



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

КУРСОВАЯ РАБОТА

по дисциплине «Разработка мобильных приложений»

**Тема курсовой работы «Информационно-справочная система "Жесткие
диски"»**

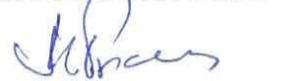
Студент группы ИКБО-03-22

Борзов Дмитрий Олегович


(подпись студента)

Руководитель курсовой работы

доцент Рысин М.Л.


(подпись руководителя)

Работа представлена к защите

«25» мая 2024 г.

Допущен к защите

«25» мая 2024 г.


Москва 2024



МИНОБРНАУКИ РОССИИ

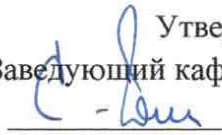
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю
Заведующий кафедрой МОСИТ
 Головин С.А.
«19» февраля 2024 г.

ЗАДАНИЕ
на выполнение курсовой работы по дисциплине
«Разработка мобильных приложений»

Студент Борзов Дмитрий Олегович

Группа ИКБО-03-22

Тема работы: «Информационно-справочная система "Жесткие диски"»

Исходные данные:

Разрабатываемый прототип должен предоставлять возможности полностью интерактивной системы с понятным дружественным UI и обеспечивать необходимую функциональность, которая формируется в зависимости от заданной темы и предметной области изучаемых вопросов.


Перечень вопросов, подлежащих разработке, и обязательного графического материала:

Установка и настройка Android с применением виртуальных сред. Установка и настройка IDE (Android Studio, Eclipse, IntelliJ IDEA, Kivi и т.п.). Установка и настройка эмуляторов Android. Анализ предметной области разрабатываемой программной системы, сбор и анализ требований, составление ТЗ согласно ГОСТ 19.201-78.

Изучение жизненного цикла программ и создание мобильного программного комплекса с применением языка программирования Java, согласно теме курсовой работы. Реализация в создаваемом программном комплексе хранения данных в виде БД, файлов или пар ключ-значение. Обеспечение их создания, чтения, записи во внутреннем и внешнем хранилище. Обеспечение безопасности информации при работе планового программного комплекса (обеспечение работы ролевой модели безопасности). Тестирование и отладка кода созданного программного продукта, контрольные примеры работы. Возможно портирование написанной программной системы на внешних хостах в сети Интернет. Отчет по курсовой работе в виде пояснительной записки.


Срок представления к защите курсовой работы:

Задание на курсовую работу выдал

 до «25» мая 2024 г.
доцент Рысин М.Л.

«19» февраля 2024 г.

Задание на курсовую работу получил

 (Борзов Д.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	5
2 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ ПРИЛОЖЕНИЯ	7
3 ТЕХНИЧЕСКОЕ ЗАДАНИЕ.....	8
4 ОСНОВНАЯ ЧАСТЬ.....	10
4.1 АРХИТЕКТУРА ПРОГРАММНОЙ СРЕДЫ	10
4.2 РЕАЛИЗАЦИЯ КОДА ПРОГРАММЫ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ JAVA	13
4.2.1 СТРУКТУРА ОБЪЕКТОВ	13
4.2.2 ЗАВИСИМОСТИ ПРИЛОЖЕНИЯ, РАЗРЕШЕНИЯ И СПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ.....	14
4.2.3 РЕАЛИЗОВАННЫЙ ФУНКЦИОНАЛ	14
4.2.4 РАЗЛИЧНЫЕ УЛУЧШЕНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА	15
4.3 ХРАНЕНИЕ ДАННЫХ В ПРОГРАММНОМ КОМПЛЕКСЕ.....	17
4.4 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ИНФОРМАЦИИ ПРИ РАБОТЕ ПЛАНОВОГО ПРОГРАММНОГО КОМПЛЕКСА.....	19
4.5 СИСТЕМА КОНТРОЛЯ ВЕРСИЙ.....	21
5 ПОЛЬЗОВАТЕЛЬСКИЙ СЦЕНАРИЙ И WIREFRAME ПРИЛОЖЕНИЯ ..	22
6 КОНТРОЛЬНЫЙ ПРИМЕР РАБОТЫ ПРИЛОЖЕНИЯ.....	25
7 ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ.....	29
ЗАКЛЮЧЕНИЕ	30
СПИСОК ЛИТЕРАТУРЫ.....	31
ПРИЛОЖЕНИЕ	32

ВВЕДЕНИЕ

В современном мире поток информации постоянно растет, и доступ к ней становится все более важным и актуальным. Одним из важных источников информации являются жесткие диски, которые хранят и обрабатывают огромные объемы данных в компьютерных системах. В этом контексте разработка информационно-справочной системы "Жесткие Диски" становится весьма значимой.

Цель курсовой работы: разработать мобильное приложение – информационно-справочную систему “Жёсткие диски” средствами языка Java на платформе Android, предоставляющего пользователям удобный доступ к информации о жестких дисках, их характеристиках, работе и обслуживании.

Разработка мобильного приложения "Жесткие Диски" на языке Java позволит пользователям получать необходимую информацию о жестких дисках непосредственно на своих мобильных устройствах.

Задачи курсовой работы:

- 1) проанализировать предметную область мобильного приложения;
- 2) сделать обзор существующих аналогов разрабатываемого приложения;
- 3) сформировать техническое задание на разработку программы в соответствии с ГОСТ 19.201-78;
- 4) описать архитектуру программной системы, привести структурную и функциональную диаграммы, схему базы данных;
- 5) спроектировать интерфейс мобильного приложения;
- 6) реализовать код программы на языке высокого уровня Java, протестировать его и отладить;
- 7) реализовать контрольный пример работы программы, начиная с открытия, показать все этапы работы вашего приложения.

Структура пояснительной записки: введение, 7 пунктов, заключение, список из 9 используемых источников, 50 приложений.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В мире технологий хранения данных жесткие диски (ЖД) играют ключевую роль. Эти устройства предоставляют надежное и долговечное хранение информации для широкого спектра устройств - от персональных компьютеров до крупных серверных систем. Постоянные улучшения в технологиях производства и инновации в конструкции приводят к появлению новых возможностей и улучшению характеристик жестких дисков.

Однако с развитием технологий хранения данных возникают и новые проблемы. Безопасность данных становится все более актуальной проблемой в свете угроз кибербезопасности. Шифрование данных, методы защиты от несанкционированного доступа и надежные методы удаления информации становятся неотъемлемой частью разработки и использования жестких дисков.

Важным аспектом также является управление данными на жестких дисках. Оптимизация производительности, резервное копирование, восстановление данных и фрагментация диска — все это важные аспекты эффективного управления информацией на уровне жесткого диска.

Приложение предоставляет информацию о различных моделях жестких дисков, их характеристиках, производителях. Пользователи могут просматривать детальную информацию о каждом жестком диске, сравнивать различные модели и просматривать отзывы других пользователей.

Для информационно-справочной системы создается база данных «Жёсткие диски», которая содержит следующие данные:

- Сведения о различных моделях жестких дисков (наименование, краткое описание с необходимой информацией, полное описание).
- Отзывы пользователей о различных моделях жестких дисков.
- Данные пользователей (Имя, почта, пароль, ссылка на изображение пользователя).

- Изображения дисков.
- Изображения пользователей.

С базой данных могут работать администраторы, для которых доступны следующие задачи:

- Добавление информации о новых дисках в систему.
- Удаление не актуальной информации о дисках.
- Удаление нерелевантных отзывов пользователей.
- Переключение статуса администратора другим пользователям.

В курсовой работе информационная модель данных предметной области объявляется с помощью ER-диаграммы (Entity-Relationship Diagram). ER-диаграмма представляет собой графическое отображение структуры данных, включающее сущности (объекты), их атрибуты и связи между ними на рисунке в приложении к курсовой работе №2.

В данной предметной области, бизнес-процессы, в которых будет задействована программа, включают управление информацией о жестких дисках, добавление новой информации, изучение и анализ существующей информации, а также обновление информации. Эти процессы взаимосвязаны и вместе формируют цикл управления информацией в системе. Детальное представление этих процессов можно увидеть на представленной диаграмме IDEF0 в приложении к курсовой работе №3-4.

Вывод: Информационно-справочные системы, такие как "Жесткие диски", представляют собой необходимый инструмент в современном мире, обеспечивая доступ к широкому спектру знаний и данных. Они упрощают процесс получения информации, повышают эффективность работы и способствуют принятию обоснованных решений.

2 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ ПРИЛОЖЕНИЯ

На рынке Android-приложений нет прямых аналогов приложения. Существующие аналоги в виде маркетплейсов, таких как Яндекс.Маркет или DNS, не являются специализированными для данной области и обладают рядом недостатков.

Пользователи отмечают, что в маркетплейсах не всегда удобно ориентироваться из-за избытка информации и разнообразия товаров. Также нередко возникают проблемы с навигацией, фильтрацией и сравнением товаров. Отсутствие персонализированных рекомендаций и ограниченные возможности поиска также являются серьезными недостатками существующих платформ.

В целом, отзывы пользователей указывают на необходимость специализированного приложения, которое бы предлагало удобный интерфейс, интуитивно понятную навигацию и надежное взаимодействие с приложением даже без доступа в интернет, что не предоставляют маркетплейсы. Такое приложение могло бы значительно улучшить опыт поиска необходимой информации о жестких дисках.

Вывод: Существующие аналоги не удовлетворяют полностью потребности пользователей из-за недостатков в навигации, поиске и персонализации. Разработка специализированного приложения имеет смысл для улучшения опыта пользователей и обеспечения им более удобного доступа к необходимой информации и продуктам.

3 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

С полным техническим заданием можно ознакомиться в приложении №1 к курсовой работе.

4. Требования к программе или программному изделию

4.1. Требования к функциональным характеристикам:

4.1.2 Возможность авторизации пользователя, которая включает в себя процесс входа и регистрации.

4.1.3 Предоставление общего описания жёстких дисков.

4.1.4 Предоставление удобного списка жёстких дисков.

4.1.5 Предоставление обновляющегося информационного списка с ресурса IXBT [6].

4.1.6 Возможность добавление информации о дисках администратором.

4.1.7 Возможность оставить отзыв конкретному диску.

4.2. Требования к надежности:

4.2.1 Приложение должно безотказно предоставлять доступ к полному функционалу.

4.2.2 Приложение должно предусматривать защиту от несанкционированного доступа к данным, например, ролевою модель с использованием механизмов аутентификации и авторизации.

4.2.3 Приложение должно быть устойчивым к внешним атакам, таким как атаки на основе внедрения вредоносного кода или перехвата данных.

4.2.4 Приложение должно иметь механизм контроля целостности данных, чтобы обнаруживать и предотвращать их изменение или подделку.

4.2.5 Приложение должно обеспечивать конфиденциальность данных пользователей путем шифрования чувствительной информации, передаваемой по сети или хранящейся на устройстве.

4.4. Требования к составу и параметрам технических средств:

Приложение должно быть доступно для использования на мобильных устройствах с операционной системой Android. Минимальные требования к устройствам для корректной работы приложения:

1. Процессор: Для Android - Snapdragon 660 или эквивалентный;
2. Оперативная память: Минимум 2 ГБ.
3. Свободное место: Не менее 100 МБ.
4. Разрешение экрана: Минимум 720x1280 пикселей.
5. Версия операционной системы: Android 7.0 (Nougat) и выше.

Эти требования установлены на основе минимальных аппаратных возможностей, необходимых для поддержания производительности, заданных требованиями разработчика инфо.

4.5. Требования к маркировке и упаковке:

Приложение должно обладать адаптивной иконкой.

5. Техничко-экономические показатели:

Ориентировочная экономическая эффективность разработки приложения оценивается как положительная, учитывая следующие факторы:

1. Предполагаемая годовая потребность: Предполагается, что приложение будет активно использоваться пользователями, что приведет к стабильному и постоянному спросу на продукт.

2. Экономические преимущества по сравнению с аналогами: Приложение предлагает современные и удобные функции, что делает его более привлекательным для пользователей, чем аналогичные приложения на рынке. Кроме того, его разработка предполагается более эффективной и экономичной благодаря использованию современных технологий и методов разработки.

4 ОСНОВНАЯ ЧАСТЬ

4.1 АРХИТЕКТУРА ПРОГРАММНОЙ СРЕДЫ

Общее описание архитектуры программной системы включает следующие основные аспекты:

1. Модульная структура: Система организована на основе модулей, каждый из которых отвечает за определенную функциональность или компонент приложения.

2. Клиент-серверная архитектура: В случае мобильного приложения, клиентская часть взаимодействует с сервером для получения данных, обновлений и других ресурсов. Это позволяет обеспечить масштабируемость, удобство обновлений и централизованное управление данными.

3. Пользовательский интерфейс: Архитектура включает в себя компоненты, отвечающие за отображение пользовательского интерфейса, обработку пользовательских действий и взаимодействие с бэкендом.

4. Бэкенд: это часть системы, отвечающая за обработку бизнес-логики, работу с базой данных, аутентификацию и авторизацию пользователей, обработку запросов от клиентской части приложения.

5. База данных: Система использует NoSQL (Not Only SQL) базу данных Firebase для хранения и управления данными, в зависимости от требований и особенностей приложения.

6. Коммуникация: Механизмы коммуникации между компонентами системы, такие как API, сетевые протоколы, сообщения и т.д., обеспечивают эффективное взаимодействие и передачу данных.

7. Безопасность: Архитектура включает в себя механизмы защиты данных, аутентификации и авторизации пользователей, обработку ошибок и предотвращение уязвимостей.

8. Масштабируемость и производительность: Система спроектирована с учетом возможности масштабирования и обеспечения высокой производительности при работе с большим объемом данных и пользовательским трафиком. Система также спроектирована с возможностью расширения функционала.

Процесс проектирования программного обеспечения включает в себя определение структурных компонентов программной системы и связей между ними [2]. Результат уточнения структуры может быть представлен в виде структурной схемы, которая дает достаточно полное представление о проектируемом программном обеспечении. На рис. 1 приведена структурная схема программного обеспечения информационно-справочной системы «Жесткие диски».



Рисунок 1. – Структурная схема программной системы.

Функциональные схемы более информативны, чем структурные [2]. На рис. 2 приведена функциональная схема программной системы, реализующей операцию получения информации о жестких дисках.

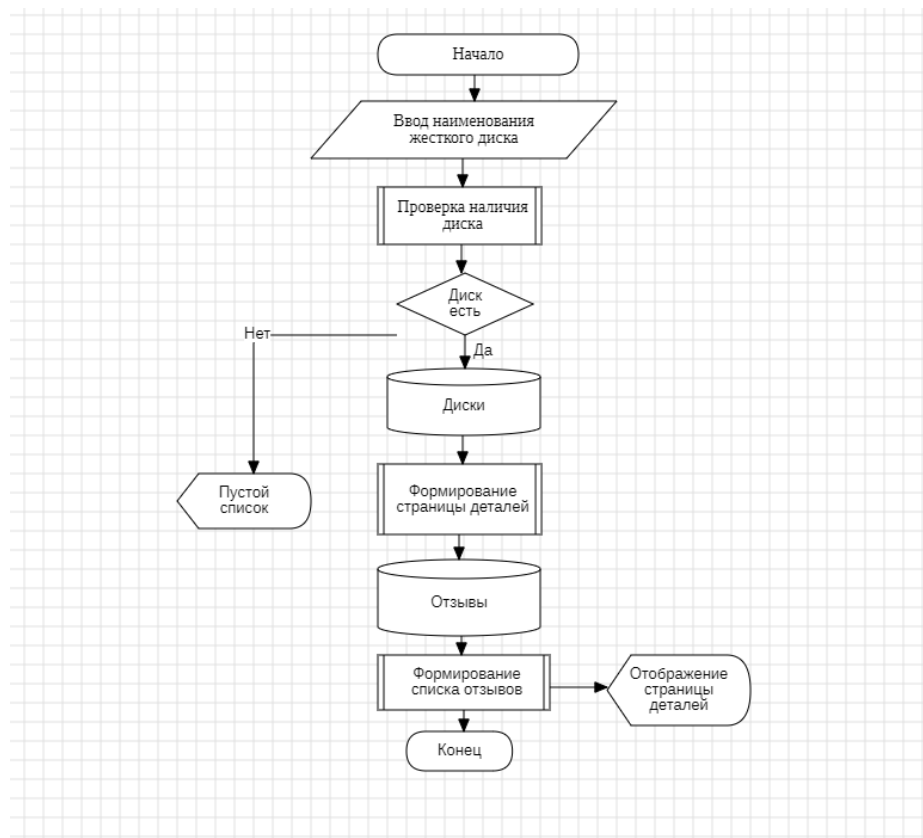


Рисунок 2. – Функциональная схема программной системы.

Схема базы данных предоставляет полное представление о структуре данных и их взаимосвязях в базе данных, что позволяет проектировщикам и разработчикам лучше понять и организовать хранение и доступ к данным. На рисунке - 3 приведена схема базы данных для информационной системы "Жесткие диски. Схема представлена в нотации IDEF1X [4]. Физический уровень модели доступен в приложении к курсовой работе №5.

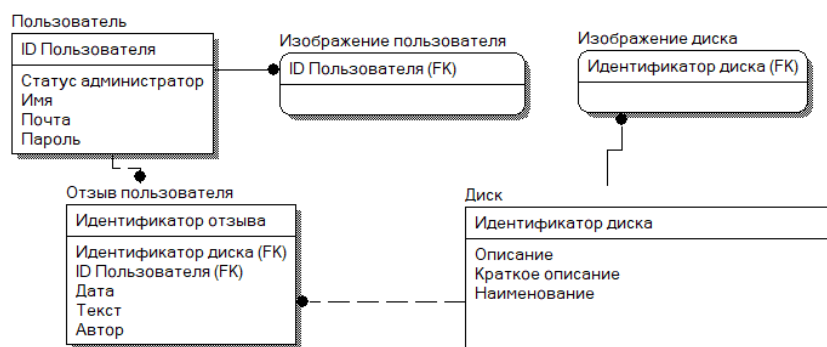


Рисунок 3. – Логический уровень.

4.2 РЕАЛИЗАЦИЯ КОДА ПРОГРАММЫ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ JAVA

Java – это объектно-ориентированный язык программирования, реализующий, к тому же, и структурную, и обобщенную, и функциональную методологии (мультипарадигменность). Подробнее о Java можно прочитать на сайте правообладателя [8], которым в настоящее время является компания Oracle.

Технология Java включает в себя стандартизованный язык программирования и программную реализацию – платформу [3].

Платформы Java представлены в нескольких вариантах, например:

- Standard Edition (SE) – для локальных и небольших сетевых приложений;
- Enterprise Edition (EE) – для создания сетевых приложений уровня предприятия.

4.2.1 СТРУКТУРА ОБЪЕКТОВ

Архитектура приложения включает следующие объекты (рис. 15):

1. MainActivity: Это основная активность приложения, которая включает в себя четыре фрагмента: HomeFragment, DiskListFragment, ProfileFragment и AboutFragment.

2. HomeFragment, DiskListFragment, ProfileFragment: Это основные фрагменты, которые пользователь видит при взаимодействии с приложением.

3. AboutFragment: Этот фрагмент содержит два вложенных фрагмента: AppAboutFragment и DeveloperAboutFragment, которые предоставляют информацию о приложении и разработчике соответственно.

4. RegistrationActivity, LoginActivity, DiskDetailsActivity, DiskAddActivity: Это отдельные активности, которые обрабатывают

регистрацию пользователя, вход в систему, просмотр деталей диска и добавление диска соответственно.

В приложении к курсовой работе №8 приведена диаграмма классов, иллюстрирующая структуру и взаимосвязи между основными компонентами информационно-справочной системы, предназначенной для предоставления информации о жестких дисках.

Эта архитектура представляет собой типичную модель для Android-приложений, где есть одна основная активность, содержащая несколько фрагментов для различных частей пользовательского интерфейса, а также отдельные активности для обработки специфических задач. Это обеспечивает хорошую модульность и повторное использование кода.

4.2.2 ЗАВИСИМОСТИ ПРИЛОЖЕНИЯ, РАЗРЕШЕНИЯ И СПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ

Gradle в рамках Android Studio является мощным инструментом сборки, который управляет процессом компиляции, сборки и развертывания Android-приложений. Он используется для автоматизации задач сборки и предоставляет гибкую систему управления зависимостями и конфигурацией проекта. Также в приложении были использованы современных методы решения задач, подробно изложено в приложении к курсовой работе №6.

4.2.3 РЕАЛИЗОВАННЫЙ ФУНКЦИОНАЛ

В результате было получено: функциональность новостной ленты, загружающей данные с RSS-канала с использованием библиотеки OkHttp и XmlPullParser для обработки XML-документов. Также было реализовано взаимодействие с камерой и галереей для выбора и создания изображений, их обработка и преобразование, а также загрузка изображений в облачное хранилище Firebase. Кроме того, был создан список дисков и отзывов с собственным адаптером, который связывает данные с пользовательским интерфейсом. Все эти функции были тщательно протестированы и отлажены

для обеспечения корректной работы. Подробное описание реализации доступно в приложении к курсовой работе №2.

4.2.4 РАЗЛИЧНЫЕ УЛУЧШЕНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА АНИМАЦИИ

В приложении используются различные типы анимаций для улучшения пользовательского интерфейса и обеспечения плавного взаимодействия. Рассмотрим подробнее два примера анимаций: анимация пролистывания дисков и анимация перехода между фрагментами.

Анимация пролистывания дисков

Для создания эффекта анимации пролистывания дисков используется следующая конструкция – листинг 9:

Листинг 9. DiskAdapter.java

```
137 // Применяем анимацию к convertView
138 Animation animation = AnimationUtils.loadAnimation(getContext(), R.anim.translate);
139 convertView.startAnimation(animation);
```

Анимация перехода между фрагментами

Для создания плавного перехода между фрагментами используется следующий код – листинг 10:

Листинг 10. MainActivity.java

```
102 getSupportFragmentManager() FragmentManager
103     .beginTransaction() FragmentTransaction
104     .setCustomAnimations(R.anim.slide_in, R.anim.slide_out) // Добавляем анимацию
105     .replace(R.id.fragmentView, fragment)
106     .setReorderingAllowed(true)
107     .addToBackStack( name: "backstack")
108     .commit();
```

Вывод

Использование анимаций в приложении улучшает пользовательский интерфейс, делая взаимодействие с приложением более плавным и интуитивно понятным. Анимация пролистывания дисков и перехода между фрагментами создаёт визуально приятные эффекты, которые повышают общее впечатление от работы с приложением.

ВОЗМОЖНОСТЬ СМЕНЫ ТЕМЫ ПРИЛОЖЕНИЯ

В приложении реализована возможность смены темы пользователем, что позволяет адаптировать внешний вид интерфейса в соответствии с личными предпочтениями. Рассмотрим, как это реализовано на примере фрагмента кода.

Установка темы пользователем

В зависимости от выбранной темы, приложение устанавливает соответствующий стиль и фоновое изображение. Это достигается через условный оператор switch, который проверяет название темы и применяет соответствующие ресурсы. Реализация представлена на листинге 11.

Листинг 10. MainActivity.java

```
58 // Устанавливаем тему
59 switch (themeName) {
60     case "Base.Theme.DiskWizard.Green":
61         setTheme(R.style.Base_Theme_DiskWizard_Green);
62         backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
63         break;
64     case "Base.Theme.DiskWizard.DeepPurple":
65         setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);
66         backGroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
67         break;
68     case "Base.Theme.DiskWizard.LightBlue":
69         setTheme(R.style.Base_Theme_DiskWizard_LightBlue);
70         backGroundView.setBackgroundResource(R.drawable.backgroundskies);
71         break;
72     case "Base.Theme.DiskWizard.Orange":
73         setTheme(R.style.Base_Theme_DiskWizard_Orange);
74         backGroundView.setBackgroundResource(R.drawable.backgroundorange);
75         break;
76     default:
77         setTheme(R.style.Base_Theme_DiskWizard);
78         backGroundView.setBackgroundResource(R.drawable.background);
79         break;
80 }
81 }
```

Смена темы происходит путем нажатия на соответствующий кружок в фрагменте профиля – рисунок 4.

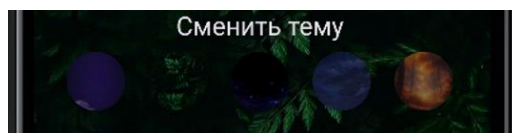


Рисунок 4. Смена темы приложения.

Вывод

Приложение поддерживает несколько predefined тем, каждая из которых включает определённый стиль и фон, обеспечивая консистентный и эстетически приятный пользовательский интерфейс.

4.3 ХРАНЕНИЕ ДАННЫХ В ПРОГРАММНОМ КОМПЛЕКСЕ

Для хранения данных в программном комплексе используется Firebase, который обеспечивает различные сервисы для управления данными, авторизации и хранения файлов. Основные компоненты Firebase, применяемые в приложении, включают Realtime Database, Authentication и Storage.

Firebase Realtime Database:

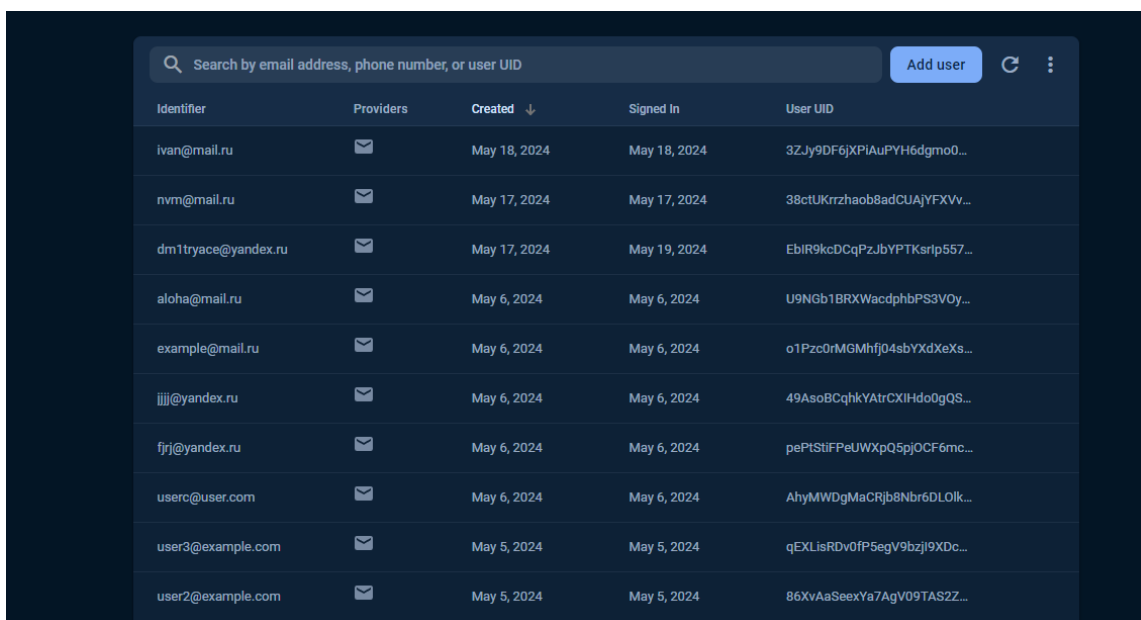
Firebase Realtime Database используется для хранения и синхронизации данных в реальном времени. Данная база данных позволяет хранить информацию о дисках и данных пользователей в формате JSON и обеспечивает мгновенную синхронизацию с клиентами. Вся информация хранится в соответствии с идентификаторами – рисунок 5.



Рисунок 5. *Firebase Realtime Database.*

Firestore Authentication:

Firestore Authentication используется для управления аутентификацией пользователей. Он поддерживает различные методы входа, включая электронную почту и пароль. Для работы с Firestore в программном комплексе необходимо было интегрировать соответствующие библиотеки и настроить проект. В первую очередь, были добавлены зависимости Firestore в файл `'build.gradle'`, включая библиотеки для Realtime Database, Authentication и Storage. Затем проект был настроен в консоли Firestore, где были созданы необходимые сервисы и получен файл конфигурации `'google-services.json'`, который добавлен в проект для инициализации Firestore – рисунок 6. Полный код доступен в приложении к курсовой работе №24, 26.



The screenshot shows the Firebase Authentication console interface. At the top, there is a search bar with the placeholder text "Search by email address, phone number, or user UID" and an "Add user" button. Below the search bar is a table listing users. The table has five columns: "Identifier", "Providers", "Created", "Signed In", and "User UID". The table contains ten rows of user data.

Identifier	Providers	Created	Signed In	User UID
ivan@mail.ru	✉	May 18, 2024	May 18, 2024	3ZJy9DF6jXPIAuPYH6dgm0...
nvm@mail.ru	✉	May 17, 2024	May 17, 2024	38ctUKrrzhaob8adCUA)YFXVv...
dm1tryace@yandex.ru	✉	May 17, 2024	May 19, 2024	EblR9kcDCqPzJbYPTKsrp557...
aloha@mail.ru	✉	May 6, 2024	May 6, 2024	U9NGb1BRXWacdpbPS3VOy...
example@mail.ru	✉	May 6, 2024	May 6, 2024	o1Pzc0rMGMhfj04sbYXdXeXs...
ijij@yandex.ru	✉	May 6, 2024	May 6, 2024	49AsoBCqghkYATRcXIHdo0gQS...
fjrl@yandex.ru	✉	May 6, 2024	May 6, 2024	peP1StiIFPeUWXpQ5pjOCF6mc...
userc@user.com	✉	May 6, 2024	May 6, 2024	AhyMWDgMaCRjb8Nbr6DLOlk...
user3@example.com	✉	May 5, 2024	May 5, 2024	qEXLisRDv0fP5egV9bzj9XDC...
user2@example.com	✉	May 5, 2024	May 5, 2024	86XvAaSeexYa7AgV09TAS2Z...

Рисунок 6. Firestore Authentication.

Firestore Storage:

Firestore Storage используется для хранения и извлечения различных файлов, таких как изображения. В данном программном комплексе Firestore Storage применяется для хранения изображений обложек дисков и изображений профилей пользователей. Полный код доступен в приложении к

курсовой работе №18, 25. Пример хранения изображения диска в Firebase Storage – рис. 7.

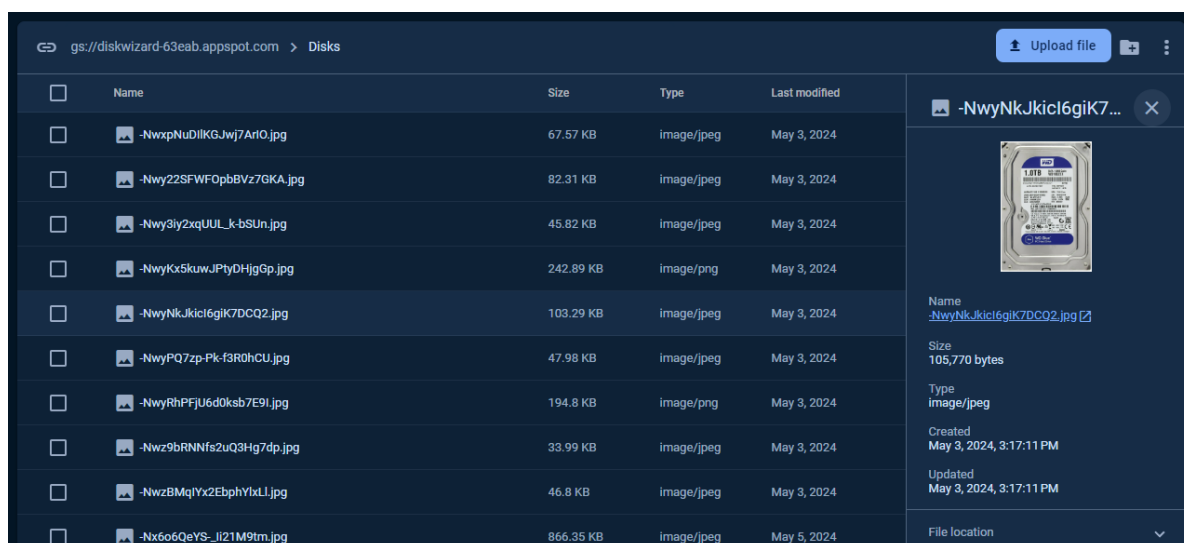


Рисунок 7. Firebase Storage.

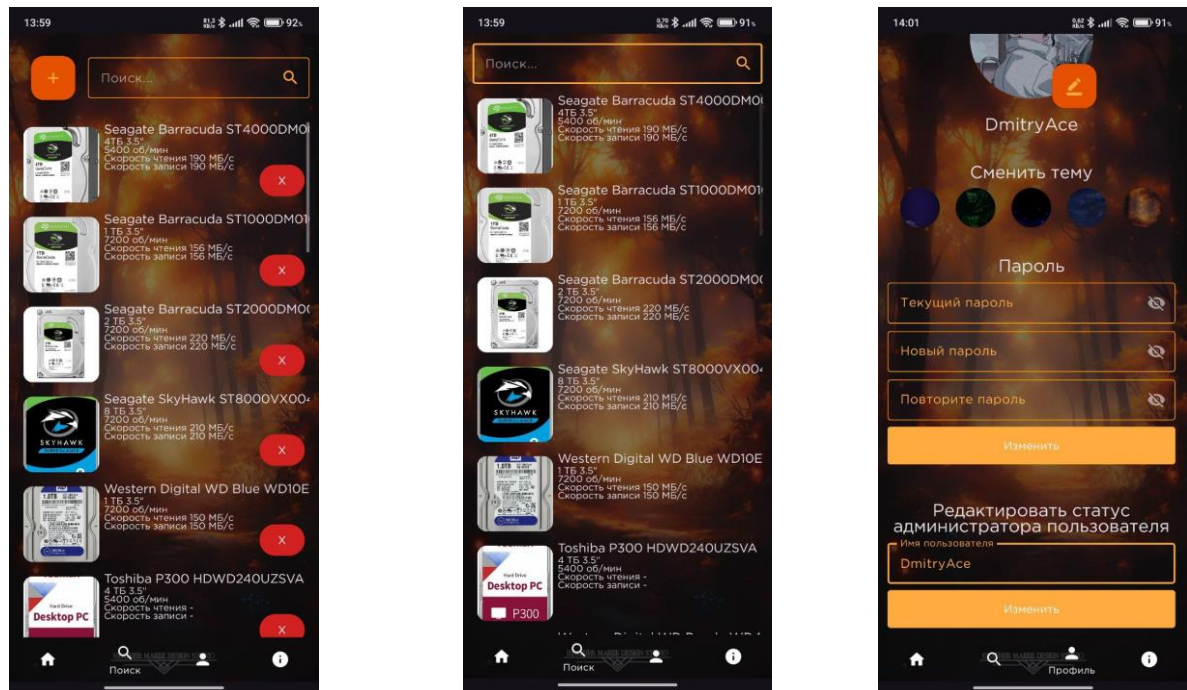
Вывод

Использование Firebase в программном комплексе обеспечивает надежное и эффективное управление данными пользователей и дисков. Realtime Database позволяет сохранять и синхронизировать данные в реальном времени, Authentication обеспечивает безопасный доступ к системе, а Storage предоставляет возможности для хранения и управления мультимедийными файлами. Это позволяет создавать современные и масштабируемые приложения с богатым функционалом и удобным интерфейсом.

4.4 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ИНФОРМАЦИИ ПРИ РАБОТЕ ПЛАНОВОГО ПРОГРАММНОГО КОМПЛЕКСА

В приложении реализована ролевая модель для обеспечения безопасности данных, представленная в виде трех ролей: Пользователь, Администратор и Создатель. Данная модель определяет различные уровни доступа и привилегии для каждой из ролей. Администратор имеет возможность добавлять и удалять диски, тогда как Создатель может

редактировать статус администрирования пользователей. Пример представлен на рисунке 8.



а) вид

б) вид Пользователя

в) вид Создателя

Администратора

Рисунок 8. Ролевая модель приложения.

Для обеспечения безопасности пользовательских паролей перед отправкой их в базу данных, используется хеширование на клиенте с использованием алгоритма SHA-256. Реализация представлена на листинге 10.

Листинг 10. LoginActivitiy.java

```

157 // Метод для хеширования пароля с использованием SHA-256
158 1 usage ± DmitryAce
159 @ private String hashPassword(String password) {
160     try {
161         MessageDigest digest = MessageDigest.getInstance("SHA-256");
162         byte[] hash = digest.digest(password.getBytes());
163         StringBuilder hexString = new StringBuilder();
164
165         for (byte b : hash) {
166             String hex = Integer.toHexString(0xff & b);
167             if (hex.length() == 1) hexString.append('0');
168             hexString.append(hex);
169         }
170
171         return hexString.toString();
172     } catch (NoSuchAlgorithmException e) {
173         e.printStackTrace();
174         return null;
175     }
176 }
177 }

```

Вывод: Этот подход к безопасности данных в приложении обеспечивает защиту конфиденциальной информации и предотвращает возможные атаки на систему, сохраняя целостность и конфиденциальность данных пользователей.

4.5 СИСТЕМА КОНТРОЛЯ ВЕРСИЙ

При разработке приложения использовалась система контроля версий Git [10], что позволило эффективно управлять изменениями в коде и ресурсах проекта. Использование системы контроля версий, такой как Git, обеспечило возможность отслеживать и сохранять историю изменений, а также управлять различными версиями приложения. Кроме того, использование ветвления позволило разрабатывать новые функции и исправлять ошибки в изолированных средах, что повысило стабильность и надежность процесса разработки. Регулярное коммитирование изменений позволило быстро возвращаться к предыдущим версиям при необходимости и обеспечило сохранность кодовой базы в случае возникновения проблем.

Проект доступен по ссылке: <https://github.com/DmitryAce/DiskWizard>

5 ПОЛЬЗОВАТЕЛЬСКИЙ СЦЕНАРИЙ И WIREFRAME ПРИЛОЖЕНИЯ

Далее приведены основные элементы, реализующие пользовательский сценарий:

1. Редактирование персональных данных:

- Пользователь переходит в профиль, нажав на свой профиль – 3 элемент навигационной панели.
- На экране профиля пользователь видит свои текущие данные (имя, изображение профиля).
- Пользователь нажимает плавающую кнопку на изображении либо вводит новые данные в поля.
- После внесения изменений пользователь нажимает кнопку "Сохранить".
- Система проверяет введенные данные, и в случае успешного обновления профиля выводит уведомление об успешном сохранении изменений.

2. Просмотр информации о диске и оставление отзыва.

- Пользователь переходит в список дисков, нажав на соответствующую иконку – 2 элемент навигационной панели.
- На экране профиля пользователь видит перечень доступных в системе дисков. Если пользователь Администратор, то также он способен видеть – элементы, реализующие возможность удаления/добавления дисков.
- Пользователь нажимает на выбранный им диск.
- После вызывается новая активность с загрузкой данных из Firebase.
- Пользователь заполняет поле для отзыва.
- Пользователь нажимает на кнопку оставить отзыв.
- Система проверяет введенные данные, и в случае успешного добавления в систему выводит уведомление об успешном сохранении изменений.

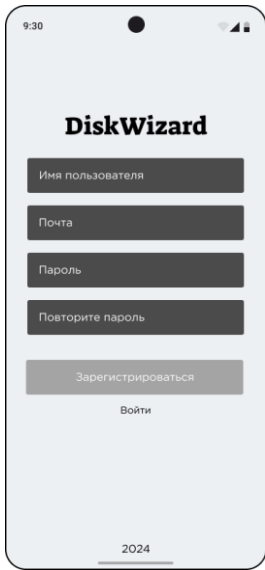
Вайрфрейм (Wireframe)

Вайрфрейм (англ. wireframe) представляет собой набор простых, схематичных чертежей или макетов, которые используются в проектировании

веб-сайтов, мобильных приложений и других интерфейсов. Это низкоуровневые графические представления, обычно в виде чёрно-белых или серых изображений, которые обозначают расположение элементов на экране и основные функциональные блоки интерфейса [5]. Он помогает выявить основные проблемы и потребности пользователей на ранних стадиях проекта, улучшить понимание концепции и сократить время и затраты на разработку путём предварительной проверки и оптимизации интерфейса. Вайрфрейм приложения представлен на рис 9.



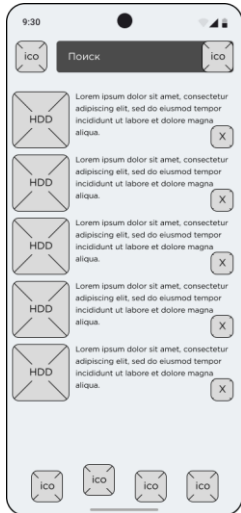
а) Вход



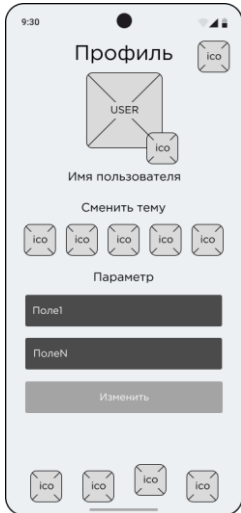
б) Регистрация



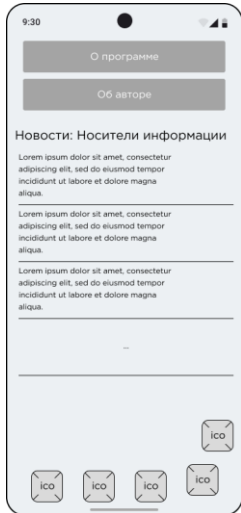
в) Главная



г) Список дисков



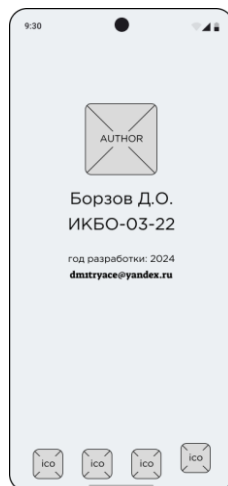
д) Профиль



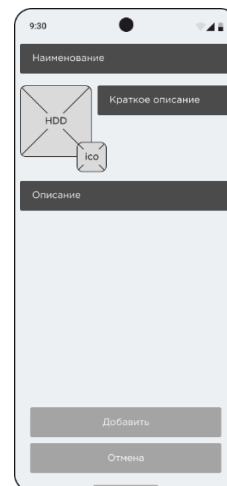
е) Новости



ё) О приложении



ж) Об Авторе



з) Добавление диска



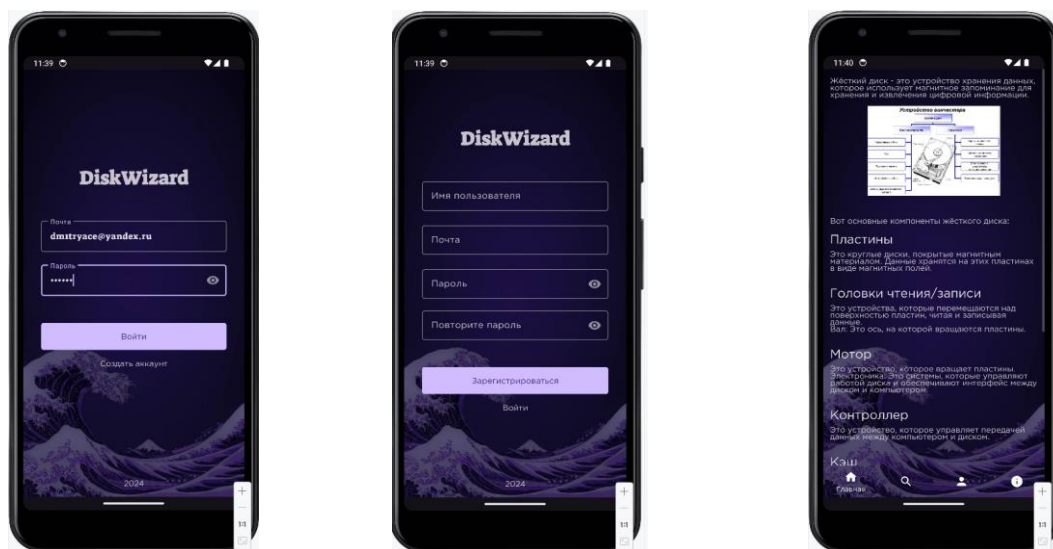
и) Детали диска

Рисунок 9. Wireframe приложения.

6 КОНТРОЛЬНЫЙ ПРИМЕР РАБОТЫ ПРИЛОЖЕНИЯ

Далее приведены все варианты использования от разных ролей.

1. Вход/Регистрация пользователя, после чего переход на главную страницу – рисунок 10.



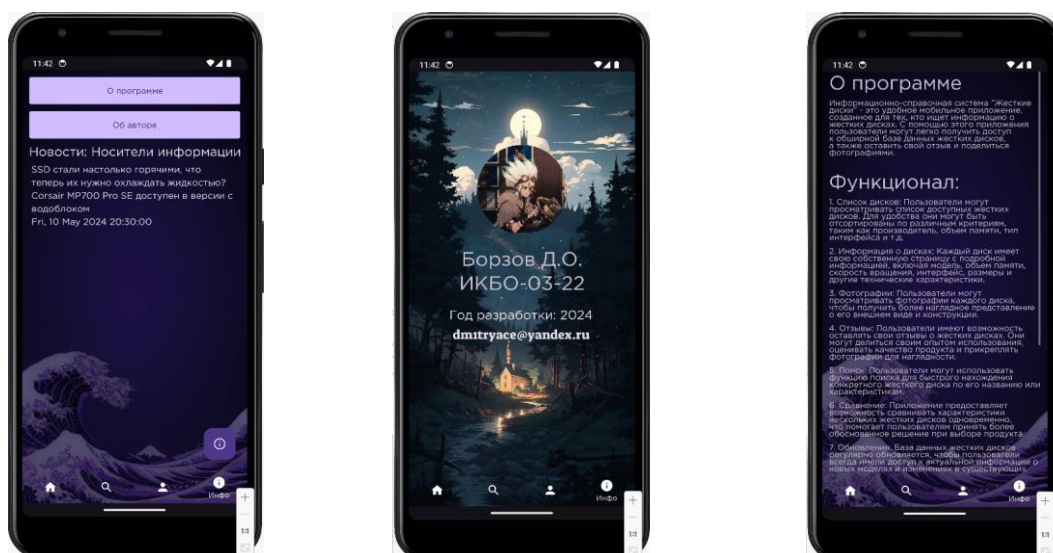
а) Вход

б) Регистрация

в) Главная

Рисунок 10. Вход/Регистрация.

2. Просмотр новостной ленты, информации о приложении и информации об авторе представлен на рисунке 11.



а) Новости

б) Об Авторе

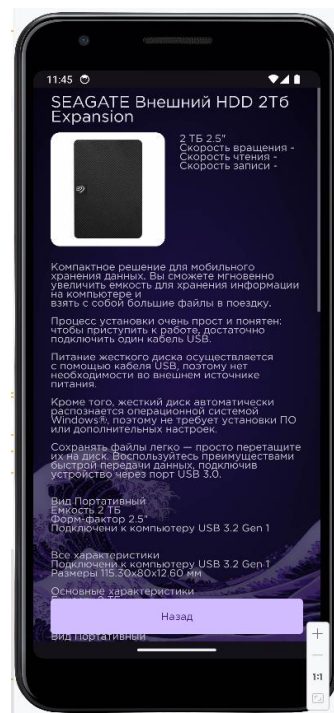
в) О приложении

Рисунок 11. Новости и различная информация о приложении.

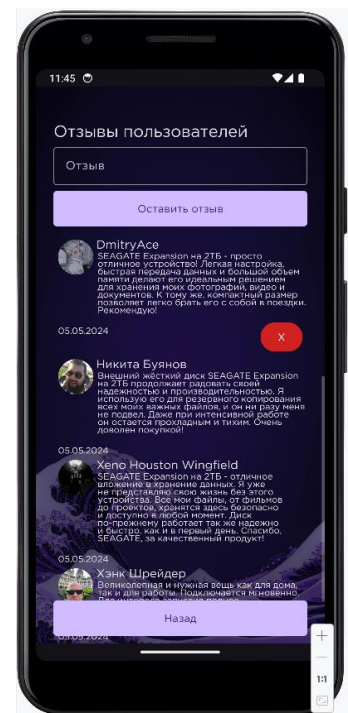
3. Взаимодействие с списком пользователем представлено на рисунке 12.



а) Список



б) Детали



в) Отзывы

Рисунок 12. Взаимодействие с списком - пользователь.

4. Взаимодействие с списком в роле администратора представлено на рисунке 13.



а) Список



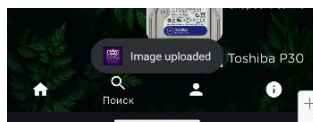
б) Детали



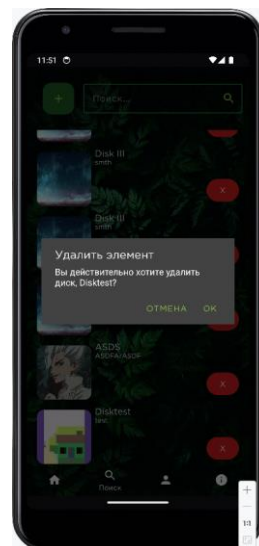
в) Отзывы



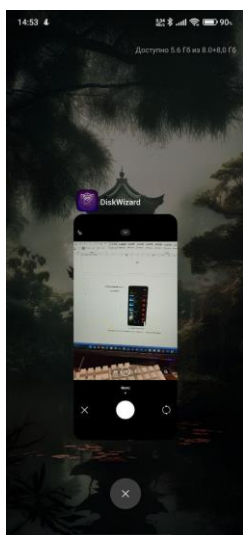
г) Добавление



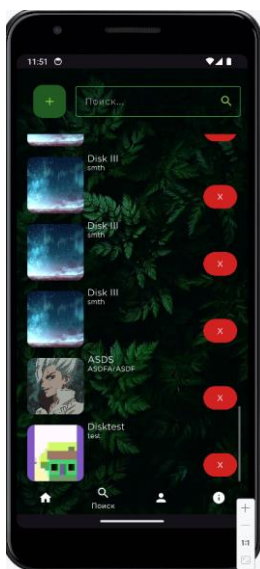
д) Уведомление о добавлении



е) Предупреждения об удалении



ё) Взаимодействие с камерой

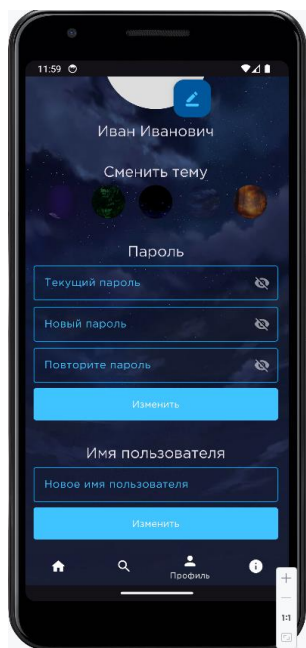


и) Диск добавлен

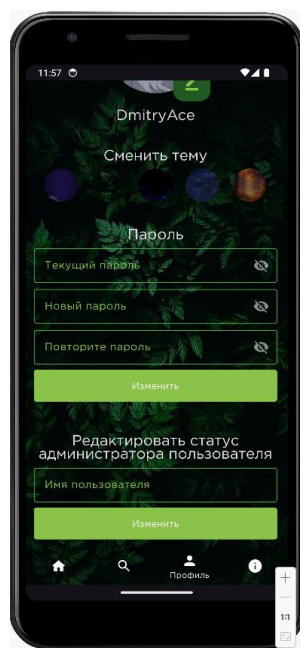
Рисунок 13. Взаимодействие с диском - администратор.

Как можно заметить, предусмотрена осторожность при удалении диска/отзыва путём вызова диалогового окна-фрагмента. Также наглядным образом показано взаимодействие пользователя/администратора с камерой.

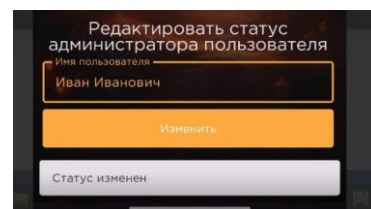
5. Далее на рисунке 30 показан профиль пользователя в различных ролях.



а) Пользователь



б) Создатель



*в) Уведомление
Snackbar*

Рисунок 14. Профиль.

7 ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ

Меню приложения

Главная

Раздел «Главная» предоставляет пользователю информацию о принципах работы жестких дисков. Здесь вы найдете подробные описания различных типов дисков, их конструкцию, а также основные характеристики и технологии, используемые в современных накопителях.

Поиск

Раздел «Поиск» содержит каталог жестких дисков. В верхней части экрана расположена строка поиска, позволяющая быстро найти нужный диск по названию или характеристикам. При выборе любого диска из списка вы сможете получить подробную информацию о нем, включая технические спецификации, отзывы других пользователей и возможность оставить свой собственный отзыв.

Профиль

В разделе «Профиль» пользователи могут изменять настройки своего аккаунта, такие как имя пользователя, электронную почту и пароль. Также здесь доступна функция изменения темы приложения, позволяющая настроить внешний вид интерфейса под свои предпочтения.

Инфо

Раздел «Инфо» содержит общую информацию о самом приложении, его разработчиках и целевой аудитории. В этом разделе также представлена новостная лента, где публикуются последние новости и статьи в области технологий хранения данных, обновления программного обеспечения и важные анонсы.

Эта инструкция поможет быстро освоить основные функции приложения и эффективно использовать его для поиска и управления информацией о жестких дисках.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы успешно были решены все поставленные задачи, что подтверждает достижение цели разработки мобильного приложения – информационно-справочной системы "Жесткие Диски".

Первоначально была проведена глубокая аналитика предметной области приложения, что позволило более полно понять ее специфику и особенности. Далее был осуществлен обзор существующих аналогов разрабатываемого продукта, что позволило выделить уникальные особенности и преимущества приложения.

Сформулированное техническое задание на разработку программы стало основой для четкого определения требований и ожиданий от продукта. Затем была разработана архитектура программной системы, что обеспечило эффективное проектирование и последующую реализацию приложения.

Проектирование интерфейса мобильного приложения было выполнено с акцентом на удобство и простоту использования для конечных пользователей. Далее был написан и отлажен код программы на языке высокого уровня Java, а также проведено тестирование, что обеспечило стабильную и надежную работу приложения.

Завершающим этапом было реализовано контрольное испытание работы программы, в результате которого были показаны все этапы работы приложения, что подтвердило его функциональность и соответствие заявленным требованиям.

Таким образом, выполнение всех поставленных задач позволило достичь цели курсовой работы и создать мобильное приложение, предоставляющее пользователям удобный доступ к информации о жестких дисках, их характеристиках, работе и обслуживании.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
2. Рудаков А.В. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. - 4 -е изд. – М.: Издательский центр «Академия», 2014. – 192 с.
3. Колисниченко Д. Н. Программирование для Android. Самоучитель. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2020 — 288 с.
4. Цуканова О. А. Методология и инструментарий моделирования бизнес - процессов: учебное пособие – СПб.: Университет ИТМО, 2015 – 100 с.
5. Sparx Systems User Interaction & Experience Изд-во Enterprise Architect 2022 – 43 стр.
6. IXBT [Электронный ресурс]. URL: <https://www.ixbt.com/> (дата обращения 16.05.2024).
7. Android Developers [Электронный ресурс]. URL: <https://developer.android.com/> (дата обращения 16.05.2024).
8. The Destination for Java Developers [Электронный ресурс]. URL: <https://dev.java/> (дата обращения 16.05.2024).
9. Github: Where the world builds software [Электронный ресурс]. URL: <https://github.com/> (дата обращения 18.05.2024).

ПРИЛОЖЕНИЕ

Приложение 1 – Полное техническое задание

1. Введение

Настоящее техническое задание разработано для создания мобильного приложения, предназначенного для получения информации о жестких дисках с целью обеспечения эффективности взаимодействия.

2. Основания для разработки

2.1. Документы, на основании которых ведется разработка:

Задание на выполнение курсовой работы по дисциплине
“Разработка мобильных приложений”

2.2. Организация, утвердившая данный документ, и дата его утверждения:

Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет» РТУ МИРЭА - «19» февраля 2024 г.

2.3. Наименование и (или) условное обозначение темы разработки:

Информационно-справочная система “Жёсткие диски”

3. Назначение разработки

Цель разработки данного мобильного приложения заключается в обеспечении пользователей возможностью:

3.1 Получения приятного и удобного опыта использования приложения.

3.2 Получения релевантной информации о жёстких дисках.

3.3 Оставления отзывов о любом из доступных дисков.

3.4 Изучения отзывов других пользователей.

3.5 Получения актуальных новостей из области накопителей информации.

4. Требования к программе или программному изделию

4.1. Требования к функциональным характеристикам:

4.1.2 Возможность авторизации пользователя, которая включает в себя процесс входа и регистрации.

4.1.3 Предоставление общего описания жёстких дисков.

4.1.4 Предоставление удобного списка жёстких дисков.

4.1.5 Предоставление обновляющегося информационного списка с ресурса IXBT [6].

4.1.6 Возможность добавление информации о дисках администратором.

4.1.7 Возможность оставить отзыв о диске.

4.2. Требования к надежности:

4.2.1 Приложение должно безотказно предоставлять доступ к полному функционалу.

4.2.2 Приложение должно предусматривать защиту от несанкционированного доступа к данным, например, ролевою модель с использованием механизмов аутентификации и авторизации.

4.2.3 Приложение должно быть устойчивым к внешним атакам, таким как атаки на основе внедрения вредоносного кода или перехвата данных.

4.2.4 Приложение должно иметь механизм контроля целостности данных, чтобы обнаруживать и предотвращать их изменение или подделку.

4.2.5 Приложение должно обеспечивать конфиденциальность данных пользователей путем шифрования чувствительной информации, передаваемой по сети или хранящейся на устройстве.

4.3. Условия эксплуатации:

Приложение должно обеспечивать стабильную работу на различных устройствах, учитывая их технические характеристики и операционные системы. Температурные условия и влажность должны

соответствовать стандартным параметрам для различных мобильных устройств, предполагаемых к использованию.

4.4. Требования к составу и параметрам технических средств:

Приложение должно быть доступно для использования на мобильных устройствах с операционной системой Android. Минимальные требования к устройствам для корректной работы приложения:

1. Процессор: Для Android - Snapdragon 660 или эквивалентный;
2. Оперативная память: Минимум 2 ГБ.
3. Свободное место на диске: Не менее 100 МБ.
4. Разрешение экрана: Минимум 720x1280 пикселей.
5. Версия операционной системы: Android 7.0 (Nougat) и выше.

Эти требования установлены на основе минимальных аппаратных возможностей, необходимых для поддержания производительности, заданных требованиями разработчика инфо.

4.5. Требования к маркировке и упаковке:

Приложение должно обладать адаптивной иконкой.

5. Техничко-экономические показатели:

Ориентировочная экономическая эффективность разработки приложения оценивается как положительная, учитывая следующие факторы:

1. Предполагаемая годовая потребность: Предполагается, что приложение будет активно использоваться пользователями, что приведет к стабильному и постоянному спросу на продукт.

2. Экономические преимущества по сравнению с аналогами: Приложение предлагает современные и удобные функции, что делает его более привлекательным для пользователей, чем аналогичные приложения на рынке. Кроме того, его разработка предполагается более

эффективной и экономичной благодаря использованию современных технологий и методов разработки.

Эти факторы позволяют ожидать хорошую рентабельность проекта и долгосрочную устойчивость на рынке мобильных приложений.

6. Стадии и этапы разработки

1. Планирование (до 1 апреля 2024 г.):

- Определение требований к приложению.
- Составление технического задания.
- Определение архитектуры приложения.
- Разработка плана работ и расписания.
- Результат: Четкое видение проекта и план действий.

2. Проектирование (до 15 апреля 2024 г.):

- Создание диаграммы классов и взаимодействия.
- Проектирование пользовательского интерфейса.
- Выбор технологий и инструментов разработки.
- Результат: Полное техническое описание и дизайн интерфейса.

3. Разработка (до 1 мая 2024 г.):

- Написание кода приложения.
- Тестирование отдельных модулей.
- Интеграция компонентов приложения.
- Результат: Рабочая версия приложения.

4. Тестирование (до 7 мая 2024 г.):

- Проведение модульного тестирования.
- Тестирование функциональности приложения.
- Выявление и исправление ошибок.
- Результат: Готовое к использованию приложение без критических ошибок.

5. Оформление курсовой работы (до 21 мая 2024 г.):

- Написание введения, описания проекта, методологии, результатов и заключения.
- Форматирование текста в соответствии с требованиями курсовой работы.
- Создание списка использованных источников.
- Результат: Полностью оформленная курсовая работа.

Исполнитель:

- Студент, являюсь единственным исполнителем всех этапов разработки и подготовки курсовой работы.

7. Порядок контроля и приемки

Виды испытаний:

1. Тестирование на эмуляторе: Проведение тестирования функциональности и совместимости на различных эмуляторах мобильных устройств для проверки работы приложения в различных сценариях.

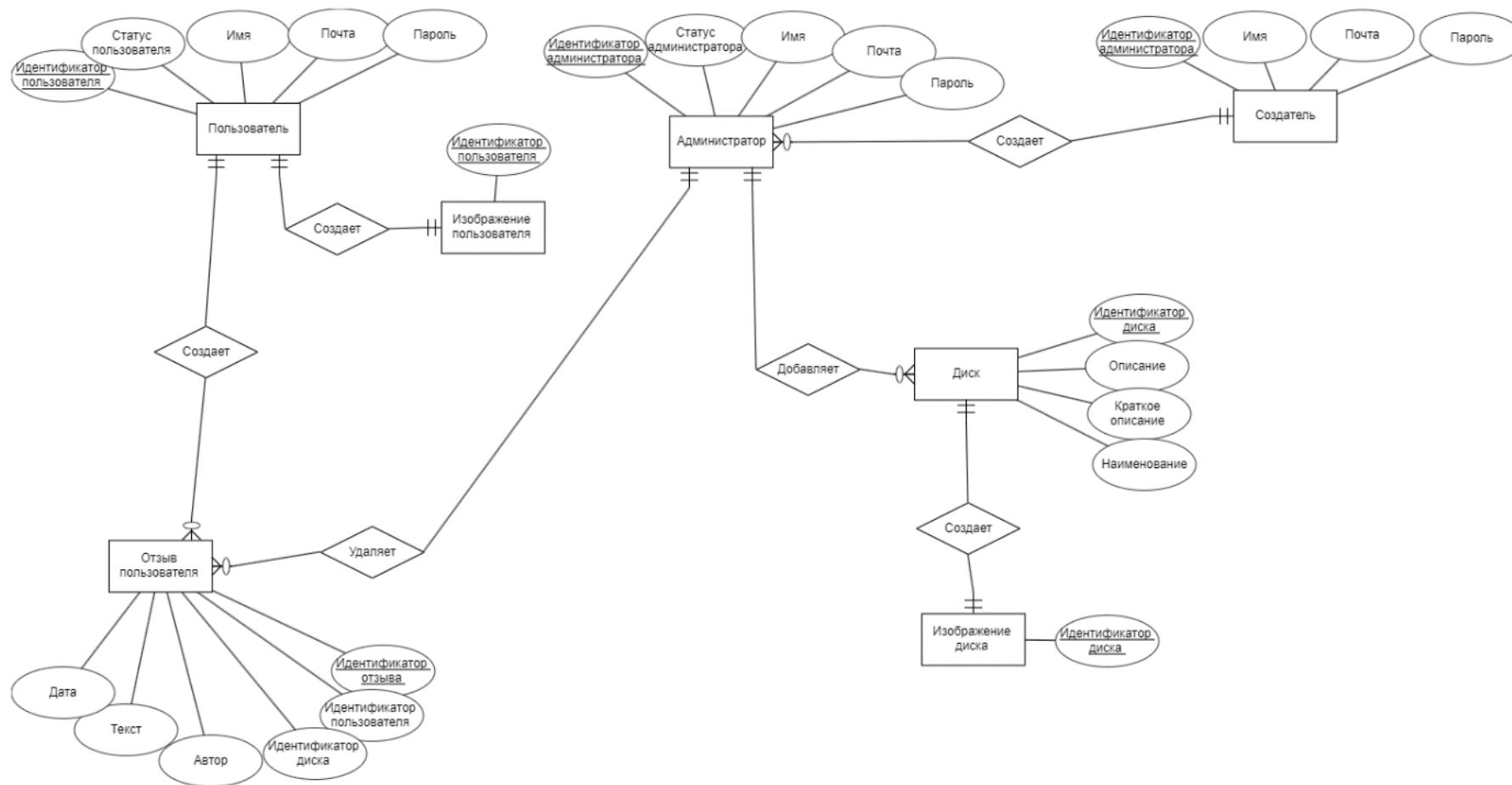
2. Тестирование на реальных устройствах: Проверка работоспособности и интерфейса приложения на реальных мобильных устройствах.

Общие требования к приемке работы:

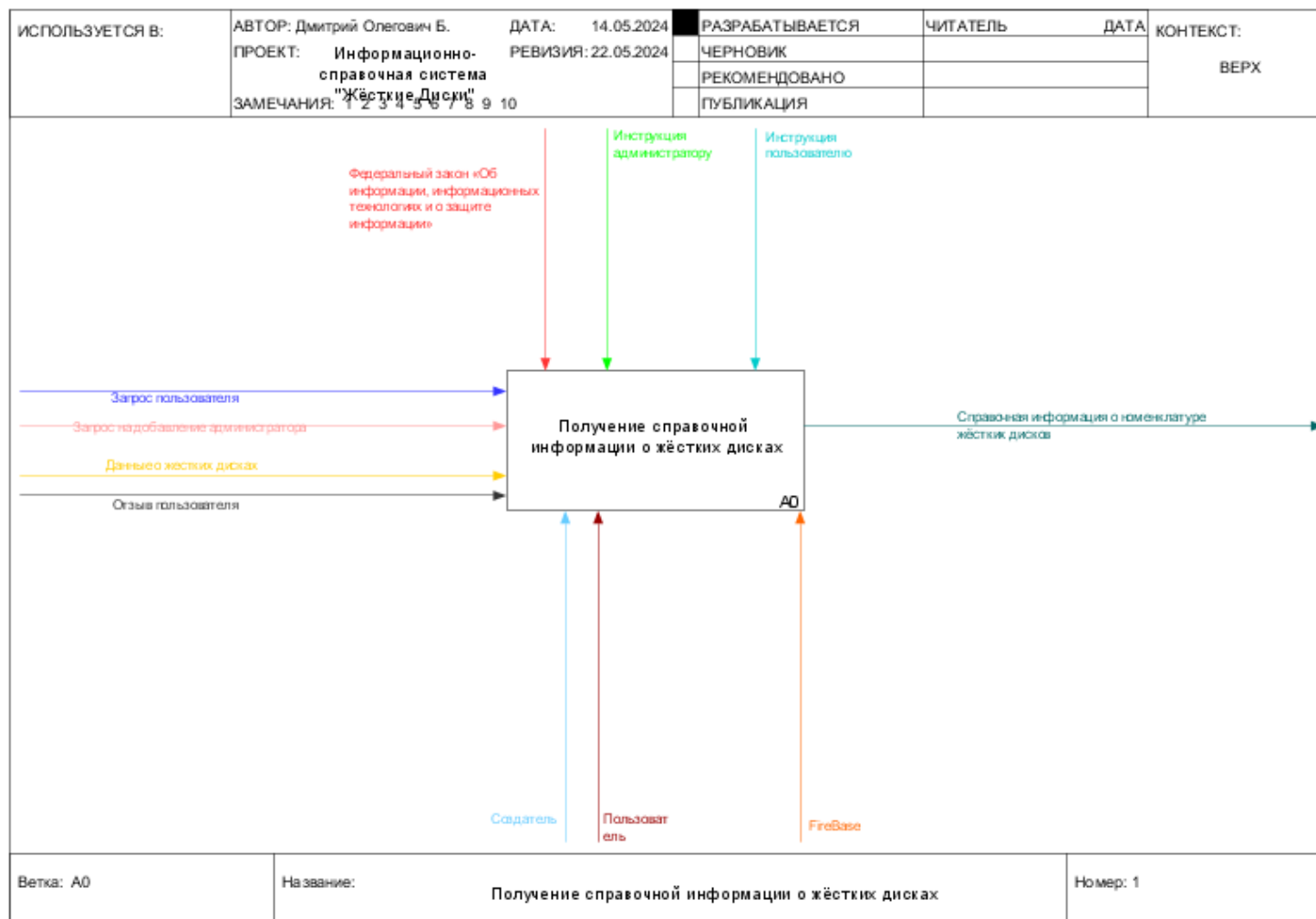
- Приложение должно быть стабильным и безотказно функционировать на различных устройствах и под разными условиями эксплуатации.
- Пользовательский интерфейс должен быть интуитивно понятным и удобным для использования.
- Приложение должно соответствовать требованиям безопасности и конфиденциальности данных пользователей.

Техническое задание составлено в соответствии с ГОСТ 19.201-78 [1].

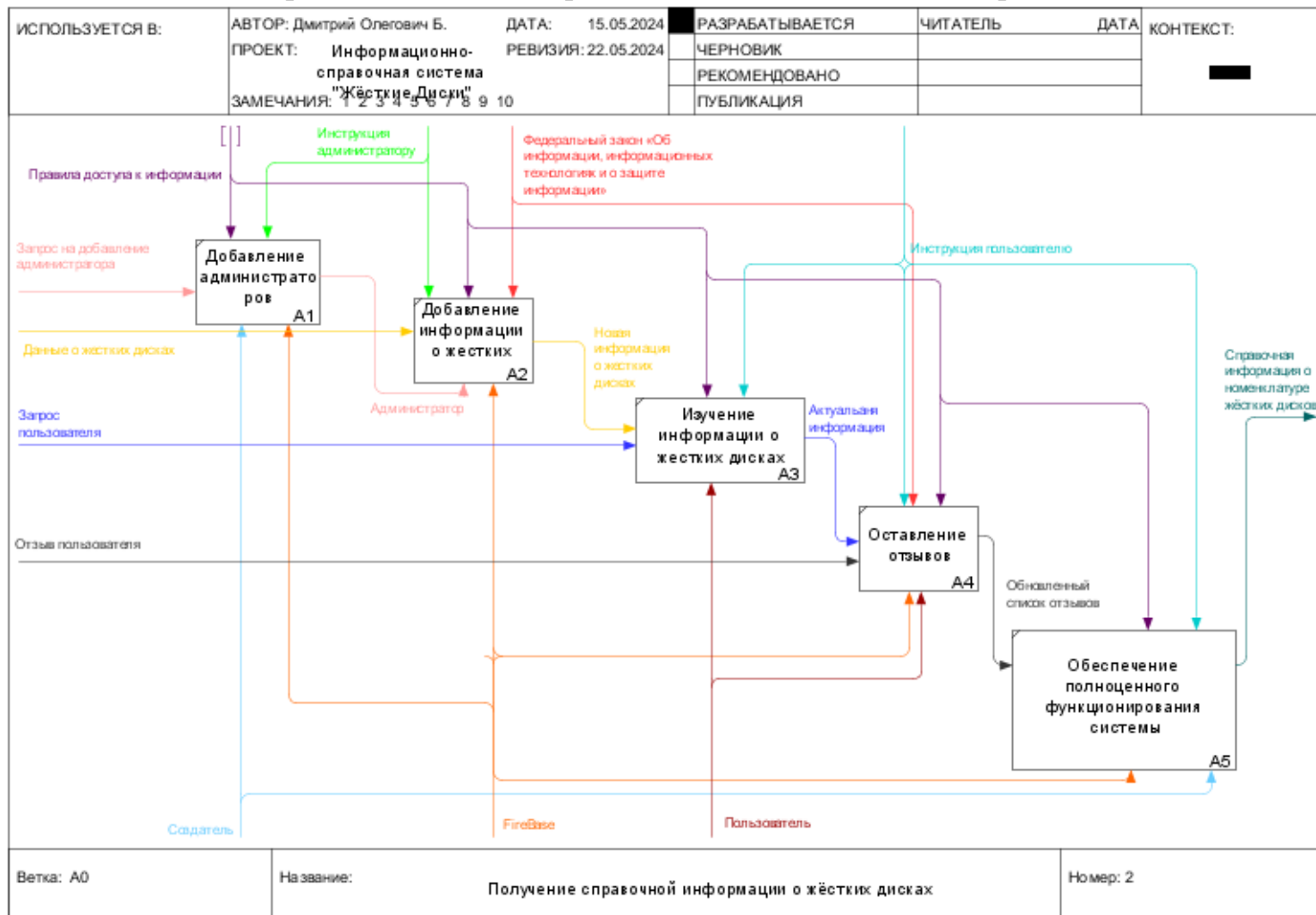
*Приложение 2 - Информационная модель данных предметной области объявлений
с помощью ER-диаграммы в нотации Чена*



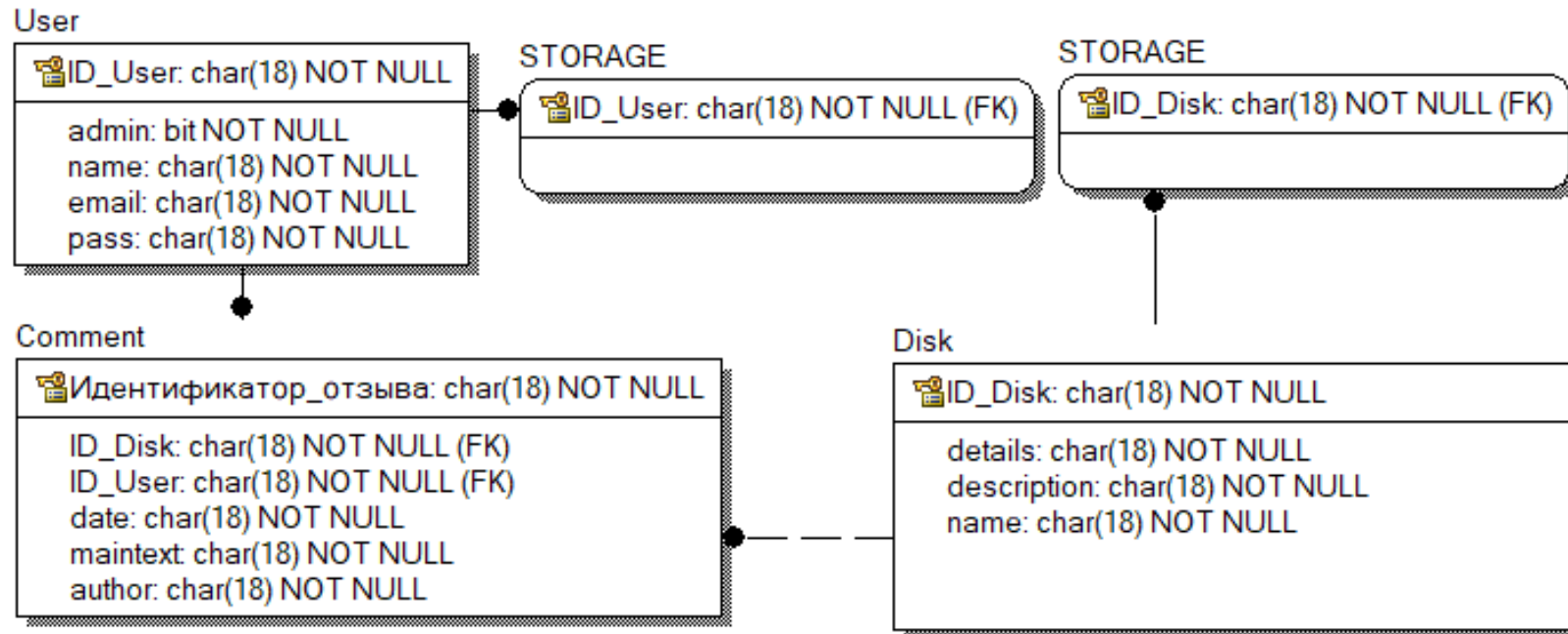
Приложение 3 - Диаграмма IDEF0, контекстный уровень.



Приложение 4 - Диаграмма IDEF0, декомпозиция процесса.



Приложение 5 - IDEF1X – Физический уровень.



Приложение 6 – Зависимости, разрешения и технологии использованные в программном комплексе

В контексте курсовой работы Gradle позволяет легко управлять зависимостями и настройками проекта, обеспечивая эффективную и стандартизированную сборку приложения – листинг п.1.

Листинг п.1. *Build.gradle.kst (Module :app)*

```
35 dependencies {
36     implementation(libs.appcompat)
37     implementation(libs.material)
38     implementation(libs.activity)
39     implementation(libs.constraintlayout)
40     implementation(libs.firebase.database)
41     implementation(libs.firebase.auth)
42     implementation(libs.firebase.messaging)
43     testImplementation(libs.junit)
44     androidTestImplementation(libs.ext.junit)
45     androidTestImplementation(libs.espresso.core)
46     implementation(platform("com.google.firebase:firebase-bom:32.8.1"))
47     implementation("com.google.firebase:firebase-analytics")
48     implementation("com.google.firebase:firebase-storage:20.0.0")
49     implementation("com.firebaseui:firebase-ui-storage:7.2.0")
50     implementation("com.squareup.okhttp3:okhttp:4.9.0")
51     implementation("com.android.volley:volley:1.2.1")
52 }
```

Эти зависимости обеспечивают основную функциональность приложения и помогают обеспечить лучший пользовательский опыт. Они включают в себя все, от обработки сетевых запросов до обеспечения современного пользовательского интерфейса. Приложение использует следующие разрешения и функции (листинг п.2):

Листинг п.2. *AndroidManifest.xml*

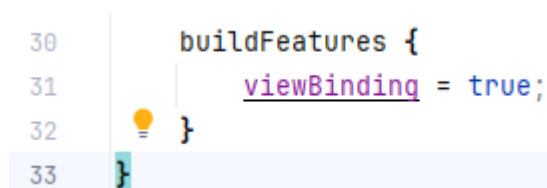
```
5 <uses-feature
6     android:name="android.hardware.camera"
7     android:required="false" />
8
9 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
10 <uses-permission android:name="android.permission.INTERNET" />
11 <uses-permission android:name="android.permission.CAMERA" />
12 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
13 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Учитывается, что для защиты приватности пользователей Android требует, чтобы приложение запросило разрешение пользователя перед получением доступа к определенным системным функциям и данным.

ИСПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ

VIEW BINDING

В данной курсовой работе используется современный подход для взаимодействия с элементами пользовательского интерфейса, а именно ViewBinding[7]. ViewBinding является удобной и безопасной заменой для традиционного метода `findViewById()`, предоставляя прямые ссылки на элементы интерфейса через автоматически сгенерированные классы привязки. Включив ViewBinding в проект (рис. 16), каждый макет XML файла создает соответствующий класс, что позволяет избежать ошибок привязки и улучшает читаемость кода.



```
30     buildFeatures {
31         viewBinding = true;
32     }
33 }
```

Рисунок 16. Включение viewBinding в проект.

ACTIVITY RESULT API

В данной курсовой работе используется современный API для получения результатов активности (Activity Result API [7]), используется для взаимодействия с камерой и галереей. Этот API предоставляет удобный и безопасный способ запроса и обработки результатов от других компонентов приложения, таких как вызов камеры для съемки фото или выбор изображения из галереи. Вместо устаревших методов `startActivityForResult` и `onActivityResult`, Activity Result API использует контрактный подход, что упрощает управление результатами и делает код более чистым и понятным. С помощью этого API в курсовой работе реализованы запросы к камере и галерее, а также обработка полученных изображений, что обеспечивает современный и эффективный подход к работе с мультимедийными данными в Android-приложениях.

Приложение 7 – Реализованный функционал

НОВОСТНАЯ ЛЕНТА

В приложении реализована новостная лента, загружающая данные с ресурса IXBT (http://www.ixbt.com/export/sec_optical.rss) [6]. Ниже приведено описание ключевых компонентов и процессов, обеспечивающих функционирование данной функции.

Загрузка RSS-канала

Для загрузки данных с RSS-канала используется библиотека OkHttp. Метод `loadRSS()` выполняет HTTP-запрос к указанному URL-адресу RSS-канала и обрабатывает ответ асинхронно – представлено на листинге п.3.

Листинг п.3. AboutFragment.java

```
1 usage  ± DmitryAce
61 private void loadRSS() {
62     Request request = new Request.Builder()
63         .url("http://www.ixbt.com/export/sec_optical.rss")
64         .build();
65
66     ± DmitryAce
67     client.newCall(request).enqueue(new Callback() {
68         ± DmitryAce
69         @Override
70         public void onFailure(Call call, IOException e) { e.printStackTrace(); }
71
72         ± DmitryAce
73         @Override
74         public void onResponse(Call call, Response response) throws IOException {
75             if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);
76             parseRSS(response.body().string());
77         }
78     });
79 }
80
```

1. Создание запроса: С помощью `Request.Builder` создается запрос к URL-адресу RSS-канала - строка 62.
2. Отправка запроса: Запрос отправляется с помощью метода `newCall()` объекта `client` – строка 66.
3. Обработка ответа: В случае успешного ответа метод `onResponse()` вызывает функцию `parseRSS()`, которая обрабатывает полученные данные – строка 73. В случае ошибки вызывается метод `onFailure()` – строка 68.

Парсинг RSS-данных

Метод `parseRSS()` использует `XmlPullParser` для разбора XML-документа, полученного от RSS-канала представлено на листинге п.4.

Листинг п.4. *AboutFragment.java*

```
81     private void parseRSS(String rss) {
82         try {
83             XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
84             XmlPullParser xpp = factory.newPullParser();
85
86             xpp.setInput(new StringReader(rss));
87             int eventType = xpp.getEventType();
88             String title = null;
89             String date = null;
90             boolean isFirstTitle = true;
91             while (eventType != XmlPullParser.END_DOCUMENT) {
92                 if (eventType == XmlPullParser.START_TAG) {
93                     if (xpp.getName().equalsIgnoreCase("title")) {
94                         eventType = xpp.next();
95                         if (isFirstTitle) {
96                             isFirstTitle = false;
97                         } else {
98                             title = xpp.getText();
99                         }
100                     } else if (xpp.getName().equalsIgnoreCase("pubDate")) {
101                         eventType = xpp.next();
102                         date = xpp.getText().replaceFirst("\\+\\d{4}$", "");
103                     }
104                 } else if (eventType == XmlPullParser.END_TAG) {
105                     if (xpp.getName().equalsIgnoreCase("item")) {
106                         if (title != null && date != null) {
107                             news.add(title + "\n" + date);
108                             title = null;
109                             date = null;
110                         }
111                     }
112                 }
113                 eventType = xpp.next();
114             }
115
116             requireActivity().runOnUiThread() -> adapter.notifyDataSetChanged();
117
118         } catch (Exception e) {
119             e.printStackTrace();
120         }
121     }
```

1. Инициализация парсера: Создается объект `XmlPullParser` и устанавливается входной поток данных – строка 84.
2. Парсинг XML: В цикле проверяются теги XML-документа. При нахождении тега `<title>` извлекается текст заголовка. При нахождении тега `<pubDate>` извлекается и форматируется дата - строка 91.

3. Добавление данных в список: При закрытии тега `<item>`, заголовок и дата добавляются в список новостей - строка 107.
4. Обновление UI: После завершения парсинга вызывается метод `notifyDataSetChanged()` адаптера для обновления пользовательского интерфейса - строка 116.

Итог

В приложении реализована функциональность загрузки и отображения новостной ленты с использованием RSS-канала. Для этого используется библиотека `OkHttp` для выполнения HTTP-запросов и `XmlPullParser` для обработки XML-документов. Загруженные данные асинхронно парсятся и обновляют пользовательский интерфейс. Результат реализации новостной ленты на рисунке 15.

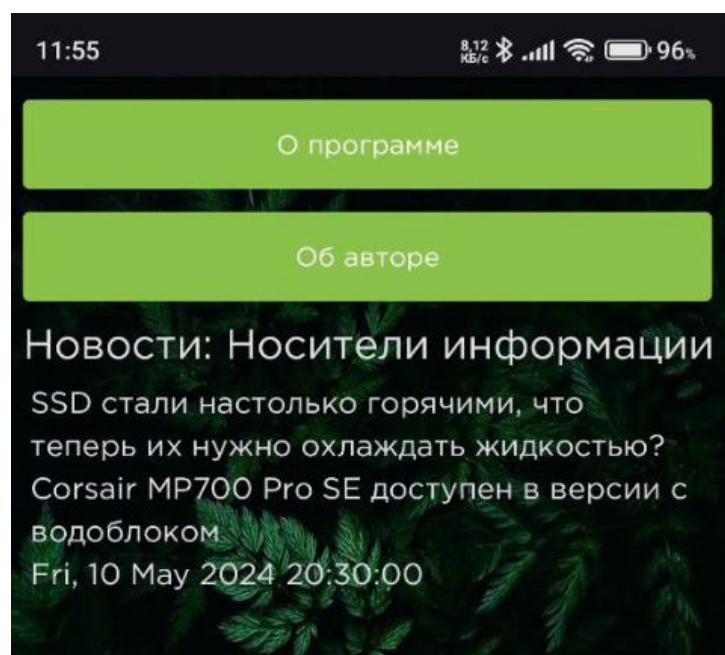


Рисунок 15. Новостная лента

ВЗАИМОДЕЙСТВИЕ С КАМЕРОЙ И ИЗОБРАЖЕНИЯМИ

В данном разделе описывается функциональность, связанная с выбором и загрузкой изображений из галереи или камеры, а также их последующей обработкой и загрузкой в облачное хранилище.

Выбор и обработка изображения

Для выбора изображения из галереи или создания нового с помощью камеры используется `ActivityResultLauncher<Intent>`, который регистрируется с использованием контракта `StartActivityForResult` – листинг п.5.

Листинг п.5. ProfileFragment.java

```
228     ActivityResultLauncher<Intent> mGetContent = registerForActivityResult(  
229         new ActivityResultContracts.StartActivityForResult(),  
230         result -> {  
231             if (result.getResultCode() == Activity.RESULT_OK) {  
232                 Intent data = result.getData();  
233                 Uri imageUri = data.getData();  
234                 if (imageUri != null) {  
235                     String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
236                     uploadImage(imageUri, userId);  
237                 } else {  
238                     Bundle extras = data.getExtras();  
239                     Bitmap imageBitmap = (Bitmap) extras.get("data");  
240                     // Convert bitmap to Uri  
241                     imageUri = getImageUri(getContext(), imageBitmap);  
242                     String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
243                     uploadImage(imageUri, userId);  
244                 }  
245             }  
246         }  
247     );
```

1. Регистрация `ActivityResultLauncher`: `mGetContent` регистрируется для обработки результата активности выбора изображения – строка 228.
2. Обработка результата: В случае успешного результата проверяется, было ли изображение выбрано из галереи (получение `Uri`) или создано с помощью камеры (получение `Bitmap` и его преобразование в `Uri`) – строка 234.

Запуск выбора изображения

Метод `chooseImage()` создает и запускает намерение для выбора изображения из галереи или для создания нового с помощью камеры – листинг п.6. Пример на рисунке 16.

Листинг п.6. ProfileFragment.java

```
249 private void chooseImage() {  
250     Intent galleryIntent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);  
251     Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
252  
253     Intent chooser = Intent.createChooser(galleryIntent, title: "Выберите приложение");  
254     chooser.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[] { cameraIntent });  
255  
256     // Запуск ActivityResultLauncher  
257     mGetContent.launch(chooser);  
258 }  
259
```

1. Создание намерений: создаются намерения для выбора изображения из галереи (`galleryIntent`) и для захвата изображения с камеры (`cameraIntent`) - строки 250 и 251.
2. Запуск намерения: используется `Intent.createChooser` для создания диалогового окна выбора приложения, после чего вызывается `mGetContent.launch` для запуска выбора – строка 253.

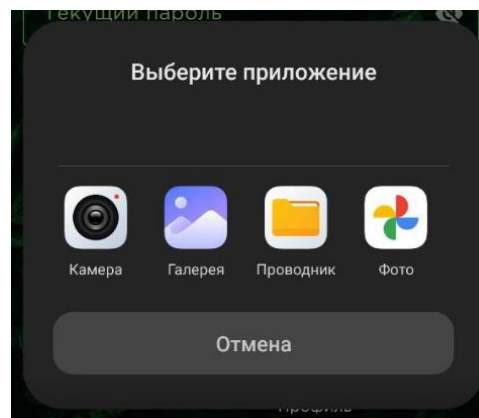


Рисунок 16. Выбор способа получения изображения.

Преобразование Bitmap в Uri

Метод `getImageUri()` преобразует изображение из формата Bitmap в Uri представлено на листинге п.7.

Листинг п.7. ProfileFragment.java

```
1 usage  ± DmitryAce
260 @ private Uri getImageUri(Context inContext, Bitmap inImage) {
261     ByteArrayOutputStream bytes = new ByteArrayOutputStream();
262     inImage.compress(Bitmap.CompressFormat.JPEG, quality: 100, bytes);
263     String path = MediaStore.Images.Media.insertImage(inContext.getContentResolver(), inImage, title: "Title",
264     return Uri.parse(path);
265 }
266
```

1. Компрессия изображения: Изображение в формате Bitmap сжимается и записывается в байтовый массив – строка 262.
2. Вставка изображения в медиатеку: Используется `MediaStore.Images.Media.insertImage` для вставки изображения в медиатеку устройства, что возвращает Uri изображения - строка 263.

Загрузка изображения

Метод `uploadImage()` загружает выбранное или созданное изображение в облачное хранилище Firebase – листинг 8.

Листинг п.8. ProfileFragment.java

```
280 @ private void loadNewImage(StorageReference fileRef) {
281
282     fileRef.getDownloadUrl().addOnSuccessListener(uri -> {
283         Glide.with( fragment: this) RequestManager
284             .load(uri) RequestBuilder<Drawable>
285             .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем кэширование
286             .apply(new RequestOptions().centerCrop())
287             .into(binding.imageView);
288     });
289 }
```

Итог

В данном разделе описано взаимодействие с камерой и галереей для выбора и создания изображений, их обработка и преобразование, а также

загрузка изображений в облачное хранилище Firebase. Эта функциональность обеспечивает пользователю возможность выбирать и загружать изображения, что может быть полезно для настройки профиля. Реализация загрузки изображения дисков выглядит аналогичным образом, полная реализация доступна в приложении к курсовой работе №18, 25.

НАПОЛНЕНИЕ СПИСКОВ ДИСКОВ И ОТЗЫВОВ

Процесс создания списка с собственным адаптером на Java включает в себя несколько основных шагов, начиная с создания адаптера, который связывает данные с пользовательским интерфейсом, и заканчивая его использованием в соответствующих фрагментах или активностях:

1. Создание адаптера (DiskAdapter.java):

- В этом шаге создаем собственный адаптер, который расширяет базовый адаптер (ArrayAdapter) и определяет методы для управления данными и создания представлений элементов списка.

- В классе адаптера реализуется метод `getView` для получения view элемента списка.

2. Использование адаптера в фрагментах (DiskListFragment.java и DiskListItemFragment.java):

- После создания адаптера используем его в соответствующих фрагментах приложения. В фрагменте, отображающем список (DiskListFragment), создается экземпляр адаптера и устанавливаем его для списка (ListView).

3. Связывание данных с адаптером:

- После создания адаптера и его использования в фрагментах, нужно связать данные с адаптером. Это включает в себя загрузку данных из Firebase и передачу их в адаптер для отображения в пользовательском интерфейсе.

4. Обновление данных и обработка событий:

- Реализованы методы для обновления данных в адаптере (например, добавление новых элементов или удаление существующих) и обработки событий, таких как нажатия на элементы списка.

5. Тестирование и отладка:

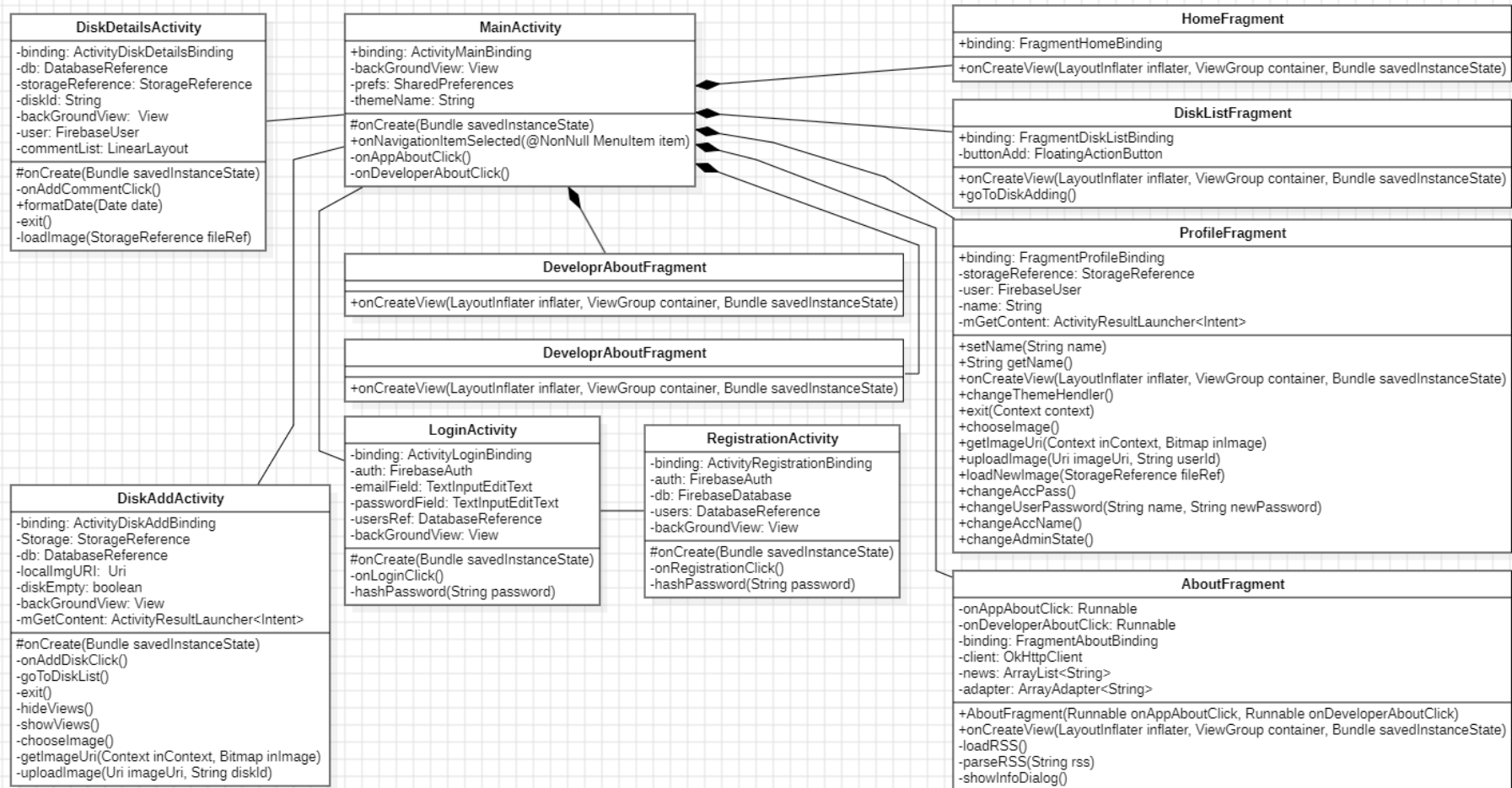
- В конечном итоге созданный список с собственным адаптером был протестирован, чтобы убедиться, что он работает правильно и соответствует ожиданиям. Обращайтесь к листингам кода, доступных в приложении к курсовой работе №20-22, в файлах `DiskAdapter.java`, `DiskListFragment.java` и `DiskListItemFragment.java` для дополнительной информации и конкретных примеров реализации каждого шага. Результат представлен на рисунке 17.



Рисунок 17. Список дисков.

Реализация списка отзывов схожа с реализацией списка дисков, но в отличии от дисков у отзывов используется `LinearLayout`, код также доступен в приложении к курсовой работе №19.

Приложение 8 – Диаграмма классов программного комплекса



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>

    <application
        android:requestLegacyExternalStorage="true"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:roundIcon="@drawable/icon"
        android:supportsRtl="true"
        android:theme="@style/Theme.DiskWizard"
        android:usesCleartextTraffic="true"
        tools:targetApi="31">
        <activity
            android:name=".presentation.disk.details.DiskDetails"
            android:exported="false" />
        <activity
            android:name=".presentation.disk.add.DiskADD"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:windowSoftInputMode="adjustPan"/>
        <activity
            android:name=".presentation.registration.RegistrationActivity"
            android:exported="false" />
        <activity
            android:name=".presentation.login.LoginActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

```

package com.example.diskwizard.domain.model;

public class Comment {

    String id;
    String diskId;
    String userId;
    private String author;
    private String maintext;
    private String date;

    public Comment() {}

    public Comment(String Author, String maintext, String date) {
        this.author = Author;
        this.maintext = maintext;
        this.date = date;
    }

    public Comment(String id, String diskId, String userId, String Author,
String maintext, String date) {
        this.id = id;
        this.diskId = diskId;
        this.userId = userId;
        this.author = Author;
        this.maintext = maintext;
        this.date = date;
    }

    public String getAuthor() {
        return this.author;
    }

    public String getMaintext() {
        return this.maintext;
    }

    public String getDate() {
        return this.date;
    }

    public String getId() {
        return this.id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getDiskId() {
        return diskId;
    }

    public void setDiskId(String diskId) {
        this.diskId = diskId;
    }

    public String getUserId() {
        return userId;
    }
}

```

```

        public void setUserId(String userId) {
            this.userId = userId;
        }
    }
}

```

Приложение 11 - Disk.java

```

package com.example.diskwizard.domain.model;

public class Disk {

    String id;
    private String name;
    private String description;
    private String details;

    public Disk() {}

    public Disk(String name, String description, String details) {
        this.name = name;
        this.description = description;
        this.details = details;
    }

    public Disk(String id, String name, String description, String details) {
        this.id = id;
        this.name = name;
        this.description = description;
        this.details = details;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public String getDetails() {
        return this.details;
    }

    public String getId() {
        return this.id;
    }

    public void setId(String id) {
        this.id = id;
    }
}

```

Приложение 12 - User.java

```

package com.example.diskwizard.domain.model;

public class User {

```

```

private String name, email;
private boolean admin;
public User() {
}

public User(String name, String email, String pass) {
    this.name = name;
    this.email = email;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public boolean getAdmin() {
    return admin;
}

public void setAdmin(boolean admin) {
    this.admin = admin;
}
}

```

Приложение 13 - DiskService.java

```

package com.example.diskwizard.domain.service.disk;

import com.example.diskwizard.domain.model.Disk;

import java.util.List;

public interface DiskService {

    List<Disk> getAll();
}

```

Приложение 14 - FirebaseDiskService.java

```

package com.example.diskwizard.domain.service.disk;

import android.util.Log;

import com.example.diskwizard.domain.model.Disk;
import com.google.firebase.database.*;
import com.google.firebase.storage.*;

import java.util.ArrayList;
import java.util.List;

```



```

public class FirebaseDiskService implements DiskService {

    private DatabaseReference mDatabase;
    private StorageReference mStorageRef;

    public FirebaseDiskService() {
        mDatabase =
        FirebaseDatabase.getInstance().getReference().child("Disks");
        mStorageRef =
        FirebaseStorage.getInstance().getReference().child("Disks");
    }

    @Override
    public List<Disk> getAll() {
        final List<Disk> disks = new ArrayList<>();
        mDatabase.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                for (DataSnapshot postSnapshot: dataSnapshot.getChildren()) {
                    Disk disk = postSnapshot.getValue(Disk.class);
                    disks.add(disk);
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Handle possible errors.
            }
        });
        return disks;
    }

    public StorageReference getImage(String id) {
        Log.d("MyLogs", "получаем изображение из Storge: "+id);
        return mStorageRef.child(id+".jpg");
    }
}

```

Приложение 15 - AppAboutFragment.java

```

package com.example.diskwizard.presentation.about.app;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.diskwizard.R;

public class AppAboutFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
    ViewGroup container, @Nullable Bundle savedInstanceState) {

```

```

        return inflater.inflate(R.layout.fragment_app_about, container,
false);
    }

}

```

Приложение 16 - DeveloperAboutFragment.java

```

package com.example.diskwizard.presentation.about.developer;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.example.diskwizard.R;

public class DeveloperAboutFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_developer_about, container,
false);
    }

}

```

Приложение 17 - AboutFragment.java

```

package com.example.diskwizard.presentation.about;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;
import com.example.diskwizard.databinding.FragmentAboutBinding;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.IOException;
import java.io.StringReader;
import java.util.ArrayList;
import okhttp3.*;

public class AboutFragment extends Fragment {

    private final Runnable onAppAboutClick;
    private final Runnable onDeveloperAboutClick;
}

```

```

        FragmentAboutBinding binding;
        private final OkHttpClient client = new OkHttpClient();
        private ArrayList<String> news = new ArrayList<>();
        private ArrayAdapter<String> adapter;

        public AboutFragment(Runnable onAppAboutClick, Runnable
onDeveloperAboutClick) {
            this.onDeveloperAboutClick = onDeveloperAboutClick;
            this.onAppAboutClick = onAppAboutClick;
        }

        @Nullable
        @Override
        public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
            binding = FragmentAboutBinding.inflate(getLayoutInflater());
            View view = binding.getRoot();
            Button appAboutButton = binding.appAboutButton;
            Button developerAboutButton = binding.developerAboutButton;

            appAboutButton.setOnClickListener(v -> onAppAboutClick.run());
            developerAboutButton.setOnClickListener(v ->
onDeveloperAboutClick.run());

            ListView listView = binding.listView;
            adapter = new ArrayAdapter<>(requireContext(),
android.R.layout.simple_list_item_1, news);
            listView.setAdapter(adapter);

            loadRSS();

            FloatingActionButton moreInfoButton = binding.moreInfo;
            moreInfoButton.setOnClickListener(v -> showInfoDialog());

            return view;
        }

        private void loadRSS() {
            Request request = new Request.Builder()
                .url("http://www.ixbt.com/export/sec_optical.rss")
                .build();

            client.newCall(request).enqueue(new Callback() {
                @Override
                public void onFailure(Call call, IOException e) {
                    e.printStackTrace();
                }

                @Override
                public void onResponse(Call call, Response response) throws
IOException {
                    if (!response.isSuccessful()) throw new
IOException("Unexpected code " + response);

                    parseRSS(response.body().string());
                }
            });
        }

        private void parseRSS(String rss) {

```

```

        try {
            XmlPullParserFactory factory =
                XmlPullParserFactory.newInstance();
            XmlPullParser xpp = factory.newPullParser();

            xpp.setInput(new StringReader(rss));
            int eventType = xpp.getEventType();
            String title = null;
            String date = null;
            boolean isFirstTitle = true;
            while (eventType != XmlPullParser.END_DOCUMENT) {
                if (eventType == XmlPullParser.START_TAG) {
                    if (xpp.getName().equalsIgnoreCase("title")) {
                        eventType = xpp.next();
                        if (isFirstTitle) {
                            isFirstTitle = false;
                        } else {
                            title = xpp.getText();
                        }
                    } else if (xpp.getName().equalsIgnoreCase("pubDate")) {
                        eventType = xpp.next();
                        date = xpp.getText().replaceFirst("\\\\+\\d{4}$", "");
                    }
                } else if (eventType == XmlPullParser.END_TAG) {
                    if (xpp.getName().equalsIgnoreCase("item")) {
                        if (title != null && date != null) {
                            news.add(title + "\\n" + date);
                            title = null;
                            date = null;
                        }
                    }
                }
                eventType = xpp.next();
            }

            requireActivity().runOnUiThread(() ->
                adapter.notifyDataSetChanged());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void showInfoDialog() {
        AlertDialog.Builder builder = new
        AlertDialog.Builder(requireContext());
        builder.setTitle("iXBT.com: Носители информации");
        builder.setMessage("iXBT.com (https://www.ixbt.com) --
        специализированный российский информационно-аналитический сервер, освещающий
        вопросы аппаратного обеспечения персональных компьютеров, коммуникаций и
        серверов, 3D-графики и звука, цифрового фото и видео, Hi-Fi аппаратуры и
        проекторов, мобильной связи и периферии, игровых приложений и многого
        другого.");
        builder.setPositiveButton("OK", (dialog, which) -> {
            dialog.dismiss();
        });
        builder.show();
    }
}

```

```
package com.example.diskwizard.presentation.disk.add;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.Rect;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.Toast;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.example.diskwizard.MainActivity;
import com.example.diskwizard.R;
import com.example.diskwizard.databinding.ActivityDiskAddBinding;
import com.example.diskwizard.domain.model.Disk;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

import java.io.ByteArrayOutputStream;

public class DiskADD extends AppCompatActivity {
    ActivityDiskAddBinding binding;
    StorageReference Storage;
    DatabaseReference db;
    Uri localImgURI;

    boolean diskEmpty = true;
    View backGroundView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SharedPreferences prefs = getSharedPreferences("APP_PREFERENCES",
MODE_PRIVATE);
        String themeName = prefs.getString("THEME", "Base.Theme.DiskWizard");

        // Устанавливаем тему
        switch (themeName) {
            case "Base.Theme.DiskWizard.Green":
                setTheme(R.style.Base_Theme_DiskWizard_Green);
                break;
            case "Base.Theme.DiskWizard.DeepPurple":
                setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);
                break;
            case "Base.Theme.DiskWizard.LightBlue":
                setTheme(R.style.Base_Theme_DiskWizard_LightBlue);
                break;
        }
    }
}
```

```

        break;
    case "Base.Theme.DiskWizard.Orange":
        setTheme(R.style.Base_Theme_DiskWizard_Orange);
        break;
    default:
        setTheme(R.style.Base_Theme_DiskWizard);
        break;
}

super.onCreate(savedInstanceState);

binding = ActivityDiskAddBinding.inflate(getLayoutInflater());
View view = binding.getRoot();
setContentView(view); // Set the content view
backgroundView = binding.mainbackground;

// УСТАНАВЛИВАЕМ ФОН
switch (themeName) {
    case "Base.Theme.DiskWizard.Green":

backgroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
        break;
    case "Base.Theme.DiskWizard.DeepPurple":

backgroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
        break;
    case "Base.Theme.DiskWizard.LightBlue":

backgroundView.setBackgroundResource(R.drawable.backgroundskies);
        break;
    case "Base.Theme.DiskWizard.Orange":

backgroundView.setBackgroundResource(R.drawable.backgroundorange);
        break;
    default:
        backgroundView.setBackgroundResource(R.drawable.background);
        break;
}

Button buttonBack = binding.backToListDisk;
Button buttonAdd = binding.addDiskButton;

db = FirebaseDatabase.getInstance().getReference();
Storage = FirebaseStorage.getInstance().getReference();

buttonBack.setOnClickListener(view1 -> goToDiskList());
buttonAdd.setOnClickListener(view1 -> onAddDiskClick());
RelativeLayout relay = binding.relmain;

relay.getViewTreeObserver().addOnGlobalLayoutListener(() -> {
    Rect r = new Rect();
    relay.getWindowVisibleDisplayFrame(r);
    int screenHeight = relay.getRootView().getHeight();

    int keypadHeight = screenHeight - r.bottom;

    if (keypadHeight > screenHeight * 0.15) {
        hideViews();
    } else {
        showViews();
    }
});

```

```

    }
    });

    FloatingActionButton addImg = binding.setDiskImg;
    addImg.setOnClickListener(v -> chooseImage());

}

private void onAddDiskClick() {
    String diskName = binding.diskName.getText().toString();
    String smallDescription =
binding.SmallDescriptionText.getText().toString();
    String description = binding.DescriptionText.getText().toString();

    if (diskName.isEmpty() || smallDescription.isEmpty() ||
description.isEmpty() || diskEmpty) {
        // Поля пустые, показать ошибку
        Toast.makeText(this, "Заполните все поля и выберите изображение",
Toast.LENGTH_SHORT).show();
    } else {
        // Поля заполнены, добавить в Firebase
        DatabaseReference disksRef = db.child("Disks");

        String id = disksRef.push().getKey(); // Генерация уникального
ключа
        Disk disk = new Disk(id, diskName, smallDescription,
description);
        disksRef.child(id).setValue(disk);

        // Сохранить изображение в Storage
        uploadImage(localImgURI, id);
    }
}

private void goToDiskList() {
    Intent intent = new Intent(this, MainActivity.class);
    intent.putExtra("fragmentToLoad", "search");
    startActivity(intent);
    finish();
}

private void exit() {
    Intent intent = new Intent(this, MainActivity.class);
    intent.putExtra("fragmentToLoad", "search");
    intent.putExtra("toast", "Диск успешно добавлен");
    startActivity(intent);
    finish();
}

private void hideViews() {
    // Скрыть нужные вам элементы интерфейса
    binding.backToListDisk.setVisibility(View.GONE);
    binding.addDiskButton.setVisibility(View.GONE);
}

private void showViews() {
    // Показать скрытые элементы интерфейса
    binding.backToListDisk.setVisibility(View.VISIBLE);
    binding.addDiskButton.setVisibility(View.VISIBLE);
}

```

```

// Обработка изображения

// Создайте экземпляр ActivityResultLauncher
ActivityResultLauncher<Intent> mGetContent = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == Activity.RESULT_OK) {
            Intent data = result.getData();
            localImgURI = data.getData();
            if (localImgURI == null) {
                Bundle extras = data.getExtras();
                Bitmap imageBitmap = (Bitmap) extras.get("data");
                // Convert bitmap to Uri
                localImgURI = getImageUri(this, imageBitmap);
            }
            diskEmpty = false;
            Glide.with(this)
                .load(localImgURI)
                .apply(new RequestOptions().centerCrop())
                .into(binding.imageView);
        }
    }
);

private void chooseImage() {
    Intent galleryIntent = new Intent(Intent.ACTION_PICK,
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    Intent chooser = Intent.createChooser(galleryIntent, "Выберите приложение");
    chooser.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[] {
        cameraIntent });

    // Запустите ActivityResultLauncher
    mGetContent.launch(chooser);
}

private Uri getImageUri(Context inContext, Bitmap inImage) {
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
    String path =
        MediaStore.Images.Media.insertImage(inContext.getContentResolver(), inImage,
        "Title", null);
    return Uri.parse(path);
}

private void uploadImage(Uri imageUri, String diskId) {
    StorageReference fileRef = Storage.child("Disks/"+diskId+".jpg");

    fileRef.putFile(imageUri)
        .addOnSuccessListener(taskSnapshot -> {
            Toast.makeText(this, "Image uploaded",
                Toast.LENGTH_SHORT).show();
            exit();
        })
        .addOnFailureListener(e -> {

```



```

        Toast.makeText(this, "Error uploading image",
            Toast.LENGTH_SHORT).show();
    });
}
}

```

Приложение 19 - DiskDetails.java

```

package com.example.diskwizard.presentation.disk.details;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.bumptech.glide.request.RequestOptions;
import com.example.diskwizard.MainActivity;
import com.example.diskwizard.R;
import com.example.diskwizard.databinding.ActivityDiskDetailsBinding;
import com.example.diskwizard.databinding.FragmentCommentItemBinding;
import com.example.diskwizard.domain.model.Comment;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Objects;

public class DiskDetails extends AppCompatActivity {

    ActivityDiskDetailsBinding binding;
    DatabaseReference db = FirebaseDatabase.getInstance().getReference();
    StorageReference storageReference =
        FirebaseStorage.getInstance().getReference();
    String diskId;
    View backGroundView;
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    LinearLayout commentList;

    @Override

```

```

        protected void onCreate(Bundle savedInstanceState) {
            SharedPreferences prefs = getSharedPreferences("APP_PREFERENCES",
MODE_PRIVATE);
            String themeName = prefs.getString("THEME", "Base.Theme.DiskWizard");

            // Устанавливаем тему
            switch (themeName) {
                case "Base.Theme.DiskWizard.Green":
                    setTheme(R.style.Base_Theme_DiskWizard_Green);
                    break;
                case "Base.Theme.DiskWizard.DeepPurple":
                    setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);
                    break;
                case "Base.Theme.DiskWizard.LightBlue":
                    setTheme(R.style.Base_Theme_DiskWizard_LightBlue);
                    break;
                case "Base.Theme.DiskWizard.Orange":
                    setTheme(R.style.Base_Theme_DiskWizard_Orange);
                    break;
                default:
                    setTheme(R.style.Base_Theme_DiskWizard);
                    break;
            }

            super.onCreate(savedInstanceState);

            binding = ActivityDiskDetailsBinding.inflate(getLayoutInflater());
            View view = binding.getRoot();
            setContentView(view);
            backGroundView = binding.mainbackground;
            commentList = binding.commentList;

            // Устанавливаем тему
            switch (themeName) {
                case "Base.Theme.DiskWizard.Green":

backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
                    break;
                case "Base.Theme.DiskWizard.DeepPurple":

backGroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
                    break;
                case "Base.Theme.DiskWizard.LightBlue":

backGroundView.setBackgroundResource(R.drawable.backgroundskies);
                    break;
                case "Base.Theme.DiskWizard.Orange":

backGroundView.setBackgroundResource(R.drawable.backgroundorange);
                    break;
                default:
                    backGroundView.setBackgroundResource(R.drawable.background);
                    break;
            }

            Intent intent = getIntent();
            diskId = intent.getStringExtra("diskId");
            String diskName = intent.getStringExtra("diskName");
            String diskSmallDescription =
intent.getStringExtra("diskDescription");

```

```

binding.SmallDescriptionText.setText(diskSmallDescription);
binding.tv1.setText(diskName);
binding.addComment.setOnClickListener(view1 -> onAddCommentClick());

StorageReference fileRef =
storageReference.child("Disks/"+diskId+".jpg");
loadImage(fileRef);

db.child("Disks").child(diskId).addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.exists()) {
            String details =
dataSnapshot.child("details").getValue(String.class);
            binding.DescriptionText.setText(details);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Обработка ошибок
    }
});

// Извлекаем данные о комментариях из базы данных и заполняем список
db.child("Comments").addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        List<Comment> comments = new ArrayList<>();
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Comment comment = snapshot.getValue(Comment.class);
            if (comment != null &&
(Objects.equals(comment.getDiskId(), diskId))) {
                comments.add(comment);
            }
        }

        try {
            for (Comment comment : comments) {
                FragmentCommentItemBinding commentBinding =
FragmentCommentItemBinding.inflate(getLayoutInflater());

                TextView nametext = commentBinding.nametext;
                TextView maintext = commentBinding.maintext;
                ImageView avaView = commentBinding.avaView;
                TextView date = commentBinding.datetext;
                Button delbut = commentBinding.delelement;

                nametext.setText(comment.getAuthor());
                maintext.setText(comment.getMaintext());
                date.setText(comment.getDate());

                StorageReference fileRef =
storageReference.child(comment.getUserId() + "/pfp.jpg");

                fileRef.getDownloadUrl().addOnSuccessListener(uri ->
Glide.with(DiskDetails.this)

```

```

        .load(uri)
        .diskCacheStrategy(DiskCacheStrategy.ALL) //
Включаем кэширование

        .apply(new RequestOptions().centerCrop())
        .into(avatarView));

        SharedPreferences sharedPreferences =
DiskDetails.this.getSharedPreferences("MySharedPref",
DiskDetails.this.MODE_PRIVATE);
        boolean isAdmin =
sharedPreferences.getBoolean("isAdmin", false);

        String curUserName = user.getDisplayName();

        if (isAdmin || Objects.equals(comment.getAuthor(),
curUserName)) {
            delbut.setVisibility(View.VISIBLE);
        } else {
            delbut.setVisibility(View.GONE);
        }

        delbut.setOnClickListener(v -> new
AlertDialog.Builder(DiskDetails.this)
            .setTitle("Удалить элемент")
            .setMessage("Вы действительно хотите удалить
отзыв пользователя " + comment.getAuthor() + "?")
            .setPositiveButton("Ok", (dialog, which) -> {
                // Получаем ID диска, который нужно
удалить

                String commentId = comment.getId();

                // Удаляем диск из Firebase
                DatabaseReference diskRef =
db.child("Comments").child(commentId);
                diskRef.removeValue();

                // Удаляем диск из списка и обновляем
адаптер

                int position = comments.indexOf(comment);
                comments.remove(position);
                commentList.removeViewAt(position);
            })
            .setNegativeButton("Отмена", null)
            .show());

        binding.commentList.addView(commentBinding.getRoot());
    }
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

@Override
public void onCancelled(@NonNull DatabaseError error) {
}
});

binding.backtomain.setOnClickListener(view1 -> exit());

```

```

    }

    private void onAddCommentClick() {
        String author = user.getDisplayName();
        String userId = user.getId();
        String maintext = binding.newCommentText.getText().toString();
        Date currentDate = new Date();
        String date = formatDate(currentDate);

        if (maintext.isEmpty()) {
            // Поля пустые, показать ошибку
            Toast.makeText(this, "Нельзя оставить пустой отзыв",
Toast.LENGTH_SHORT).show();
        } else {
            // Поля заполнены, добавить в Firebase
            DatabaseReference commentsRef = db.child("Comments");

            String id = commentsRef.push().getKey(); // Генерация уникального
ключа
            Comment comment = new Comment(id, diskId, userId, author,
maintext, date);
            commentsRef.child(id).setValue(comment);

            // Создаем новое представление для комментария и добавляем его в
LinearLayout
            FragmentCommentItemBinding commentBinding =
            FragmentCommentItemBinding.inflate(getLayoutInflater());
            commentBinding.nametext.setText(comment.getAuthor());
            commentBinding.maintext.setText(comment.getMaintext());
            commentBinding.datetext.setText(comment.getDate());

            // Загрузка изображения
            StorageReference fileRef =
storageReference.child(comment.getUserId() + "/pfp.jpg");
            fileRef.getDownloadUrl().addOnSuccessListener(uri ->
Glide.with(DiskDetails.this)
                .load(uri)
                .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
                .apply(new RequestOptions().centerCrop())
                .into(commentBinding.imageView));

            commentList.addView(commentBinding.getRoot());
        }
    }

    public static String formatDate(Date date) {
        SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy",
Locale.getDefault());
        return sdf.format(date);
    }

    private void exit() {
        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra("fragmentToLoad", "search");
        startActivity(intent);
        finish();
    }

    private void loadImage(StorageReference fileRef) {

```

```

        fileRef.getDownloadUrl().addOnSuccessListener(uri -> Glide.with(this)
            .load(uri)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
            .apply(new RequestOptions().centerCrop())
            .into(binding.imageView));
    }
}

```

Приложение 20 - DiskAdapter.java

```

package com.example.diskwizard.presentation.disk.list;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ImageView;
import android.app.AlertDialog;
import android.graphics.drawable.Drawable;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.bumptech.glide.request.RequestOptions;
import com.bumptech.glide.request.target.CustomTarget;
import com.bumptech.glide.request.transition.Transition;
import com.example.diskwizard.R;
import com.example.diskwizard.databinding.FragmentDiskListItemBinding;
import com.example.diskwizard.domain.model.Disk;
import com.example.diskwizard.domain.service.disk.FirebaseDiskService;
import com.example.diskwizard.presentation.disk.details.DiskDetails;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;

public class DiskAdapter extends ArrayAdapter<Disk> {

    private int layout;
    private List<Disk> disks;
    private LayoutInflater inflater;
    private FirebaseDiskService diskService;

    private Activity activity;

```

```

public DiskAdapter(Activity activity, int resource, List<Disk> disks) {
    super(activity, resource, disks);
    this.disks = disks;
    this.layout = resource;
    this.inflater = LayoutInflater.from(activity);
    this.diskService = new FirebaseDiskService();
    this.activity = activity;
}

// Создаем класс ViewHolder
private class ViewHolder {
    ImageView image;
    Button delbut;
    TextView title;
    TextView description;
}

@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
    ViewHolder viewHolder;
    FragmentDiskListItemBinding binding;

    if (convertView == null) {
        binding = FragmentDiskListItemBinding.inflate(inflater, parent,
false);
        convertView = binding.getRoot();
        viewHolder = new ViewHolder();
        viewHolder.image = binding.imageView;
        viewHolder.delbut = binding.delelement;
        viewHolder.title = binding.title;
        viewHolder.description = binding.description;
        convertView.setTag(viewHolder);
    } else {
        viewHolder = (ViewHolder) convertView.getTag();
    }

    Disk disk = disks.get(position);

    viewHolder.title.setText(disk.getName());
    viewHolder.description.setText(disk.getDescription());

    StorageReference imageRef = diskService.getImage(disk.getId());

    File file = new File(getContext().getFilesDir(), disk.getId() +
".png");
    if (file.exists()) {
        // Если файл существует, загружаем его
        Glide.with(getContext())
            .load(file)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(viewHolder.image);
    } else {
        // Если файла нет, загружаем изображение из Firebase и сохраняем
его
        imageRef.getDownloadUrl().addOnSuccessListener(uri -> {

```

```

        if (!activity.isDestroyed()) {
            Glide.with(getContext())
                .asDrawable()
                .load(uri)
                .diskCacheStrategy(DiskCacheStrategy.ALL) //
Включаем кэширование
                .apply(new RequestOptions().centerCrop())
                .into(new CustomTarget<Drawable>() {
                    @Override
                    public void onResourceReady(@NonNull Drawable
resource, @Nullable Transition<? super Drawable> transition) {
viewHolder.image.setImageDrawable(resource);

// Сохраняем изображение локально
try {
// Открываем файловый поток и
сохраняем изображение
FileOutputStream fos =
getContext().openFileOutput(disk.getId() + ".png", Context.MODE_PRIVATE);
Bitmap bitmap = ((BitmapDrawable)
resource).getBitmap();

bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);
fos.close();
} catch (IOException e) {
e.printStackTrace();
}

@Override
public void onLoadCleared(@Nullable Drawable
placeholder) {
// Очистка ресурсов, если загрузка была
отменена
}

});
    }

// Применяем анимацию к convertView
Animation animation = AnimationUtils.loadAnimation(getContext(),
R.anim.translate);
convertView.startAnimation(animation);

SharedPreferences sharedPreferences =
getContext().getSharedPreferences("MySharedPref", getContext().MODE_PRIVATE);
boolean isAdmin = sharedPreferences.getBoolean("isAdmin", false);

if (isAdmin) {
    viewHolder.delbut.setVisibility(View.VISIBLE);
} else {
    viewHolder.delbut.setVisibility(View.GONE);
}

viewHolder.delbut.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
    .setTitle("Удалить элемент")
    .setMessage("Вы действительно хотите удалить диск, " +

```



```

disk.getName() + "?")
    .setPositiveButton("Ok", (dialog, which) -> {
        // Получаем ID диска, который нужно удалить
        String diskId = disk.getId();

        // Удаляем диск из Firebase
        DatabaseReference diskRef =
FirebaseDatabase.getInstance().getReference().child("Disks").child(diskId);
        StorageReference StorageRef =
FirebaseStorage.getInstance().getReference().child("Disks").child(diskId+".jpg");

        diskRef.removeValue();
        StorageRef.delete();

        // Удаляем диск из списка и обновляем адаптер
        disks.remove(position);
        notifyDataSetChanged();
    })
    .setNegativeButton("Отмена", null)
    .show());

convertView.setOnClickListener(v -> {
    Intent intent = new Intent(getContext(), DiskDetails.class);
    intent.putExtra("diskId", disk.getId());
    intent.putExtra("diskName", disk.getName());
    intent.putExtra("diskDescription", disk.getDescription());
    getContext().startActivity(intent);
    activity.finish();
});

return convertView;
}

}

```

Приложение 21 - DiskListFraagment.java

```

package com.example.diskwizard.presentation.disk.list;

import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListView;
import androidx.fragment.app.Fragment;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import android.content.SharedPreferences;
import com.example.diskwizard.databinding.FragmentDiskListBinding;
import com.example.diskwizard.R;
import com.example.diskwizard.domain.model.Disk;
import com.example.diskwizard.presentation.disk.add.DiskADD;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

```

```

import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.List;

public class DiskListFragment extends Fragment {
    public FragmentDiskListBinding binding;
    FloatingActionButton buttonAdd;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {
        binding = FragmentDiskListBinding.inflate(inflater, container,
false);
        View view = binding.getRoot();

        ListView listView = binding.diskList;
        buttonAdd = binding.addAdminButton;

        buttonAdd.setOnClickListener(view1 -> goToDiskAdding());

        SharedPreferences sharedPreferences =
getActivity().getSharedPreferences("MySharedPref",
getActivity().MODE_PRIVATE);
        boolean isAdmin = sharedPreferences.getBoolean("isAdmin", false);

        if (isAdmin) {
            buttonAdd.setVisibility(View.VISIBLE);
        } else {
            buttonAdd.setVisibility(View.GONE);
        }

        DatabaseReference disksRef =
FirebaseDatabase.getInstance().getReference().child("Disks");

        binding.searchtext.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
                // Ничего не делать
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before,
int count) {
                // Ничего не делать
            }

            @Override
            public void afterTextChanged(Editable s) {
                String searchText = s.toString();
                disksRef.addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                        List<Disk> disks = new ArrayList<>();
                        for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                            Disk disk = snapshot.getValue(Disk.class);

```

```

        if (disk != null) {
            String diskName =
disk.getName().replaceAll("\\s", "").toLowerCase();
            String search =
searchText.replaceAll("\\s", "").toLowerCase();
            if (diskName.contains(search)) {
                disks.add(disk);
            }
        }
        DiskAdapter diskAdapter = new
DiskAdapter(requireActivity(), R.layout.fragment_disk_list_item, disks);

        listView.setAdapter(diskAdapter);
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Обработка ошибок
    }
}

});

});

// Извлекаем данные о дисках из базы данных и заполняем список
disksRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        List<Disk> disks = new ArrayList<>();
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Disk disk = snapshot.getValue(Disk.class);
            if (disk != null) {
                disks.add(disk);
            }
        }
        DiskAdapter diskAdapter = new DiskAdapter(requireActivity(),
R.layout.fragment_disk_list_item, disks);

        listView.setAdapter(diskAdapter);
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Обработка ошибок
    }
}

});

return view;
}

public void goToDiskAdding() {
    Intent intent = new Intent(requireActivity(), DiskADD.class);
    startActivity(intent);
    requireActivity().finish();
}
}

```

Приложение 22 - DiskListItemFragment.java

```

package com.example.diskwizard.presentation.disk.list;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.example.diskwizard.R;

public class DiskListItemFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_disk_list_item, container,
            false);
    }

}

```

Приложение 23 - HomeFragment.java

```

package com.example.diskwizard.presentation.home;

import android.os.Bundle;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import com.example.diskwizard.databinding.FragmentHomeBinding;

public class HomeFragment extends Fragment {
    public FragmentHomeBinding binding;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {
        binding = FragmentHomeBinding.inflate(inflater, container, false);
        View view = binding.getRoot();
        return view;
    }

}

```

Приложение 24 - LoginActivity.java

```

package com.example.diskwizard.presentation.login;

import android.content.Intent;
import android.content.SharedPreferences;

```

```

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.example.diskwizard.MainActivity;
import com.example.diskwizard.R;
import com.example.diskwizard.presentation.registration.RegistrationActivity;
import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;
import com.example.diskwizard.databinding.ActivityLoginBinding;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class LoginActivity extends AppCompatActivity {
    ActivityLoginBinding binding;
    private FirebaseAuth auth = FirebaseAuth.getInstance();
    private TextInputEditText emailField;
    private TextInputEditText passwordField;
    DatabaseReference usersRef =
FirebaseDatabase.getInstance().getReference().child("Users");
    View backGroundView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SharedPreferences prefs = getSharedPreferences("APP_PREFERENCES",
MODE_PRIVATE);
        String themeName = prefs.getString("THEME", "Base.Theme.DiskWizard");

        // При запуске приложения проверяем состояние авторизации
        SharedPreferences sharedPreferences =
getSharedPreferences("MySharedPref", MODE_PRIVATE);
        boolean isLoggedIn = sharedPreferences.getBoolean("isLoggedIn",
false);
        if (isLoggedIn) {
            Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
            startActivity(intent);
            finish();
        }

        // Устанавливаем тему
        switch (themeName) {
            case "Base.Theme.DiskWizard.Green":
                setTheme(R.style.Base_Theme_DiskWizard_Green);
                break;
            case "Base.Theme.DiskWizard.DeepPurple":
                setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);
                break;
            case "Base.Theme.DiskWizard.LightBlue":
                setTheme(R.style.Base_Theme_DiskWizard_LightBlue);
                break;
            case "Base.Theme.DiskWizard.Orange":

```

```

        setTheme(R.style.Base_Theme_DiskWizard_Orange);
        break;
    default:
        setTheme(R.style.Base_Theme_DiskWizard);
        break;
    }

    super.onCreate(savedInstanceState);

    binding = ActivityLoginBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
    backGroundView = binding.mainbackground;

    // Устанавливаем фон
    switch (themeName) {
        case "Base.Theme.DiskWizard.Green":
            backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
            break;
        case "Base.Theme.DiskWizard.DeepPurple":
            backGroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
            break;
        case "Base.Theme.DiskWizard.LightBlue":
            backGroundView.setBackgroundResource(R.drawable.backgroundskies);
            break;
        case "Base.Theme.DiskWizard.Orange":
            backGroundView.setBackgroundResource(R.drawable.backgroundorange);
            break;
        default:
            backGroundView.setBackgroundResource(R.drawable.background);
            break;
    }

    emailField = binding.emailField;
    passwordField = binding.passwordField;

    binding.loginButton.setOnClickListener(view12 -> onLoginClick());

    binding.textView4.setOnClickListener(view1 -> {
        Intent intent = new Intent(LoginActivity.this,
RegistrationActivity.class);
        startActivity(intent);
        finish();
    });

}

private void onLoginClick() {
    String email = emailField.getText().toString();
    String pass = passwordField.getText().toString();

    if (email.isEmpty() || pass.isEmpty()) {
        Toast.makeText(this, "Почта/Пароль не могут быть пустыми",
Toast.LENGTH_LONG).show();
        return;
    }
}

```

```

    }

    String hashedPassword = hashPassword(pass);

    auth.signInWithEmailAndPassword(email, hashedPassword)
        .addOnSuccessListener(authResult -> {
            FirebaseUser user = auth.getCurrentUser();

            usersRef.child(user.getUid()).addListenerForSingleValueEvent(new
            ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot
                dataSnapshot) {
                    if (dataSnapshot.exists()) {
                        boolean isAdmin =
                        dataSnapshot.child("admin").getValue(Boolean.class);

                        // Сохранение значения isAdmin в
                        SharedPreferences
                        SharedPreferences sharedPreferences =
                        getSharedPreferences("MySharedPref", MODE_PRIVATE);
                        SharedPreferences.Editor myEdit =
                        sharedPreferences.edit();

                        myEdit.putBoolean("isAdmin", isAdmin);
                        myEdit.apply();
                    }
                }
                @Override
                public void onCancelled(@NonNull DatabaseError error)
            {
                // Обработка ошибок
            }
        });

        // При успешной авторизации сохраняем состояние
        авторизации
        SharedPreferences sharedPreferences =
        getSharedPreferences("MySharedPref", MODE_PRIVATE);
        SharedPreferences.Editor myEdit =
        sharedPreferences.edit();
        myEdit.putBoolean("isLoggedIn", true);
        myEdit.apply();

        Intent intent = new Intent(LoginActivity.this,
        MainActivity.class);
        startActivity(intent);
        finish();
    })

    .addOnFailureListener(e -> Snackbar.make(binding.textView,
        "Authorisation Error: " + e.getMessage(),
        Snackbar.LENGTH_SHORT).show());
}

// Метод для хеширования пароля с использованием SHA-256
private String hashPassword(String password) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(password.getBytes());
    }

```

```

        StringBuilder hexString = new StringBuilder();

        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }

        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
}
}

```

Приложение 25 - ProfileFragment.java

```

package com.example.diskwizard.presentation.profile;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;
import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.bumptech.glide.request.RequestOptions;
import com.example.diskwizard.MainActivity;
import com.example.diskwizard.R;
import com.example.diskwizard.databinding.FragmentProfileBinding;
import com.example.diskwizard.presentation.login.LoginActivity;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.auth.FirebaseAuth;

```



```

import com.google.firebase.auth.FirebaseAuth;
import java.io.ByteArrayOutputStream;

public class ProfileFragment extends Fragment {

    public FragmentProfileBinding binding;
    private StorageReference storageReference =
FirebaseStorage.getInstance().getReference();
    FirebaseAuth user = FirebaseAuth.getInstance().getCurrentUser();
    String name = user.getDisplayName();

    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return this.name;
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        binding = FragmentProfileBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        binding.userName.setText(name);
        binding.exit.setOnClickListener(view1 -> exit(requireContext()));
        binding.changeAVA.setOnClickListener(view1 -> chooseImage());

        String userId = user.getId();
        StorageReference fileRef = storageReference.child(userId +
"/pfp.jpg");
        loadNewImage(fileRef);

        binding.changePass.setOnClickListener(view1 -> changeAccPass());
        binding.changeName.setOnClickListener(view1 -> changeAccName());

        //Изменение статуса админа
        if (user != null && name != null && name.equals("DmitryAce")) {
            binding.textView31.setVisibility(View.VISIBLE);
            binding.adminNameBlock.setVisibility(View.VISIBLE);
            binding.changeAdmin.setVisibility(View.VISIBLE);
            binding.textView30.setVisibility(View.GONE);
            binding.newName.setVisibility(View.GONE);
            binding.changeName.setVisibility(View.GONE);
        } else {
            binding.textView31.setVisibility(View.GONE);
            binding.adminNameBlock.setVisibility(View.GONE);
            binding.changeAdmin.setVisibility(View.GONE);
            binding.textView30.setVisibility(View.VISIBLE);
            binding.newName.setVisibility(View.VISIBLE);
            binding.changeName.setVisibility(View.VISIBLE);
        }

        binding.changeAdmin.setOnClickListener(view1 -> changeAdminState());

        changeThemeHendler();
        return view;
    }

    private void changeThemeHendler() {

```

```

        Glide.with(this)
            .load(R.drawable.background)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(binding.CardImageView1);

        Glide.with(this)
            .load(R.drawable.backgrounddeepgreen)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(binding.CardImageView2);

        Glide.with(this)
            .load(R.drawable.backgrounddeeppurple)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(binding.CardImageView3);

        Glide.with(this)
            .load(R.drawable.backgroundskies)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(binding.CardImageView4);

        Glide.with(this)
            .load(R.drawable.backgroundorange)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
кэширование
            .apply(new RequestOptions().centerCrop())
            .into(binding.CardImageView5);

        binding.themeCard1.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
            .setTitle("Смена темы")
            .setMessage("После смены темы приложение будет
перезапущено.")
            .setPositiveButton("Ok", (dialog, which) -> {
                // Сохраняем выбранную тему в SharedPreferences
                SharedPreferences.Editor editor =
getActivity().getSharedPreferences("APP_PREFERENCES",
Context.MODE_PRIVATE).edit();
                editor.putString("THEME", "Base_Theme_DiskWizard");
                editor.apply();

                Toast.makeText(getContext(), "Тема изменена",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(getContext(),
LoginActivity.class);
                startActivity(intent);
                getActivity().finish();
            })
            .setNegativeButton("Отмена", null)
            .show());

        binding.themeCard2.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
            .setTitle("Смена темы")

```

```

        .setMessage("После смены темы приложение будет
перезапущено.")
        .setPositiveButton("Ok", (dialog, which) -> {
            // Сохраняем выбранную тему в SharedPreferences
            SharedPreferences.Editor editor =
getActivity().getSharedPreferences("APP_PREFERENCES",
Context.MODE_PRIVATE).edit();
            editor.putString("THEME", "Base.Theme.DiskWizard.Green");
            editor.apply();

            Toast.makeText(getContext(), "Тема изменена",
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(getContext(),
LoginActivity.class);
            startActivity(intent);
            getActivity().finish();
        })
        .setNegativeButton("Отмена", null)
        .show());

        binding.themeCard3.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
            .setTitle("Смена темы")
            .setMessage("После смены темы приложение будет
перезапущено.")
            .setPositiveButton("Ok", (dialog, which) -> {
                // Сохраняем выбранную тему в SharedPreferences
                SharedPreferences.Editor editor =
getActivity().getSharedPreferences("APP_PREFERENCES",
Context.MODE_PRIVATE).edit();
                editor.putString("THEME",
"Base.Theme.DiskWizard.DeepPurple");
                editor.apply();

                Toast.makeText(getContext(), "Тема изменена",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(getContext(),
LoginActivity.class);
                startActivity(intent);
                getActivity().finish();
            })
            .setNegativeButton("Отмена", null)
            .show());

        binding.themeCard4.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
            .setTitle("Смена темы")
            .setMessage("После смены темы приложение будет
перезапущено.")
            .setPositiveButton("Ok", (dialog, which) -> {
                // Сохраняем выбранную тему в SharedPreferences
                SharedPreferences.Editor editor =
getActivity().getSharedPreferences("APP_PREFERENCES",
Context.MODE_PRIVATE).edit();
                editor.putString("THEME",
"Base.Theme.DiskWizard.LightBlue");
                editor.apply();

                Toast.makeText(getContext(), "Тема изменена",
Toast.LENGTH_SHORT).show();

```

```

        Intent intent = new Intent(getContext(),
LoginActivity.class);
        startActivity(intent);
        getActivity().finish();
    })
    .setNegativeButton("Отмена", null)
    .show());

    binding.themeCard5.setOnClickListener(v -> new
AlertDialog.Builder(getContext())
    .setTitle("Смена темы")
    .setMessage("После смены темы приложение будет
перезапущено.")
    .setPositiveButton("Ok", (dialog, which) -> {
        // Сохраняем выбранную тему в SharedPreferences
        SharedPreferences.Editor editor =
getActivity().getSharedPreferences("APP_PREFERENCES",
Context.MODE_PRIVATE).edit();
        editor.putString("THEME",
"Base.Theme.DiskWizard.Orange");
        editor.apply();

        Toast.makeText(getContext(), "Тема изменена",
Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(getContext(),
LoginActivity.class);
        startActivity(intent);
        getActivity().finish();
    })
    .setNegativeButton("Отмена", null)
    .show());
}

public void exit(Context context) {
    FirebaseAuth.getInstance().signOut();
    Intent intent = new Intent(context, LoginActivity.class);
    SharedPreferences sharedPreferences =
context.getSharedPreferences("MySharedPref", Context.MODE_PRIVATE);
    SharedPreferences.Editor myEdit = sharedPreferences.edit();
    myEdit.putBoolean("isLoggedIn", false);
    myEdit.apply();
    startActivity(intent);
    requireActivity().finish();
}

// IMAGE
// Создадим экземпляр ActivityResultLauncher
ActivityResultLauncher<Intent> mGetContent = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == Activity.RESULT_OK) {
            Intent data = result.getData();
            Uri imageUri = data.getData();
            if (imageUri != null) {
                String userId =
FirebaseAuth.getInstance().getCurrentUser().getUid();
                uploadImage(imageUri, userId);
            } else {
                Bundle extras = data.getExtras();

```

```

        Bitmap imageBitmap = (Bitmap) extras.get("data");
        // Convert bitmap to Uri
        imageUri = getImageUri(getContext(), imageBitmap);
        String userId =
        FirebaseAuth.getInstance().getCurrentUser().getUid();
        uploadImage(imageUri, userId);
    }
}

);

private void chooseImage() {
    Intent galleryIntent = new Intent(Intent.ACTION_PICK,
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    Intent chooser = Intent.createChooser(galleryIntent, "Выберите
приложение");
    chooser.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[] {
    cameraIntent });

    // Запустим ActivityResultLauncher
    mGetContent.launch(chooser);
}

private Uri getImageUri(Context inContext, Bitmap inImage) {
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
    String path =
    MediaStore.Images.Media.insertImage(inContext.getContentResolver(), inImage,
    "Title", null);
    return Uri.parse(path);
}

private void uploadImage(Uri imageUri, String userId) {
    StorageReference fileRef = storageReference.child(userId +
    "/pfp.jpg");

    fileRef.putFile(imageUri)
        .addOnSuccessListener(taskSnapshot -> {
            Toast.makeText(getContext(), "Image uploaded",
            Toast.LENGTH_SHORT).show();
            loadNewImage(fileRef);
        })
        .addOnFailureListener(e -> {
            Toast.makeText(getContext(), "Error uploading image",
            Toast.LENGTH_SHORT).show();
        });
}

private void loadNewImage(StorageReference fileRef) {

    fileRef.getDownloadUrl().addOnSuccessListener(uri -> {
        Glide.with(this)
            .load(uri)
            .diskCacheStrategy(DiskCacheStrategy.ALL) // Включаем
            .apply(new RequestOptions().centerCrop())
            .into(binding.imageView);
    });
}

```

кэширование

```

    }

    // ACCOUNT SETTINGS
    public void changeAccPass() {
        final EditText current_password = binding.curPass;
        final EditText new_password = binding.newPass;
        final EditText repeat_password = binding.repPass;

        if (TextUtils.isEmpty(current_password.getText().toString())) {
            Snackbar.make(binding.textView30, "Укажите текущий пароль",
Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(new_password.getText().toString())) {
            Snackbar.make(binding.textView30,
                "Введите новый пароль",
                Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(repeat_password.getText().toString())) {
            Snackbar.make(binding.textView30,
                "Повторите пароль",
                Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (new_password.getText().toString().length() < 5) {
            Snackbar.make(binding.textView30,
                "Пароль должен быть не менее 5 символов",
Snackbar.LENGTH_SHORT).show();
            return;
        }
        if
(!new_password.getText().toString().equals(repeat_password.getText().toString
())) {
            Snackbar.make(binding.textView30,
                "Пароли не совпадают",
                Snackbar.LENGTH_SHORT).show();
            return;
        }

        // Проверка правильности старого пароля
        AuthCredential credential =
EmailAuthProvider.getCredential(user.getEmail(),
current_password.getText().toString());
        user.reauthenticate(credential)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    // Пароль успешно подтвержден, теперь можно изменить
                    пароль
                    changeUserPassword(name,
new_password.getText().toString());
                } else {
                    Snackbar.make(binding.textView30,
                        "Текущий пароль неверный",
                        Snackbar.LENGTH_SHORT).show();
                }
            });
    }

    // Метод для изменения пароля пользователя в Firebase

```

```

        private void changeUserPassword(String name, String newPassword) {
FirebaseAuth.getInstance().getCurrentUser().updatePassword(newPassword)
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // Пароль успешно изменен в Firebase Authentication,
                обновляем в Realtime Database
                DatabaseReference usersRef =
                FirebaseDatabase.getInstance().getReference().child("Users");

                // Находим пользователя с заданным именем в базе
                данных и обновляем его пароль

                usersRef.orderByChild("name").equalTo(name).addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
dataSnapshot) {
                        for (DataSnapshot userSnapshot :
dataSnapshot.getChildren()) {
                            // Обновляем поле пароля для пользователя

                            userSnapshot.getRef().child("pass").setValue(newPassword);
                        }
                        Snackbar.make(binding.textView30, "Пароль
успешно изменен", Snackbar.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onCancelled(DatabaseError
databaseError) {
                        Snackbar.make(binding.textView30, "Ошибка при
изменении пароля в Realtime Database: " + databaseError.getMessage(),
                        Snackbar.LENGTH_SHORT).show();
                    }
                });
            } else {
                Snackbar.make(binding.textView30, "Ошибка при
изменении пароля в Firebase Authentication: " +
                task.getException().getMessage(), Snackbar.LENGTH_SHORT).show();
            }
        });
    }

    public void changeAccName() {
        final EditText newName = binding.newName;

        if (TextUtils.isEmpty(newName.getText().toString())) {
            Snackbar.make(binding.textView30, "Введите новое имя",
            Snackbar.LENGTH_SHORT).show();
            return;
        }

        if (user != null) {
            UserProfileChangeRequest profileUpdates = new
            UserProfileChangeRequest.Builder()
                .setDisplayName(newName.getText().toString())
                .build();

            user.updateProfile(profileUpdates)

```

```

        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // Имя пользователя успешно изменено в Firebase
                Authentication и отображается,
                // теперь обновляем в Realtime Database
                DatabaseReference usersRef =
                FirebaseDatabase.getInstance().getReference().child("Users");

                usersRef.orderByChild("email").equalTo(user.getEmail()).addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
                    dataSnapshot) {
                        for (DataSnapshot userSnapshot :
                        dataSnapshot.getChildren()) {
                            // Обновляем поле имени пользователя
                            в Realtime Database

                            userSnapshot.getRef().child("name").setValue(newName.getText().toString());
                        }
                        Snackbar.make(binding.textView30, "Имя
                        пользователя успешно изменено", Snackbar.LENGTH_SHORT).show();

                        binding.UserName.setText(newName.getText().toString());
                        setName(newName.getText().toString());
                    }

                    @Override
                    public void onCancelled(DatabaseError
                    databaseError) {
                        Snackbar.make(binding.textView30, "Ошибка
                        при изменении имени пользователя в Realtime Database: " +
                        databaseError.getMessage(), Snackbar.LENGTH_SHORT).show();
                    }
                });
            } else {
                Snackbar.make(binding.textView30, "Ошибка при
                изменении имени пользователя: " + task.getException().getMessage(),
                Snackbar.LENGTH_SHORT).show();
            }
        });
    } else {
        Snackbar.make(binding.textView30, "Пользователь не авторизован",
        Snackbar.LENGTH_SHORT).show();
    }
}

// Change AdminState
public void changeAdminState() {
    final EditText nameEditText = binding.adminName;
    final String adminName = nameEditText.getText().toString().trim();

    if (TextUtils.isEmpty(adminName)) {
        Snackbar.make(binding.textView30, "Введите имя",
        Snackbar.LENGTH_SHORT).show();
        return;
    }

    DatabaseReference usersRef =
    FirebaseDatabase.getInstance().getReference().child("Users");

```



```

        usersRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    String name =
snapshot.child("name").getValue(String.class);
                    if (name != null &&
name.equals(nameEditText.getText().toString())) {
                        boolean adminStatus =
snapshot.child("admin").getValue(Boolean.class);

snapshot.getRef().child("admin").setValue(!adminStatus);

                        SharedPreferences sharedPreferences =
getActivity().getSharedPreferences("MySharedPref",
getActivity().MODE_PRIVATE);
                        SharedPreferences.Editor myEdit =
sharedPreferences.edit();
                        myEdit.putBoolean("isAdmin", !adminStatus);

                        myEdit.apply();
                        Snackbar.make(binding.textView30, "Статус изменен",
Snackbar.LENGTH_SHORT).show();
                        return;
                    }
                }
                Snackbar.make(binding.textView30, "Пользователь не найден",
Snackbar.LENGTH_SHORT).show();
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Обработка ошибок
            }
        });
    }
}

```

Приложение 26 - RegistrationActivity.java

```

package com.example.diskwizard.presentation.registration;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
import com.example.diskwizard.R;
import com.example.diskwizard.domain.model.User;
import com.example.diskwizard.presentation.login.LoginActivity;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.FirebaseAuth;
import com.example.diskwizard.databinding.ActivityRegistrationBinding;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.security.MessageDigest;

```

```

import java.security.NoSuchAlgorithmException;

public class RegistrationActivity extends AppCompatActivity {
    ActivityRegistrationBinding binding;
    FirebaseAuth auth = FirebaseAuth.getInstance();
    FirebaseDatabase db = FirebaseDatabase.getInstance();

    DatabaseReference users;
    View backGroundView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SharedPreferences prefs = getSharedPreferences("APP_PREFERENCES",
MODE_PRIVATE);
        String themeName = prefs.getString("THEME", "Base.Theme.DiskWizard");

        // Устанавливаем тему
        switch (themeName) {
            case "Base.Theme.DiskWizard.Green":
                setTheme(R.style.Base_Theme_DiskWizard_Green);
                break;
            case "Base.Theme.DiskWizard.DeepPurple":
                setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);
                break;
            case "Base.Theme.DiskWizard.LightBlue":
                setTheme(R.style.Base_Theme_DiskWizard_LightBlue);
                break;
            case "Base.Theme.DiskWizard.Orange":
                setTheme(R.style.Base_Theme_DiskWizard_Orange);
                break;
            default:
                setTheme(R.style.Base_Theme_DiskWizard);
                break;
        }

        super.onCreate(savedInstanceState);

        binding = ActivityRegistrationBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);
        backGroundView = binding.mainbackground;

        // Устанавливаем тему
        switch (themeName) {
            case "Base.Theme.DiskWizard.Green":
                backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
                break;
            case "Base.Theme.DiskWizard.DeepPurple":
                backGroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
                break;
            case "Base.Theme.DiskWizard.LightBlue":
                backGroundView.setBackgroundResource(R.drawable.backgroundskies);
                break;
            case "Base.Theme.DiskWizard.Orange":
                backGroundView.setBackgroundResource(R.drawable.backgroundorange);
                break;
        }
    }
}

```

```

        default:
            backgroundImage.setBackgroundResource(R.drawable.background);
            break;
    }

    db = FirebaseDatabase.getInstance();
    users = db.getReference("Users");

    binding.regButton.setOnClickListener(view1 -> onRegistrationClick());

    binding.textView4.setOnClickListener(view12 -> {
        Intent intent = new Intent(RegistrationActivity.this,
LoginActivity.class);
        startActivity(intent);
        finish();
    });
}

private void onRegistrationClick() {
    final EditText email = binding.emailField;
    final EditText name = binding.nameField;
    final EditText pass = binding.passwordField;
    final EditText repeatpass = binding.repeatpasswordField;

    if (TextUtils.isEmpty(email.getText().toString())) {
        Snackbar.make(binding.textView4, "Enter your e-mail address",
Snackbar.LENGTH_SHORT).show();
        return;
    }
    if (TextUtils.isEmpty(name.getText().toString())) {
        Snackbar.make(binding.textView4,
            "Enter your name",
            Snackbar.LENGTH_SHORT).show();
        return;
    }
    if (pass.getText().toString().length() < 5) {
        Snackbar.make(binding.textView4,
            "Enter the password at least 5 characters",
Snackbar.LENGTH_SHORT).show();
        return;
    }
    if
(!pass.getText().toString().equals(repeatpass.getText().toString())) {
        Snackbar.make(binding.textView4,
            "The passwords don't match",
            Snackbar.LENGTH_SHORT).show();
        return;
    }

    String hashedPassword = hashPassword(pass.getText().toString());

    auth.createUserWithEmailAndPassword(email.getText().toString(),
hashedPassword)
        .addOnSuccessListener(authResult -> {

            User user = new User();
            user.setEmail(email.getText().toString());
            user.setName(name.getText().toString());
            user.setAdmin(false);

```

```

        users.child(auth.getCurrentUser().getUid())
            .setValue(user).addOnSuccessListener(unused -> {
                // Создание объекта UserProfileChangeRequest
                UserProfileChangeRequest profileUpdates = new
                UserProfileChangeRequest.Builder()

                .setDisplayName(name.getText().toString())
                    .build();

                // Обновление профиля пользователя

auth.getCurrentUser().updateProfile(profileUpdates)
                    .addOnCompleteListener(task -> {
                        if (task.isSuccessful()) {

Snackbar.make(binding.textView4, "Успешная регистрация!",
Snackbar.LENGTH_SHORT).show();

                        }
                    });

                Intent intent = new Intent(RegistrationActivity.this,
LoginActivity.class);
                startActivity(intent);
                finish();
            })
            .addOnFailureListener(e -> {
                // Display the error message to the user
                Snackbar.make(binding.textView4, "Registration failed: "
+ e.getMessage(), Snackbar.LENGTH_LONG).show();
            });
    }

    private String hashPassword(String password) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(password.getBytes());
            StringBuilder hexString = new StringBuilder();

            for (byte b : hash) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }

            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            return null;
        }
    }
}

```

Приложение 27 - MainActivity.java

```

package com.example.diskwizard;

import android.content.SharedPreferences;

```

```

import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import com.example.diskwizard.databinding.ActivityMainBinding;
import com.example.diskwizard.presentation.about.AboutFragment;
import com.example.diskwizard.presentation.about.app.AppAboutFragment;
import com.example.diskwizard.presentation.about.developer.DeveloperAboutFragment;
import com.example.diskwizard.presentation.home.HomeFragment;
import com.google.android.material.navigation.NavigationBarView;
import com.example.diskwizard.presentation.disk.list.DiskListFragment;
import com.example.diskwizard.presentation.profile.ProfileFragment;

public class MainActivity extends AppCompatActivity implements
NavigationBarView.OnItemSelectedListener {

    public ActivityMainBinding binding;
    View backgroundColor;

    SharedPreferences prefs;
    String themeName;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);

        prefs = getSharedPreferences("APP_PREFERENCES", MODE_PRIVATE);
        themeName = prefs.getString("THEME", "Base.Theme.DiskWizard");

        NavigationBarView navBar = binding.bottomNavigationView;
        navBar.setOnItemSelectedListener(this);
        backgroundColor = binding.mainbackground;

        String fragmentToLoad = getIntent().getStringExtra("fragmentToLoad");
        if (fragmentToLoad != null && fragmentToLoad.equals("search")) {
            navBar.setSelectedItemId(R.id.search);
        } else {
            navBar.setSelectedItemId(R.id.home);
        }

        String message = getIntent().getStringExtra("toast");
        if (message != null && message.equals("Диск успешно добавлен")) {
            Toast.makeText(this, "Диск успешно добавлен",
                Toast.LENGTH_SHORT).show();
        }

        // Устанавливаем тему
        switch (themeName) {
            case "Base.Theme.DiskWizard.Green":

```

```

        setTheme(R.style.Base_Theme_DiskWizard_Green);

    backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
        break;
    case "Base.Theme.DiskWizard.DeepPurple":
        setTheme(R.style.Base_Theme_DiskWizard_DeepPurple);

    backGroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
        break;
    case "Base.Theme.DiskWizard.LightBlue":
        setTheme(R.style.Base_Theme_DiskWizard_LightBlue);

    backGroundView.setBackgroundResource(R.drawable.backgroundskies);
        break;
    case "Base.Theme.DiskWizard.Orange":
        setTheme(R.style.Base_Theme_DiskWizard_Orange);

    backGroundView.setBackgroundResource(R.drawable.backgroundorange);
        break;
    default:
        setTheme(R.style.Base_Theme_DiskWizard);
        backGroundView.setBackgroundResource(R.drawable.background);
        break;
    }
}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    int itemId = item.getItemId();
    Fragment fragment = new Fragment();

    if (itemId == R.id.home) {
        fragment = new HomeFragment();
    }
    else if (itemId == R.id.search) {
        fragment = new DiskListFragment();
    }
    else if (itemId == R.id.profile) {
        fragment = new ProfileFragment();
    }
    else if (itemId == R.id.info) {
        fragment = new AboutFragment(this::onAppAboutClick,
this::onDeveloperAboutClick);
    }

    getSupportFragmentManager()
        .beginTransaction()
        .setCustomAnimations(R.anim.slide_in, R.anim.slide_out) //
Добавляем анимацию
        .replace(R.id.fragmentView, fragment)
        .setReorderingAllowed(true)
        .addToBackStack("backstack")
        .commit();

    switch (themeName) {
        case "Base.Theme.DiskWizard.Green":
            backGroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
            break;

```

```

        case "Base.Theme.DiskWizard.DeepPurple":

backgroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
        break;
        case "Base.Theme.DiskWizard.LightBlue":

backgroundView.setBackgroundResource(R.drawable.backgroundskies);
        break;
        case "Base.Theme.DiskWizard.Orange":

backgroundView.setBackgroundResource(R.drawable.backgroundorange);
        break;
        default:
            backgroundView.setBackgroundResource(R.drawable.background);
            break;
    }

    return true;
}

private void onAppAboutClick() {
    getFragmentManager()
        .beginTransaction()
        .replace(R.id.fragmentView, new AppAboutFragment())
        .addToBackStack("backstack")
        .commit();
    switch (themeName) {
        case "Base.Theme.DiskWizard.Green":

backgroundView.setBackgroundResource(R.drawable.backgrounddeepgreen);
            break;
        case "Base.Theme.DiskWizard.DeepPurple":

backgroundView.setBackgroundResource(R.drawable.backgrounddeeppurple);
            break;
        case "Base.Theme.DiskWizard.LightBlue":

backgroundView.setBackgroundResource(R.drawable.backgroundskies);
            break;
        case "Base.Theme.DiskWizard.Orange":

backgroundView.setBackgroundResource(R.drawable.backgroundorange);
            break;
        default:
            backgroundView.setBackgroundResource(R.drawable.background);
            break;
    }
}

private void onDeveloperAboutClick() {
    getFragmentManager()
        .beginTransaction()
        .replace(R.id.fragmentView, new DeveloperAboutFragment())
        .addToBackStack("backstack")
        .commit();
    backgroundView.setBackgroundResource(R.drawable.backgroundauthor);
}
}

```

Приложение 28 - side_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="100%"
        android:toXDelta="0"
        android:duration="300"/>
</set>
```

Приложение 29 - side_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="-100%"
        android:duration="300"/>
</set>
```

Приложение 30 - translate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="1000"
        android:fromXDelta="100%"
        android:fromYDelta="0%"
        android:toXDelta="0%"
        android:toYDelta="0%" />
</set>
```

Приложение 31 - button.xml

```
<ripple
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:color="?attr/rippleColor">
    <item>
        <shape android:shape="rectangle">
            <corners android:radius="4dp" /> <!-- Здесь установите желаемый
радиус скругления -->
        </shape>
    </item>
</ripple>
```

Приложение 32 - activity_disk_add.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".presentation.disk.add.DiskADD"
    android:background="@drawable/background"
    android:id="@+id/mainbackground">
```



```

<RelativeLayout
    android:id="@+id/relmain"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="643dp"
        android:layout_alignParentTop="true"
        android:layout_marginTop="0dp"
        android:paddingStart="12dp"
        android:paddingTop="24dp"
        android:paddingEnd="12dp"
        android:paddingBottom="0dp"
        android:scrollbarStyle="outsideInset"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <RelativeLayout
                android:id="@+id/relLayout"
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <com.google.android.material.textfield.TextInputLayout
                    android:id="@+id/textViewName"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_alignParentStart="true"
                    android:layout_marginStart="0dp"
                    android:layout_marginTop="10dp"
                    android:text="Добавить диск"
                    android:textAlignment="center"

                    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/diskName"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Наименование" />
                </com.google.android.material.textfield.TextInputLayout>

                <androidx.cardview.widget.CardView
                    android:id="@+id/cardView"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_below="@+id/textViewName"
                    android:layout_alignParentStart="true"

```

```

        android:layout_marginTop="12dp"
        android:layout_marginEnd="24dp"
        app:cardCornerRadius="12dp"
        app:cardElevation="0dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="156dp"
            android:layout_height="156dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_sample_avatar" />
    </androidx.cardview.widget.CardView>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/setDiskImg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textViewName"
        android:layout_alignParentStart="true"
        android:layout_gravity="end|bottom"
        android:layout_marginStart="120dp"
        android:layout_marginTop="130dp"
        android:src="@drawable/changeavatar" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/SmallDescriptionBlock"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textViewName"
        android:layout_marginStart="6dp"
        android:layout_marginTop="10dp"
        android:layout_toEndOf="@+id/cardView"
        android:text="Добавить диск"
        android:textAlignment="center"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/SmallDescriptionText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Краткое описание" />
    </com.google.android.material.textfield.TextInputLayout>

</RelativeLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/DescriptionBlock"
    android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:layout_below="@+id/relLayout"
        android:layout_alignParentStart="true"
        android:layout_marginTop="10dp"
        android:text="Добавить диск"
        android:textAlignment="center"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/DescriptionText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Описание" />
    </com.google.android.material.textfield.TextInputLayout>
</RelativeLayout>

</ScrollView>

<Button
    android:id="@+id/addDiskButton"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_above="@+id/backToListDisk"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="31dp"
    android:layout_marginEnd="33dp"
    android:layout_marginBottom="6dp"
    android:background="@drawable/button"
    android:onClick="onAddDiskClick"
    android:text="Добавить" />

<Button
    android:id="@+id/backToListDisk"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="31dp"
    android:layout_marginEnd="33dp"
    android:layout_marginBottom="23dp"
    android:background="@drawable/button"
    android:text="Отмена" />
</RelativeLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 33 - activity_disk_details.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

```

```

        android:layout_height="match_parent"
        tools:context=".presentation.disk.add.DiskADD"
        android:background="@drawable/background"
        android:id="@+id/mainbackground">

        <ScrollView
            android:id="@+id/scrollView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="0dp"
            app:layout_constraintTop_toTopOf="parent">

            <androidx.constraintlayout.widget.ConstraintLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:id="@+id/tv1"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="24dp"
                    android:layout_marginTop="6dp"
                    android:layout_marginEnd="24dp"
                    android:text="Название диска"

                    android:textAppearance="@style/TextAppearance.AppCompat.Large"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent" />

                <androidx.cardview.widget.CardView
                    android:id="@+id/cardView"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentStart="true"
                    android:layout_marginStart="24dp"
                    android:layout_marginTop="10dp"
                    app:cardCornerRadius="12dp"
                    app:cardElevation="0dp"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/tv1">

                    <ImageView
                        android:id="@+id/imageView"
                        android:layout_width="156dp"
                        android:layout_height="156dp"
                        app:layout_constraintHorizontal_bias="0.09"
                        app:srcCompat="@drawable/ic_sample_avatar" />

                </androidx.cardview.widget.CardView>

                <TextView
                    android:id="@+id/SmallDescriptionText"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="20dp"
                    android:layout_marginTop="10dp"
                    android:layout_marginEnd="20dp"
                    android:text="Краткое описание"

```

```

        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/cardView"
        app:layout_constraintTop_toBottomOf="@+id/tv1" />

<TextView
    android:id="@+id/DescriptionText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="24dp"
    android:layout_marginTop="20dp"
    android:text="Описание"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cardView" />

<TextView
    android:id="@+id/commentsHeader"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="24dp"
    android:layout_marginTop="36dp"
    android:text="Отзывы пользователей"
    android:textAlignment="textStart"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintTop_toBottomOf="@+id/DescriptionText" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="24dp"
    android:paddingTop="4dp"
    app:layout_constraintTop_toBottomOf="@+id/commentsHeader">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/new_comment_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Отзыв" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/add_comment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="24dp"
    android:layout_marginTop="8dp"
    android:background="@drawable/button"
    android:text="Оставить отзыв"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout" />

<LinearLayout
    android:id="@+id/commentList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="24dp"
    android:layout_marginTop="12dp"
    android:paddingBottom="60dp"
    android:orientation="vertical"
    app:layout_constraintTop_toBottomOf="@+id/add_comment" />

```

```

        </androidx.constraintlayout.widget.ConstraintLayout>

    </ScrollView>

    <Button
        android:id="@+id/backtomain"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="24dp"
        android:layout_marginTop="36dp"
        android:background="@drawable/button"
        android:text="Назад"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginVertical="10dp"
        tools:layout_editor_absoluteX="24dp" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 34 - activity_login.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainbackground"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".presentation.login.LoginActivity"
    android:background="@drawable/background">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout"
        android:layout_width="0dp"
        android:layout_height="62dp"
        android:layout_marginStart="32dp"
        android:layout_marginTop="48dp"
        android:layout_marginEnd="32dp"
        app:boxBackgroundMode="outline"
        app:counterEnabled="false"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:passwordToggleEnabled="false">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/emailField"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Почта"
            android:inputType="text|textWebEmailAddress"
            android:fontFamily="@font/suez_one"/>

    </com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="170dp"

```

```

        android:text="DiskWizard"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:fontFamily="@font/suez_one"/>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="32dp"
    app:boxBackgroundMode="outline"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/passwordField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Пароль"
        android:inputType="text|textPassword"
        android:fontFamily="@font/suez_one"/>
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/loginButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="48dp"
    android:text="Войти"
    android:onClick="onLoginClick"
    app:layout_constraintEnd_toEndOf="@+id/textInputLayout2"
    app:layout_constraintStart_toStartOf="@+id/textInputLayout2"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout2"
    android:background="@drawable/button"/>

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="Создать аккаунт"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/loginButton" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:text="2024"

```

```

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 35 - activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:background="@drawable/background"
    android:id="@+id/mainbackground">

    <FrameLayout
        android:id="@+id/fragmentView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/bottomNavigationView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottomNavigationView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:itemTextColor="@color/white"
        app:itemIconTint="@color/white"
        android:background="@android:color/transparent"
        app:itemRippleColor="@android:color/transparent"
        app:itemBackground="@android:color/transparent"
        app:menu="@menu/bottom_nav_menu"

        style="@style/Widget.MaterialComponents.BottomNavigationView.PrimarySurface"/
    >

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 36 - activity_registration.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

```



```

tools:context=".presentation.registration.RegistrationActivity"
android:background="@drawable/background"
android:id="@+id/mainbackground">

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout"
    android:layout_width="0dp"
    android:layout_height="62dp"
    android:layout_marginStart="32dp"
    android:layout_marginTop="48dp"
    android:layout_marginEnd="32dp"
    app:boxBackgroundMode="outline"
    app:counterEnabled="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:passwordToggleEnabled="false">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/nameField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Имя пользователя"
        android:inputType="text|textWebEmailAddress" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout2"
    android:layout_width="0dp"
    android:layout_height="62dp"
    android:layout_marginStart="32dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="32dp"
    app:boxBackgroundMode="outline"
    app:counterEnabled="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout"
    app:passwordToggleEnabled="false">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/emailField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Почта"
        android:inputType="text|textWebEmailAddress" />
</com.google.android.material.textfield.TextInputLayout>

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="100dp"
    android:text="DiskWizard"
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:fontFamily="@font/suez_one"/>

```

```

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="32dp"
    app:boxBackgroundMode="outline"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout2"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/passwordField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Пароль"
        android:inputType="text|textPassword" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout4"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="32dp"
    app:boxBackgroundMode="outline"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout3"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/repeatpasswordField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Повторите пароль"
        android:inputType="text|textPassword" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/regButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="48dp"
    android:text="Зарегистрироваться"
    android:onClick="onRegistrationClick"
    app:layout_constraintEnd_toEndOf="@+id/textInputLayout4"
    app:layout_constraintStart_toStartOf="@+id/textInputLayout4"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout4"
    android:background="@drawable/button"/>

<TextView
    android:id="@+id/textView4"

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Войти"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/regButton" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:text="2024"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 37 - fragment_about.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="presentation.about.AboutFragment">

    <Button
        android:id="@+id/appAboutButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="12dp"
        android:text="О программе"
        android:onClick="onAppAboutButtonClick"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:background="@drawable/button"/>

    <Button
        android:id="@+id/developerAboutButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="12dp"
        android:text="Об авторе"
        android:onClick="onDeveloperAboutButtonClick"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/appAboutButton"
        android:background="@drawable/button"/>

    <TextView

```

```

        android:id="@+id/textView33"
        android:layout_width="match_parent"
        android:layout_marginStart="12dp"
        android:layout_height="wrap_content"
        android:paddingVertical="10dp"
        android:text="Новости: Носители информации"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintTop_toBottomOf="@+id/developerAboutButton" />

<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@+id/textView33" />

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/moreInfo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_gravity="end|bottom"
    android:src="@drawable/moreinfo"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginRight="20dp"
    android:layout_marginBottom="20dp" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 38 - fragment_app_about.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="presentation.about.app.AppAboutFragment">

    <ScrollView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:paddingStart="12dp"
        android:paddingTop="0dp"
        android:paddingEnd="12dp"
        android:paddingBottom="0dp"
        android:scrollbarStyle="insideOverlay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="12dp"

```

```

        android:orientation="vertical">

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="О программе"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

        <TextView
            android:id="@+id/textView6"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Информационно-справочная система &quot;Жесткие
диски&quot; - это удобное мобильное приложение, созданное для тех, кто ищет
информацию о жестких дисках. С помощью этого приложения пользователи могут
легко получить доступ к обширной базе данных жестких дисков, а также оставить
свой отзыв и поделиться фотографиями." />

        <TextView
            android:id="@+id/textView7"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:text="Функционал:"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

        <TextView
            android:id="@+id/textView8"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="1. Список дисков: Пользователи могут
просматривать список доступных жестких дисков. Для удобства они могут быть
отсортированы по различным критериям, таким как производитель, объем памяти,
тип интерфейса и т.д." />

        <TextView
            android:id="@+id/textView9"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="2. Информация о дисках: Каждый диск имеет свою
собственную страницу с подробной информацией, включая модель, объем памяти,
скорость вращения, интерфейс, размеры и другие технические характеристики."
/>

        <TextView
            android:id="@+id/textView10"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="3. Фотографии: Пользователи могут просматривать
фотографии каждого диска, чтобы получить более наглядное представление о его
внешнем виде и конструкции." />

        <TextView

```

```

        android:id="@+id/textView11"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="4. Отзывы: Пользователи имеют возможность
оставлять свои отзывы о жестких дисках. Они могут делиться своим опытом
использования, оценивать качество продукта и прикреплять фотографии для
наглядности." />

<TextView
    android:id="@+id/textView12"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="5. Поиск: Пользователи могут использовать
функцию поиска для быстрого нахождения конкретного жесткого диска по его
названию или характеристикам." />

<TextView
    android:id="@+id/textView13"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="6. Сравнение: Приложение предоставляет
возможность сравнивать характеристики нескольких жестких дисков одновременно,
что помогает пользователям принять более обоснованное решение при выборе
продукта." />

<TextView
    android:id="@+id/textView14"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="7. Обновления: База данных жестких дисков
регулярно обновляется, чтобы пользователи всегда имели доступ к актуальной
информации о новых моделях и изменениях в существующих." />

<TextView
    android:id="@+id/textView15"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:text="Инструкция пользователю"

    android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

<TextView
    android:id="@+id/textView16"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="В приложении есть 4 пункта меню, доступные
внизу экрана." />

<TextView
    android:id="@+id/textView17"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Главная: Здесь содержится описание принципа

```

```
работы жестких дисков." />
```

```
    <TextView
        android:id="@+id/textView18"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Поиск: Этот раздел содержит список последних
просмотренных дисков или, если таковых нет, список недавно добавленных в
базу. Здесь же содержится строка поиска дисков. При нажатии на любой диск вы
сможете получить о нем информацию, прочитать отзывы и оставить свой
собственный. " />
```

```
    <TextView
        android:id="@+id/textView19"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Профиль: Здесь вы можете изменять настройки
своего аккаунта и тему приложения." />
```

```
    <TextView
        android:id="@+id/textView20"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Инфо: Этот пункт содержит различную информацию
о самом приложении." />
</LinearLayout>
</ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 39 -fragment_comment_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/nametext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        android:layout_marginTop="4dp"
        android:layout_weight="1"
        android:text="SeaGAYte Barracuda"
        android:textAlignment="viewStart"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textSize="16sp"
        app:layout_constraintStart_toEndOf="@+id/cardView2"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/maintext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```

        android:layout_marginStart="60dp"
        android:layout_weight="1"
        android:text="Хорошие винты, покупаю уже 8-й такой. Стоят в NAS Qnap
(да, знаю, что они не для этого), но за несколько лет пользования ни один
блок ни у одного не выпал. Не слишком тихие или шумные, не слишком быстрые
или медленные. Ничего сверхъестественного в них, просто нормальные
универсальные винты."
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textSize="12sp"
        app:layout_constraintStart_toEndOf="@+id/cardView2"
        app:layout_constraintTop_toBottomOf="@+id/nametext" />

<TextView
    android:id="@+id/datetext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="15dp"
    android:layout_marginStart="6dp"
    android:layout_weight="1"
    android:text="15.09.2004"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textSize="12sp"

    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/maintext" />

<Button
    android:id="@+id/delelement"
    android:layout_width="57dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="6dp"
    android:layout_marginTop="6dp"
    android:backgroundTint="@color/del_color"
    android:text="X"
    android:textColor="@color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/maintext" />

<androidx.cardview.widget.CardView
    android:id="@+id/cardView2"
    android:layout_width="48dp"
    android:layout_height="48dp"
    app:cardBackgroundColor="#7E57C2"
    app:cardCornerRadius="100dp"
    android:layout_margin="6dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <ImageView
        android:id="@+id/avaView"
        android:layout_width="48dp"
        android:layout_height="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
</androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 40 - fragment_developer_about.xml


```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="presentation.about.developer.DeveloperAboutFragment">

    <androidx.cardview.widget.CardView
        android:id="@+id/cardView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="128dp"
        android:layout_marginEnd="24dp"
        app:cardCornerRadius="100dp"
        app:cardElevation="0dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="156dp"
            android:layout_height="156dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/senku" />

    </androidx.cardview.widget.CardView>

    <TextView
        android:id="@+id/textView21"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Борзов Д.О."
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cardView" />

    <TextView
        android:id="@+id/textView22"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:text="ИКБО-03-22"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView21" />

    <TextView
        android:id="@+id/textView23"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"

```

```

        android:text="Год разработки: 2024"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.49"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView22" />

<TextView
    android:id="@+id/textView24"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="dmltryace@yandex.ru"
    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.492"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView23"
    android:fontFamily="@font/suez_one"/>
</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 41 - fragment_disk_list.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="12dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="12dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:orientation="horizontal">

            <com.google.android.material.floatingactionbutton.FloatingActionButton
                android:id="@+id/addAdminButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="end|bottom"
                android:src="@drawable/round_add_24"
                android:layout_margin="16dp"
                android:visibility="gone"/>

            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        app:endIconDrawable="@drawable/ic_round_search"
        app:endIconMode="custom"
        app:errorEnabled="false"
        app:hintEnabled="false">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/searchtext"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Поиск..."
            android:inputType="text|textFilter"
            android:singleLine="true" />

    </com.google.android.material.textfield.TextInputLayout>
</LinearLayout>

<ListView
    android:id="@+id/diskList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:divider="@android:color/transparent"
    android:dividerHeight="12dp"
    android:footerDividersEnabled="false"
    android:headerDividersEnabled="false" />

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 42 - fragment_disk_list_item.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="12sp"
        android:text="1 ТБ / 7200 об/мин"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        app:layout_constraintStart_toEndOf="@+id/cardView2"
        app:layout_constraintTop_toBottomOf="@+id/title"
        android:layout_marginStart="6dp"/>

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"

```

```

        android:text="SeaGAYte Barracuda"
        android:textAlignment="viewStart"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textSize="16sp"
        app:layout_constraintStart_toEndOf="@+id/cardView2"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginStart="6dp"/>

<Button
    android:id="@+id/delelement"
    android:layout_width="57dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="6dp"
    android:backgroundTint="@color/del_color"
    android:text="X"
    android:textColor="@color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="50dp"/>

<androidx.cardview.widget.CardView
    android:id="@+id/cardView2"
    android:layout_width="96dp"
    android:layout_height="96dp"
    app:cardBackgroundColor="#7E57C2"
    app:cardCornerRadius="12dp"
    android:layout_margin="6dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="96dp"
        android:layout_height="96dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 43 - fragment_home.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="32dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"

```

```

        android:layout_marginStart="12dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="12dp"
        android:layout_marginBottom="0dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView26"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Жёсткий диск - это устройство хранения данных,
которое использует магнитное запоминание для хранения и извлечения цифровой
информации." />

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="match_parent"
            android:layout_height="160dp"
            android:layout_marginTop="12dp"
            android:layout_marginBottom="36dp"
            android:src="@drawable/diskhard" />

        <TextView
            android:id="@+id/textView32"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="12dp"
            android:text="Вот основные компоненты жёсткого диска:" />

        <TextView
            android:id="@+id/textView33"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="6dp"
            android:text="Пластины"

        android:textAppearance="@style/TextAppearance.AppCompat.Large" />

        <TextView
            android:id="@+id/textView40"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
            android:text="Это круглые диски, покрытые магнитным
материалом. Данные хранятся на этих пластинах в виде магнитных полей." />

        <TextView
            android:id="@+id/textView41"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="6dp"
            android:text="Головки чтения/записи"

        android:textAppearance="@style/TextAppearance.AppCompat.Large" />

        <TextView
            android:id="@+id/textView34"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"

```

```
        android:text="Это устройства, которые перемещаются над  
поверхностью пластин, читая и записывая данные.&#xA;Вал: Это ось, на которой  
вращаются пластины." />
```

```
<TextView  
    android:id="@+id/textView42"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="6dp"  
    android:text="Мотор"
```

```
android:textAppearance="@style/TextAppearance.AppCompat.Large" />
```

```
<TextView  
    android:id="@+id/textView35"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="24dp"  
    android:text="Это устройство, которое вращает  
пластины.&#xA;Электроника: Это системы, которые управляют работой диска и  
обеспечивают интерфейс между диском и компьютером." />
```

```
<TextView  
    android:id="@+id/textView43"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="6dp"  
    android:text="Контроллер"
```

```
android:textAppearance="@style/TextAppearance.AppCompat.Large" />
```

```
<TextView  
    android:id="@+id/textView36"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="24dp"  
    android:text="Это устройство, которое управляет передачей  
данных между компьютером и диском." />
```

```
<TextView  
    android:id="@+id/textView44"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="6dp"  
    android:text="Кэш"
```

```
android:textAppearance="@style/TextAppearance.AppCompat.Large" />
```

```
<TextView  
    android:id="@+id/textView37"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="24dp"  
    android:text="Это небольшой объем быстрой памяти,  
используемый для временного хранения данных, которые часто используются или  
ожидают передачи." />
```

```
<TextView  
    android:id="@+id/textView45"  
    android:layout_width="match_parent"
```

```

        android:layout_height="wrap_content"
        android:layout_marginBottom="6dp"
        android:text="Прошивка"

        android:textAppearance="@style/TextAppearance.AppCompat.Large" />

        <TextView
            android:id="@+id/textView38"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="36dp"
            android:text="Это программное обеспечение, встроенное в
жесткий диск, которое управляет его функциями." />

        <TextView
            android:id="@+id/textView39"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
            android:text="Все эти компоненты работают вместе, чтобы
обеспечить надежное и эффективное хранение данных. Когда компьютеру требуется
прочитать или записать данные, он отправляет команду жесткому диску. Головки
чтения/записи затем перемещаются к нужной позиции на пластине, где они могут
прочитать или записать данные. Кэш используется для ускорения этого процесса,
временно храня данные, которые часто используются или ожидают передачи.
Прошивка управляет всеми этими процессами, обеспечивая, чтобы диск работал
правильно." />

    </LinearLayout>
</ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 44 - fragment_profile.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="presentation.profile.ProfileFragment">

    <ScrollView
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:paddingStart="12dp"
        android:paddingTop="0dp"
        android:paddingEnd="12dp"
        android:paddingBottom="0dp"
        android:scrollbarStyle="outsideInset"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:paddingBottom="12dp"
        android:gravity="center"
        android:orientation="vertical">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/textView25"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:text="Профиль"
                android:textAlignment="center"

                android:textAppearance="@style/TextAppearance.AppCompat.Display1"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <com.google.android.material.floatingactionbutton.FloatingActionButton
                android:id="@+id/exit"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:layout_marginRight="10dp"
                android:layout_alignParentEnd="true"
                android:layout_alignParentTop="true"
                android:layout_gravity="end|bottom"
                android:src="@drawable/exit" />
        </RelativeLayout>

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <androidx.cardview.widget.CardView
                android:id="@+id/cardView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="24dp"
                android:layout_marginTop="24dp"
                android:layout_marginEnd="24dp"
                app:cardCornerRadius="100dp"
                app:cardElevation="0dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                android:layout_centerInParent="true">

                <ImageView
                    android:id="@+id/imageView"
                    android:layout_width="156dp"
                    android:layout_height="156dp"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent"

```



```

        app:srcCompat="@drawable/ic_sample_avatar" />
</androidx.cardview.widget.CardView>

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/changeAVA"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_gravity="end|bottom"
    android:layout_marginEnd="100dp"
    android:src="@drawable/changeavatar" />

</RelativeLayout>

<TextView
    android:id="@+id/UserName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Имя пользователя"
    android:textAlignment="center"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/ChangeThemeTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:text="Сменить тему"
    android:textAlignment="center"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:gravity="center"
    android:orientation="horizontal">

    <androidx.cardview.widget.CardView
        android:id="@+id/themeCard1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="100dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_centerInParent="true"
        android:layout_marginHorizontal="10dp">

```

```

<ImageView
    android:id="@+id/CardimageView1"
    android:layout_width="50dp"
    android:layout_height="50dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/backgrounddeepgreen"/>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:id="@+id/themeCard2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:cardCornerRadius="100dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_centerInParent="true"
    android:layout_marginHorizontal="10dp">

    <ImageView
        android:id="@+id/CardimageView2"
        android:layout_width="50dp"
        android:layout_height="50dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/backgrounddeepgreen"/>
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/themeCard3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="100dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_centerInParent="true"
        android:layout_marginHorizontal="10dp">

        <ImageView
            android:id="@+id/CardimageView3"
            android:layout_width="50dp"
            android:layout_height="50dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/backgrounddeepgreen"/>
        </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/themeCard4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="100dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"

```

```

        android:layout_centerInParent="true"
        android:layout_marginHorizontal="10dp">

        <ImageView
            android:id="@+id/CardimageView4"
            android:layout_width="50dp"
            android:layout_height="50dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/backgrounddeepgreen"/>

    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/themeCard5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="100dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_centerInParent="true"
        android:layout_marginHorizontal="10dp">

        <ImageView
            android:id="@+id/CardimageView5"
            android:layout_width="50dp"
            android:layout_height="50dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/backgrounddeepgreen"/>

    </androidx.cardview.widget.CardView>

</LinearLayout>

<TextView
    android:id="@+id/textView28"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:text="Пароль"
    android:textAlignment="center"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingTop="4dp"
        app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText

```

```

        android:id="@+id/cur_pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Текущий пароль" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="4dp"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/new_pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Новый пароль" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="4dp"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/rep_pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Повторите пароль" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/change_pass"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Изменить"
    android:background="@drawable/button"/>

<TextView
    android:id="@+id/textView30"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:text="Имя пользователя"
    android:textAlignment="center"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="4dp">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/new_name"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Новое имя пользователя" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/change_name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Изменить"
    android:background="@drawable/button"/>

<TextView
    android:visibility="gone"
    android:id="@+id/textView31"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:text="Редактировать статус администратора
пользователя"
    android:textAlignment="center"

    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<com.google.android.material.textfield.TextInputLayout
    android:visibility="gone"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="4dp"
    android:id="@+id/adminNameBlock">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/adminName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Имя пользователя" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:visibility="gone"
    android:id="@+id/changeAdmin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="Изменить"
    android:background="@drawable/button"/>

</LinearLayout>

</ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>

```

Приложение 45 - bottom_nav_menu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:id="@+id/home"
    android:icon="@drawable/ic_round_home"
    android:title="Главная" />
  <item
    android:id="@+id/search"
    android:icon="@drawable/ic_round_search"
    android:title="Поиск" />
  <item
    android:id="@+id/profile"
    android:icon="@drawable/ic_round_person"
    android:title="Профиль" />
  <item
    android:id="@+id/info"
    android:icon="@drawable/ic_round_info"
    android:title="Инфо" />
</menu>

```

Приложение 46 -. colors.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
  <color name="ripple_effect">#AA5353</color>
  <color name="default_color">#6750A4</color>
  <color name="del_color">#D12121</color>
  <color name="pressed_color">#673AB7</color>
  <color name="itemcolor">#673AB7</color>

  <!-- ORANGE -->
  <color name="textOrange">#FF5722</color>

  <!-- GREEN -->
  <color name="green_primary">#4CAF50</color>
  <color name="green_primary_dark">#215A23</color>
  <color name="green_accent">#8BC34A</color>
  <color name="green_vivid">#A5FA42</color>
  <color name="green_text">#FFFFFF</color>

  <!-- DEEP PURPLE -->
  <color name="deep_purple_primary">#673AB7</color>
  <color name="deep_purple_primary_dark">#320B86</color>
  <color name="deep_purple_accent">#9575CD</color>
  <color name="deep_purple_vivid">#AA00FF</color>
  <color name="deep_purple_text">#FFFFFF</color>

  <!-- LIGHTBLUE -->
  <color name="lightblue_primary">#03A9F4</color>
  <color name="lightblue_primary_dark">#01579B</color>
  <color name="lightblue_accent">#40C4FF</color>
  <color name="lightblue_vivid">#80D8FF</color>
  <color name="lightblue_text">#FFFFFF</color>

  <!-- ORANGE -->
  <color name="orange_primary">#FF9800</color>
  <color name="orange_primary_dark">#E65100</color>

```

```

        <color name="orange_accent">#FFAB40</color>
        <color name="orange_vivid">#FFD180</color>
        <color name="orange_text">#FFFFFF</color>

</resources>

```

Приложение 47 - *dimens.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="text_margin">16dp</dimen>
</resources>

```

Приложение 48 - *orangetheme.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="OrangeTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Другие настройки темы -->
        <item name="android:textColor">@color/textOrange</item>
    </style>
</resources>

```

Приложение 49 - *strings.xml*

```

<resources>
    <string name="app_name">DiskWizard</string>
    <string name="heading">MVVM Architecture Pattern</string>
    <string name="email_hint">Enter your Email ID</string>
    <string name="password_hint">Enter your password</string>
    <string name="button_text">Login</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
</resources>

```

Приложение 50 - *themes.xml*

```

<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.DiskWizard"
parent="Theme.Material3.DayNight.NoActionBar">
        <item name="android:statusBarColor">@android:color/transparent</item>
        <item
name="android:navigationBarColor">@android:color/transparent</item>
        <item name="android:windowLightStatusBar">false</item>
        <item name="rippleColor">@color/default_color</item>
        <item name="android:fontFamily">@font/gothampro</item>
    </style>

    <style name="Base.Theme.DiskWizard.Green"
parent="Theme.Material3.DayNight.NoActionBar">
        <item name="android:statusBarColor">@android:color/transparent</item>
        <item
name="android:navigationBarColor">@android:color/transparent</item>
        <item name="android:windowLightStatusBar">false</item>

        <!-- Переопределение цветовых атрибутов -->
        <item name="colorPrimary">@color/green_accent</item> <!-- Рамка
активного поля, цвет кнопки -->
        <item name="rippleColor">@color/green_vivid</item>
    </style>

```

```

        <item name="colorOnPrimary">@color/green_text</item> <!-- Текст в
кнопке -->
        <item name="colorPrimaryContainer">@color/green_primary_dark</item>
<!-- FAB -->
        <item name="colorOnPrimaryContainer">@color/green_accent</item><!--
FAB text color-->

        <item name="colorOnSurface">@color/green_accent</item> <!-- Набираем
текст -->
        <item name="colorOnSurfaceVariant">@color/green_accent</item><!--
Поля-текст -->
        <item name="colorOutline">@color/green_primary</item><!-- Поля-
границы -->

        <item name="android:windowLightNavigationBar"
tools:targetApi="o_mr1">true</item>
        <item name="android:fontFamily">@font/gothampro</item>

</style>

<style name="Base.Theme.DiskWizard.DeepPurple"
parent="Theme.Material3.DayNight.NoActionBar">
    <item name="android:statusBarColor">@android:color/transparent</item>
    <item
name="android:navigationBarColor">@android:color/transparent</item>
    <item name="android:windowLightStatusBar">>false</item>

    <!-- Переопределение цветовых атрибутов-->
    <item name="colorPrimary">@color/deep_purple_primary_dark</item> <!--
Рамка активного поля, цвет кнопки -->
    <item name="rippleColor">@color/deep_purple_vivid</item>
    <item name="colorOnPrimary">@color/deep_purple_text</item> <!--
Текст в кнопке -->
    <item
name="colorPrimaryContainer">@color/deep_purple_primary_dark</item> <!-- FAB
-->
    <item
name="colorOnPrimaryContainer">@color/deep_purple_accent</item><!-- FAB text
color-->

    <item name="colorOnSurface">@color/deep_purple_accent</item> <!--
Набираем текст -->
    <item
name="colorOnSurfaceVariant">@color/deep_purple_accent</item><!-- Поля-текст
-->
    <item name="colorOutline">@color/deep_purple_primary</item><!-- Поля-
границы -->

    <item name="android:windowLightNavigationBar"
tools:targetApi="o_mr1">true</item>
    <item name="android:fontFamily">@font/gothampro</item>
</style>

<style name="Base.Theme.DiskWizard.LightBlue"
parent="Theme.Material3.DayNight.NoActionBar">
    <item name="android:statusBarColor">@android:color/transparent</item>
    <item
name="android:navigationBarColor">@android:color/transparent</item>
    <item name="android:windowLightStatusBar">>false</item>

```



```

        <!-- Переопределение цветовых атрибутов-->
        <item name="colorPrimary">@color/lightblue_accent</item> <!-- Рамка
активного поля, цвет кнопки -->
        <item name="rippleColor">@color/lightblue_vivid</item>
        <item name="colorOnPrimary">@color/lightblue_text</item> <!-- Текст
в кнопке -->
        <item
name="colorPrimaryContainer">@color/lightblue_primary_dark</item> <!-- FAB --
>
        <item
name="colorOnPrimaryContainer">@color/lightblue_accent</item><!-- FAB text
color-->

        <item name="colorOnSurface">@color/lightblue_accent</item> <!--
Набираем текст -->
        <item name="colorOnSurfaceVariant">@color/lightblue_accent</item><!--
Поля-текст -->
        <item name="colorOutline">@color/lightblue_primary</item><!-- Поля-
границы -->

        <item name="android:windowLightNavigationBar"
tools:targetApi="o_mr1">true</item>
        <item name="android:fontFamily">@font/gothampro</item>
    </style>

    <style name="Base.Theme.DiskWizard.Orange"
parent="Theme.Material3.DayNight.NoActionBar">
        <item name="android:statusBarColor">@android:color/transparent</item>
        <item
name="android:navigationBarColor">@android:color/transparent</item>
        <item name="android:windowLightStatusBar">false</item>

        <!-- Переопределение цветовых атрибутов-->
        <item name="colorPrimary">@color/orange_accent</item> <!-- Рамка
активного поля, цвет кнопки -->
        <item name="rippleColor">@color/orange_vivid</item>
        <item name="colorOnPrimary">@color/orange_text</item> <!-- Текст в
кнопке -->
        <item name="colorPrimaryContainer">@color/orange_primary_dark</item>
<!-- FAB -->
        <item name="colorOnPrimaryContainer">@color/orange_accent</item><!--
FAB text color-->

        <item name="colorOnSurface">@color/orange_accent</item> <!-- Набираем
текст -->
        <item name="colorOnSurfaceVariant">@color/orange_accent</item><!--
Поля-текст -->
        <item name="colorOutline">@color/orange_primary</item><!-- Поля-
границы -->

        <item name="android:windowLightNavigationBar"
tools:targetApi="o_mr1">true</item>
        <item name="android:fontFamily">@font/gothampro</item>
    </style>
</resources>

```