрана все сохраняется.

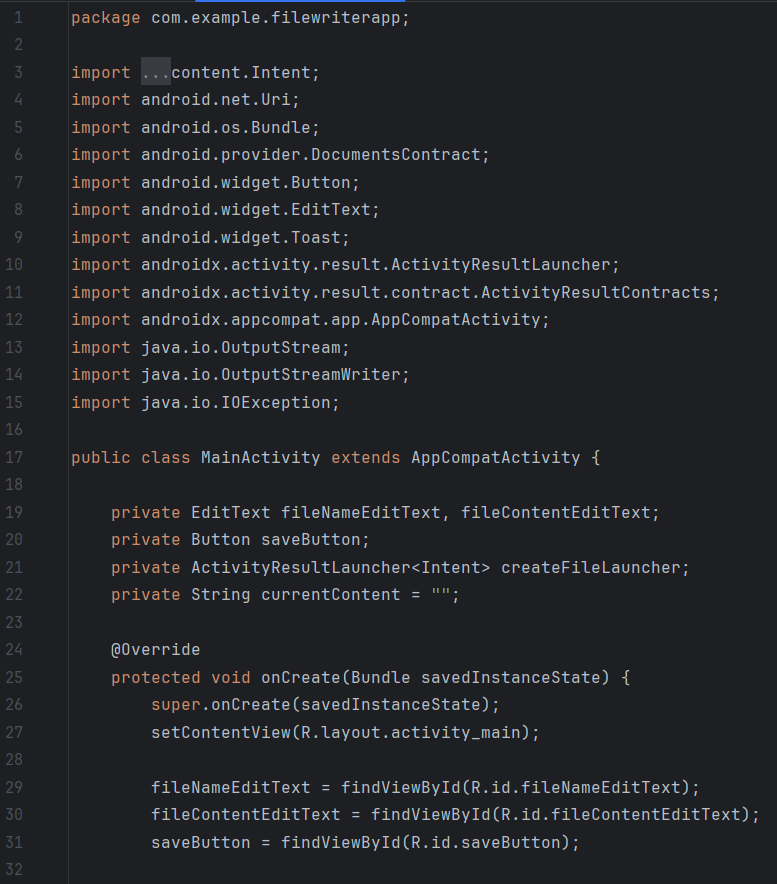
  
Рисунок 29 – Демонстрация поворота экрана

## 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 FileWriterApp. Другое для чтения – FileReaderApp. На рисунке 30 покажем реализацию и результат работы файла разметки activity\_main.xml из приложения FileWriterApp.

  
Рисунок 30 – Реализация и результат работы файла разметки activity\_main.xml

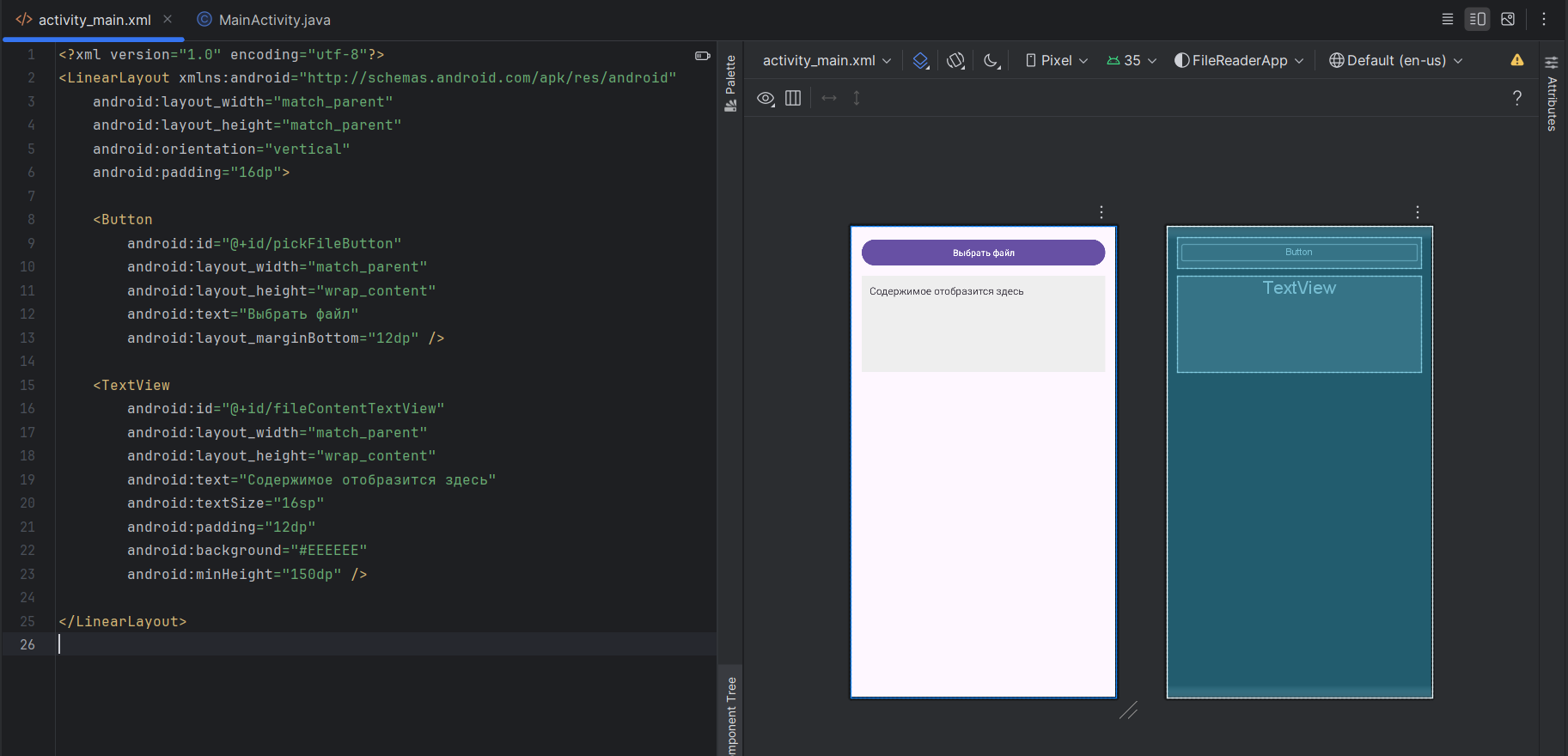
На рисунке 31 покажем первую часть кода файла MainActivity.java из FileWriterApp.

  
Рисунок 31 – Код файла MainActivity.java, ч.1

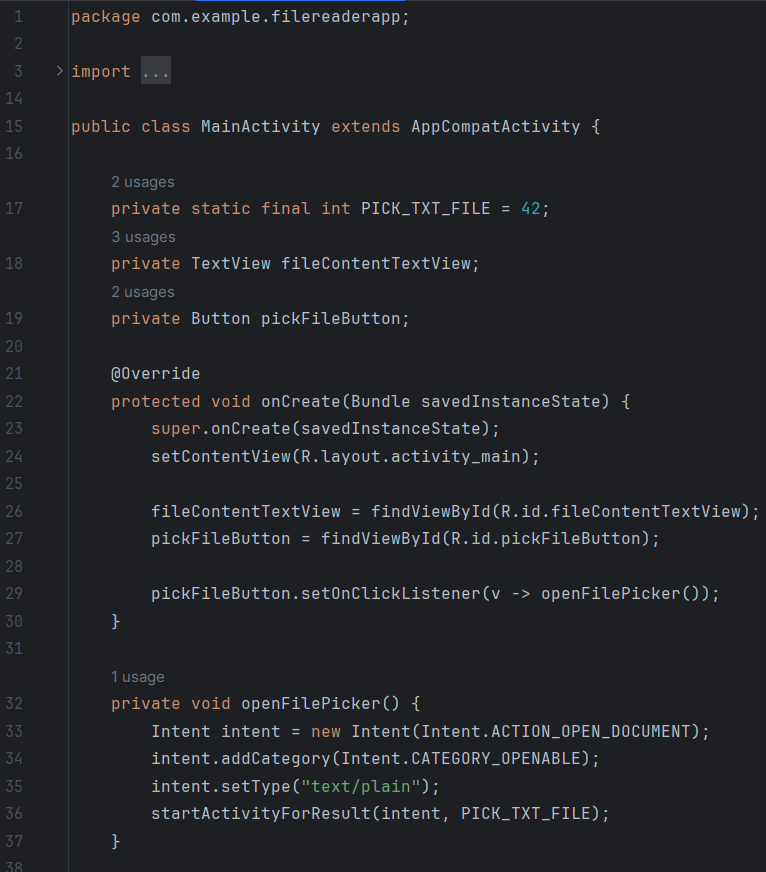
На рисунке 32 покажем вторую часть кода файла MainActivity.java из FileWriterApp.

  
Рисунок 32 – Код файла MainActivity.java, ч.2

На рисунке 33 покажем реализацию и результат работы файла разметки activity\_main.xml из приложения FileReaderApp.

  
Рисунок 33 – Реализация и результат работы файла разметки activity\_main.xml

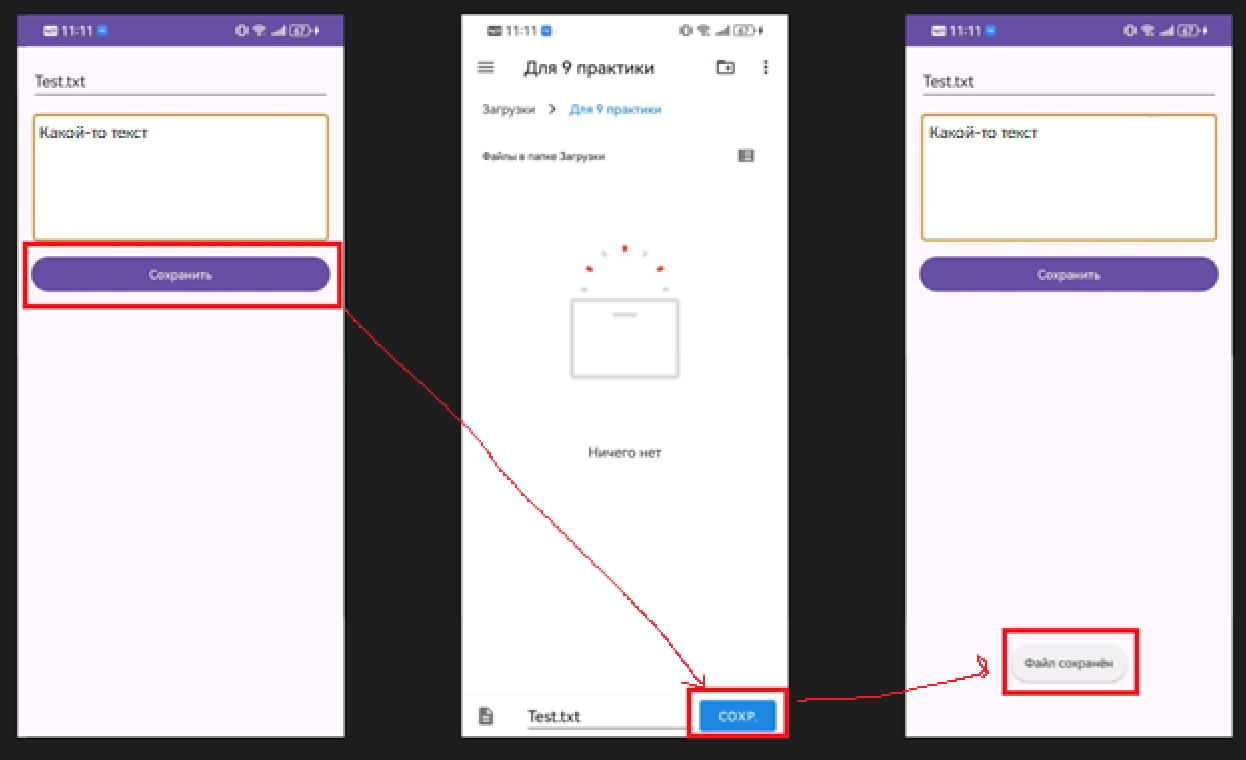
На рисунке 34 покажем первую часть кода файла MainActivity.java из FileReaderApp.

  
Рисунок 34 – Код файла MainActivity.java, ч.1

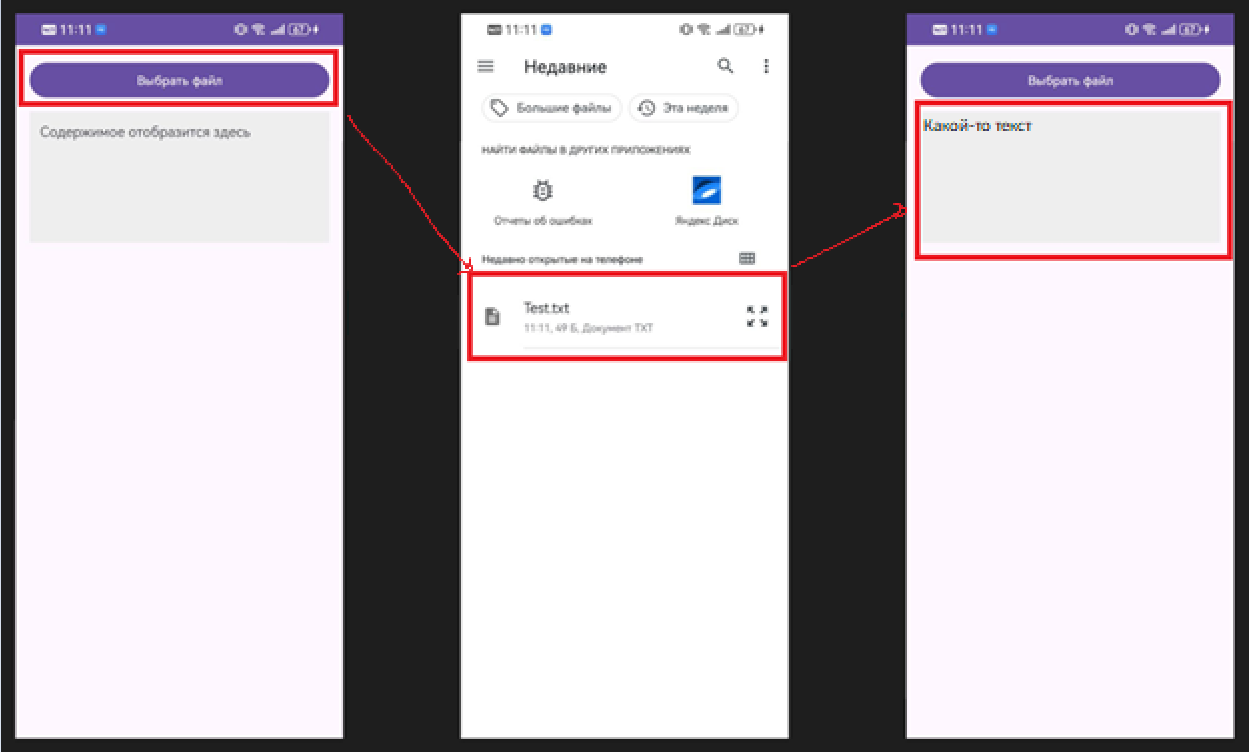
На рисунке 35 покажем вторую часть кода файла MainActivity.java из FileReaderApp.

  
Рисунок 35 – Код файла MainActivity.java, ч.2

На рисунке 36 покажем демонстрацию работы проекта FileWriterApp.

  
Рисунок 36 – Демонстрация работы проекта FileWriterApp

На рисунке 37 покажем демонстрацию работы проекта FileReaderApp.

  
Рисунок 37 – Демонстрация работы проекта FileReaderApp

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы были реализованы основные операции с файлами во внутреннем и внешнем хранилище Android-устройств.  
Реализовано создание и редактирование файлов с использованием FileOutputStream и BufferedReader, а также вывод результатов в пользовательский интерфейс. При повороте экрана успешно сохраняется состояние ввода и отображения, что позволяет пользователю продолжить работу без потери данных. Также была организована работа с внешним хранилищем, что обеспечило возможность чтения созданных файлов в другом приложении. Полученные навыки являются основополагающими для создания Android-приложений с устойчивым поведением и функциональностью, связанной с хранением и передачей данных.