МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема Разработка приложения по обучению английскому языку для программистов

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студентка 3 курса,  группы ИСп 21-2К  Козлова Дарья Алексеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2024

Аннотация

"Основы рисования" - это инновационное мобильное приложение, разработанное для тех, кто хочет освоить искусство рисования с помощью своего смартфона или планшета. Созданное на платформе Java с применением Android Studio, это приложение предлагает широкий выбор обучающих материалов и практических упражнений, которые помогут как начинающим, так и опытным художникам развивать свои навыки.

Главной особенностью "Основ рисования" является его простой и интуитивно понятный интерфейс, а также гибкая система настройки обучающего процесса.

"Основы рисования" представляют собой не просто приложение, а полноценный инструмент для творчества, доступный в любое время и в любом месте.

Ссылка на мой GitHub: <https://github.com/dariaass>

Оглавление

[Введение 4](#_Toc165375160)

[1 Описание предметной области 4](#_Toc165375161)

[1.1 Аналоги разрабатываемого приложения 4](#_Toc165375162)

[1.2 Техническое задание 11](#_Toc165375163)

[1.3 Описание структуры приложения 13](#_Toc165375164)

[Описание разработки приложения 15](#_Toc165375165)

[2.1 Обоснование средств разработки 17](#_Toc165375166)

[2.2 Разработка интерфейса 18](#_Toc165375167)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта 18](#_Toc165375168)

[3.1 Интеграционное тестирование: 42](#_Toc165375169)

[3.2 Пользовательское тестирование: 42](#_Toc165375170)

[3.3 Установка приложения: 44](#_Toc165375171)

[Заключение 45](#_Toc165375172)

[Список использованной литературы: 46](#_Toc165375173)

[Приложения 47](#_Toc165375174)

# Введение

1. Описание предметной области:

Предметной областью для мобильного приложения "Основы Рисования" является изучение и практика рисования. Задача приложения состоит в обучении пользователей основам рисования, развитии их творческого потенциала и помощи им достичь определенного уровня мастерства в этом искусстве.

Предметная область мобильного приложения "Основы рисования" охватывает широкий спектр тем, связанных с изучением и практикой рисования. Приложение предназначено для людей всех возрастов и уровней навыков, начиная от начинающих художников до опытных профессионалов, которые желают улучшить свои навыки и расширить свой творческий потенциал.

Теперь давайте рассмотрим аналоги разрабатываемого мной приложения, чтобы выявить их сильные и слабые стороны:

* 1. Аналоги разрабатываемого приложения

Sketchbook **–** Это приложение от Autodesk, которое предлагает широкий набор кистей, инструментов и эффектов для рисования на мобильных устройствах. Оно также включает в себя обучающие материалы и ресурсы для художников



Рисунок 1 – логотип Sketchbook

Функционал приложения:

1. Общие впечатления: Программа предлагает набор кистей для рисования, которые имитируют реальные инструменты художника: карандаш, кисточка, маркер, аэрозоль, ручка, фломастер. Сохранять можно в несколько разных растровых форматов, в том числе, в PSD. Поддерживаются слои и рисование стилусом.

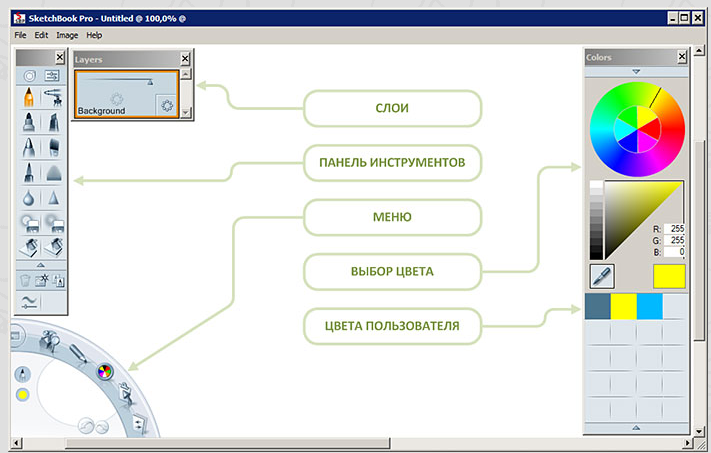


Рисунок 2 - Общие впечатления

1. Знакомство с особенностями интерфейса:
2. Контекстное меню. Организовано оно для многих инструментов таким образом: пользователь нажимает левой кнопкой мышы на активную кнопку и получает варианты подменю в виде кнопочек, выстроенных по кругу. Не отпуская левой кнопки мыши надо протянуть курсор к нужному пункту, чтобы активировать нужную функцию. Кстати, все эти меню можно настроить по своему вкусу, т.е. в меню выводить те кнопки тех инструментов, которые именно вам нужны. Это настраивается через панель меню "Edit > Preferences > Lagoon".
3. Move/Rotate/Scale. Эти три функции (передвижение/вращение/масштабирование), применяемые к слою, умещаются в крайне удобном виде на одной панельке. Это вообще очень выигрышная "фишка" программы. Вызывается эта панелька так, как указано на иллюстрации ниже. Панель Move/Rotate/Scale разделена на сектора, каждый из которых отвечает за определенный вид модификации слоя. Чтобы изменить слой надо кликнуть на нужном секторе и, держа левую кнопку мыши нажатой, потянуть мышь вправо/влево/вверх/вниз в зависимости от того, каких изменений вы хотите от слоя.
4. Быстрое изменение размера кисти. Полупрозрачная панелька для этого может все время находится на холсте. Ее можно передвигать, можно и закрыть. А вызывается она по кнопке вверху панели инструментов. Чтобы быстро изменить размер кисти, надо просто нажать на нее левой кнопкой мыши и потянуть мышь вправо/влево.
5. Вызов параметров инструмента. Отредактировать параметры любой кисти или другого инструмента можно после вызова окна настройки параметров – совершите двойной клик на кнопке инструмента. Можно также нажать на кнопку вверху панели инструментов.

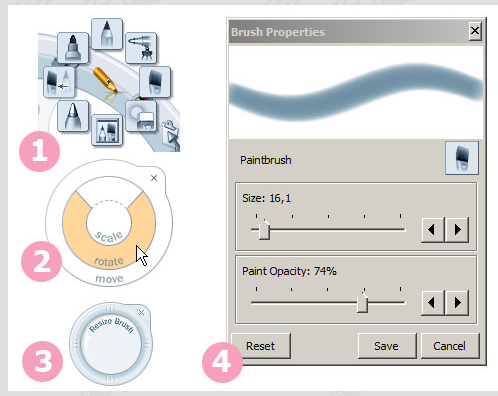


Рисунок 3 - Знакомство с особенностями интерфейса

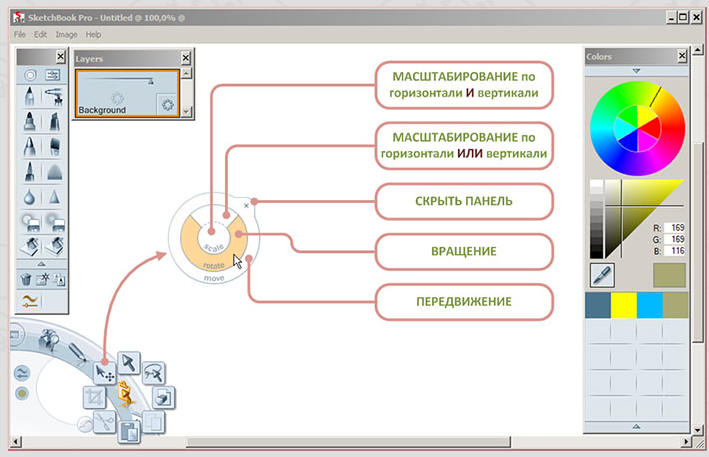


Рисунок 4 - Знакомство с особенностями интерфейса

1. Настройка кисти: в программе можно создать новую кисть – настроек будет в 5 раз больше. Чтобы создать новую кисть на панели инструментов надо кликнуть на кнопку. Выскочит окошко с выбором типа кисти. После выбора и нажатия кнопки [Create] кисть будет создана и ее иконка появится внизу панели инструментов. Чтобы отредактировать параметры новой кисти, надо совершить на иконке двойной клик или при выделенной иконке кликнуть по нужной кнопке. Чтобы удалить созданную вами кисть, надо ее выделить и нажать на кнопку.

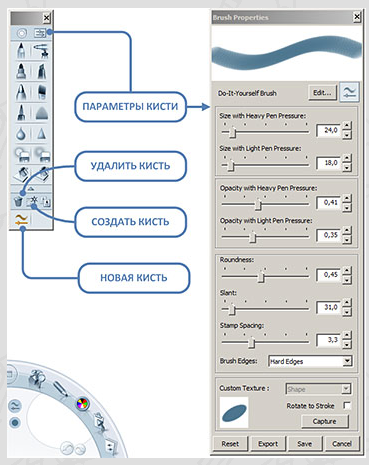


Рисунок 5 - Настройка кисти

1. Панель инструментов:

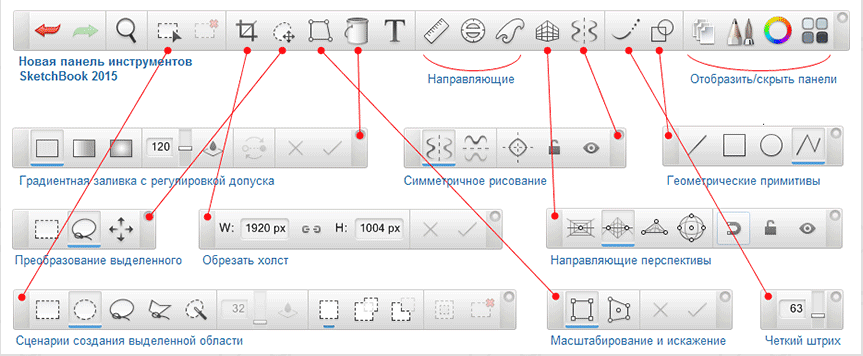


Рисунок 6 – Панель инструментов

Теперь предлагаю рассмотреть следующий аналог: ArtRage



Рисунок 7 – Логотип ArtRage

Как работает ArtRage:

ArtRage — это программа для рисования и живописи, которая имитирует рисование на обычной бумаге привычными для художника инструментами.

Основные принципы работы:

1. Программа очень точно и красиво воспроизводит масляные краски, которые даже некоторое время высыхают, и их можно всячески смешивать, размазывать.
2. В зависимости от выбранной структуры холста изменяются свойства краски, нанесённой на поверхность.
3. Многослойность холста позволяет подложить образец и рисовать по нему. Образцу задаётся масштаб, прозрачность.
4. Слои и операции над ними — всё что необходимо: смешивание при наложении, фильтры и кисти PhotoShop, а также PlugIn-ы.
5. Графический планшет позволяет не только рисовать, но и обрабатывать фотографии.
6. В данной программе имеются масса инструментов: масляная кисть, графитовый карандаш, пастель, валик и прочее.

Функционал приложения:

1. Интерфейс программы Art Rage:



Рисунок 8 - Интерфейс программы Art Rage

1. Инструменты приложения: в данном графическом редакторе 15 инструментов для рисования: кисть масляная, карандаш, мастихин, аэрограф, мелки, блѐстки (стружка), валик, тюбик, пипетка, фломастер, пастель, ластик, шаблоны.



Рисунок 9 – Инструменты приложения

1. Палитра цветов: Палитра цветов находится в нижней правой части экрана и представляет собой спектр цветов. Визуально выбираем нужный цвет, кликнув по палитре. Далее его можно выбрать по насыщенности оттенка.



Рисунок 10 – Палитра цветов

1. Техническое задание

Наше приложение представляет собой платформу для обучения, нужно сделать следующие активности

1. Авторизация: экран для входа в приложение или регистрации нового пользователя.
2. Регистрация: экран для регистрации нового пользователя.
3. MainActivity (Главная активность): Основной экран, где пользователь может просматривать видеоуроки, выполнять задания, упражнения и также рисовать.
4. Обучающие материалы: видеоуроки, с помощью котрых пользователь познает основы рисования.
5. Упражнения и задания: различные задания для пользователя.
6. Инструменты рисования: Экран, где пользователь может как просто рисовать, так и выполнять задания.

Функциональность

1. Регистрация и профили пользователей

* Пользователи могут создавать учетные записи с помощью придуманного ими логина и пароля.

1. Обучающие материалы и уроки

* Приложение должно предоставлять доступ к разнообразным обучающим материалам, включая видеоуроки и инструкции.
* Материалы должны охватывать различные аспекты рисования, включая базовые техники, перспективу, композицию, использование различных стилей.

1. Интерактивные упражнения и задания

* Пользователям предоставляются задания и упражнения для практики, включая рисование по образцу и выполнение творческих задач.

1. Инструменты рисования

* Приложение должно предоставлять набор инструментов для цифрового рисования, включая кисти, маркеры и карандаши
* Инструменты должны быть настраиваемыми, чтобы пользователи могли адаптировать их под свои потребности.

Версионный контроль:

* Проект размещен в системе контроля версий Git.
* Разработчик работает с актуальной версией проекта из репозитория Git.

1. Макет приложения:

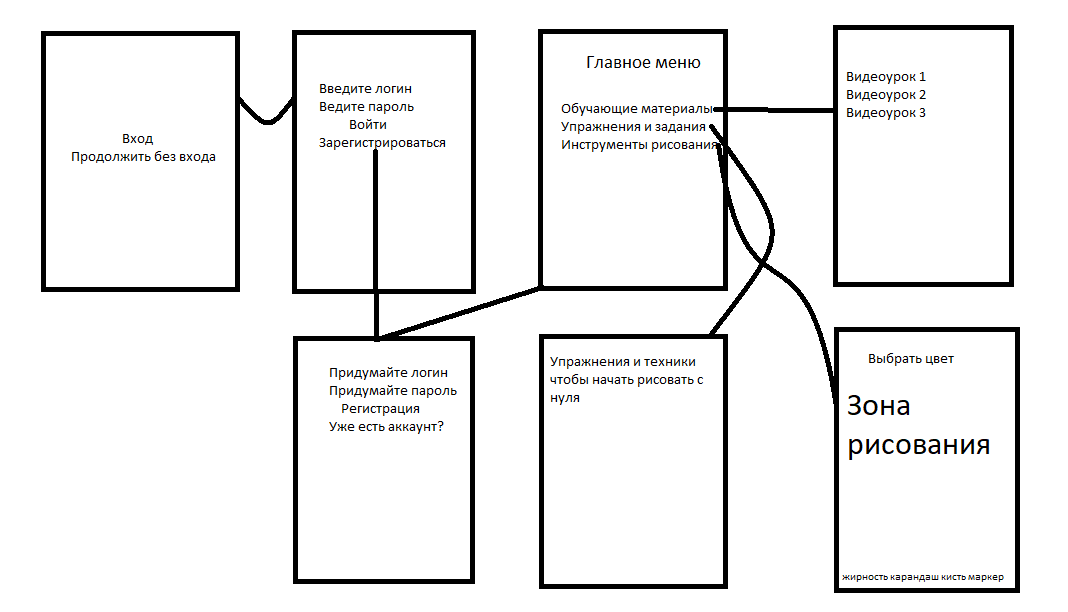


Рисунок 11 – Макет приложения

* 1. Описание структуры приложения

Мобильное приложение "Основы рисования" разрабатывается с целью предоставить пользователям доступное и удобное средство для изучения и практики рисования на мобильных устройствах. Приложение должно предоставлять обучающие материалы, интерактивные уроки, инструменты для практики и социальные функции для обмена опытом и вдохновения.

1. Начальный экран

* Возможность продолжить без входа: Пользователи также могут просматривать обучающие материалы и уроки без необходимости входа.

2. Главное меню

* Обучающие материалы: Ссылки на различные категории обучающих материалов, таких как базовые техники, перспектива, композиция, стили и материалы.
* Упражнения и задания: Раздел с интерактивными упражнениями и заданиями для практики, где пользователи могут выбирать и выполнять различные задания.
* Инструменты рисования: Доступ к инструментам для цифрового рисования, включая кисти, маркеры, карандаши и режимы рисования.

3. Профиль пользователя

* Галерея работ: Просмотр и управление загруженными работами пользователя.
* Настройки: Персонализация настроек приложения, таких как язык и уведомления

4. Экраны изучения и практики

* Уроки и материалы: Подробные материалы и уроки с инструкциями, видеороликами, демонстрациями и примерами.
* Упражнения и задания: Интерактивные упражнения и задания для практики, с возможностью загрузки своих работ

5. Дополнительные разделы

* Новости и обновления: Информация о новых уроках, материалах и функциях приложения.
* Поддержка и обратная связь: Раздел для связи с командой разработчиков, где пользователи могут задавать вопросы и отправлять отзывы.

Описание разработки приложения

1. Обоснование средств разработки
   1. Java

Плюсы:

* Официальный язык программирования для Android: Java является языком, на котором изначально написана платформа Android, что обеспечивает его бесшовную интеграцию и оптимизацию для разработки приложений.
* Большое сообщество: Благодаря своему широкому распространению Java обладает обширным и активным сообществом разработчиков, что означает наличие множества доступных ресурсов, библиотек и примеров кода, которые могут помочь вам в процессе разработки.
* Объектно-ориентированный язык: Парадигма объектно-ориентированного программирования (ООП) в Java хорошо подходит для создания модульных и масштабируемых приложений, что особенно важно для сложных проектов.
* Богатая стандартная библиотека: Java поставляется с обширной стандартной библиотекой, которая включает в себя множество функций, необходимых для разработки мобильных приложений, таких как сетевое взаимодействие, сенсорное управление, графика и работа с файлами.

Минусы:

* Производительность: Java может быть менее производительной, чем некоторые другие языки программирования, такие как C++, особенно при работе с ресурсоемкими задачами.
* Многословность: Код Java может быть более многословным по сравнению с некоторыми другими языками, что может привести к увеличению объема кода и снижению читаемости.
* Время выполнения: Компиляция Java-кода в машинный код происходит во время выполнения, что может замедлить запуск приложения по сравнению с языками, которые компилируются заранее.
  1. Android Studio

Плюсы:

* Официальная IDE для Android: Android Studio является официальной интегрированной средой разработки (IDE) от Google, созданной специально для разработки Android-приложений.
* Широкий набор функций: Android Studio предлагает множество функций, которые упрощают и оптимизируют процесс разработки, включая редактор кода с подсветкой синтаксиса, визуальный редактор макетов, встроенный эмулятор устройств, инструменты отладки и профилирования.
* Интеграция с инструментами Google: Android Studio тесно интегрируется с другими инструментами и сервисами Google, такими как Firebase, Google Cloud Platform и Google Play Console, что упрощает разработку и публикацию приложений.
* Активное сообщество: Android Studio обладает большим и активным сообществом пользователей, что означает доступ к множеству обучающих материалов, форумов и плагинов.

Минусы:

* Требует больше ресурсов: Android Studio может быть более требовательным к системным ресурсам, чем некоторые другие IDE, что может привести к замедлению работы на менее мощных компьютерах.
* Сложность освоения: Для начинающих разработчиков Android Studio может показаться сложной из-за обилия функций и инструментов.
* Зависимость от Google: Android Studio тесно связана с экосистемой Google, что может ограничить возможности использования сторонних инструментов и библиотек.
  1. IntelliJ IDEA

Плюсы:

* Мощная и гибкая среда разработки, предоставляющая множество инструментов для создания качественного программного обеспечения.
* Поддержка не только Java, но и других языков программирования, таких как Kotlin, JavaScript, Python и многих других.
* Широкий выбор плагинов и расширений, что позволяет настроить IDE под свои потребности.
* Быстрая и стабильная работа, даже с большими проектами.

Минусы:

* Не столь направленная на мобильную разработку, как Android Studio, хотя многие разработчики успешно используют её и для этой цели.

После анализа особенностей каждой из вышеперечисленных IDE и учитывая рекомендации со стороны сообщества разработчиков Android, было принято решение использовать Android Studio как основную среду разработки для данного проекта

1. Разработка интерфейса

Интерфейс разработан с учетом пользовательских потребностей, обеспечивая удобство использования и интуитивно понятное взаимодействие с приложением. Рассмотрим основные аспекты интерфейса по предоставленному коду:

1. Экран входа и возможность продолжить без входа:
   * Кнопля для входа в приложение.
   * Кнопка продолжения без входа.
2. Экран входа:
   * Поле для ввода логина.
   * Поле для ввода пороля.
   * Кнопка входа.
   * Кнопка регистрации.
3. Экран регистрации:
   * Поле для придуманного пользователем логина.
   * Поле для придуманного пользователем пароля.
   * Кнопка регистрация
   * Кнопка с вопросом «у вас уже есть аккаунт?».
4. Экран главное меню:
   * Кнопка «Обучающие материалы».
   * Кнопка «Упражнения и задания».
   * Кнопка «Инструменты рисования».
5. Экран «Обучающие материалы»:
   * Кнопка-ссылка с переходом на первый урок.
   * Кнопка-ссылка с переходом на второй урок.
   * Кнопка-ссылка с переходом на третий урок.
   * Кнопка-ссылка с переходом на четвертый урок.
6. Экран «Упражнения и задания»:

* Кнопка-ссылка на упражнения и техники для начала рисования.

1. Экран «Инструменты рисования»:

* Кнопка выбора цвета.
* Ползунок для изменения толщины.
* Кнопка, которая позволяет пользователю выбрать карандаш
* Кнопка «кисть»
* Кнопка «маркер».

1. Разработка логики работы приложения. Схема взаимодействия компонентов проекта

1. Авторизация:

* Пользователь входит в приложение или регистрируется
* При успешной авторизации пользователь переходит на главный экран (MainActivity).

2. Главный экран:

* Позволяет зайти пользователю в «Обучающие материалы»
* Позволяет зайти пользователю в «Упражнения и задания»
* Позволяет зайти пользователю в «Инструменты рисования»

3. Экран Обучающие материалы:

* Показывает список всех видеоуроков
* При нажатии на какой-либо видеоурок перекидывает пользователя по ссылке на него.

4. Экран Упражнения и задания:

* При нажатии показывает кнопку-ссылку всех упражнений и техник для рисования с нуля.
* Перекидывает пользователя с помощью ссылки в интернет, где находятся все задания и упражнения.

5. Экран Основы и рисования

* Предоставляет инструменты для рисования (кисть, палитру цветов, карандаш, маркер).
* Позволяет пользователю рисовать на холсте.

Это основной процесс работы приложения, который обеспечивает пользователю удобный доступ к образовательному контенту

1. Описание переменных, компонентов, классов и подпрограмм

AndroidManifest

1. MainActivity:

* Тип: Activity
* Описание: Главная активность приложения, отображает основной интерфейс для рисования.

1. LoginActivity:

* Тип: Activity
* Описание: Активность для входа в приложение.

1. MainMenuActivity:

* Тип: Activity
* Описание: Активность, отображающая главное меню приложения.

1. Activity\_Reg:

* Тип: Activity
* Описание: Активность для регистрации пользователей.

1. DrawingActivity:

* Тип: Activity
* Описание: Активность, позволяющая пользователю рисовать.

1. VideoLinksActivity:

* Тип: Activity
* Описание: Активность, содержащая ссылки на обучающие видеоматериалы.

1. WorkActivity:

* Тип: Activity
* Описание: Активность для выполнения упражнений и заданий.

Методы и подпрограммы:

1. onCreate() в MainActivity:
   * Описание: Этот метод вызывается при создании активности MainActivity. Здесь инициализируется интерфейс и устанавливаются обработчики событий.
2. onCreate() в LoginActivity:
   * Описание: Метод вызывается при создании активности LoginActivity. Он отвечает за инициализацию интерфейса и установку обработчиков событий.
3. onCreate() в MainMenuActivity:
   * Описание: Этот метод вызывается при создании активности MainMenuActivity. Он инициализирует интерфейс и устанавливает обработчики событий.
4. onCreate() в Activity\_Reg:
   * Описание: Метод вызывается при создании активности Activity\_Reg. Он отвечает за инициализацию интерфейса и установку обработчиков событий.
5. onCreate() в DrawingActivity:
   * Описание: Этот метод вызывается при создании активности DrawingActivity. Он инициализирует интерфейс для рисования и устанавливает обработчики событий.
6. onCreate() в VideoLinksActivity:
   * Описание: Метод вызывается при создании активности VideoLinksActivity. Он отвечает за инициализацию интерфейса и установку обработчиков событий.
7. onCreate() в WorkActivity:
   * Описание: Этот метод вызывается при создании активности WorkActivity. Он инициализирует интерфейс для выполнения упражнений и заданий и устанавливает обработчики событий.

Activity\_Reg

Класс: Activity\_Reg

* Тип: AppCompatActivity
* Описание: Этот класс представляет активность для регистрации пользователей в приложении.

Переменные:

1. etUsername
   * Тип: EditText
   * Описание: Поле ввода для имени пользователя.
2. etPassword
   * Тип: EditText
   * Описание: Поле ввода для пароля пользователя.
3. btnRegister
   * Тип: Button
   * Описание: Кнопка для запуска процесса регистрации.
4. userDatabase
   * Тип: UserDatabase
   * Описание: Объект базы данных пользователей.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод жизненного цикла активности, вызывается при ее создании. Инициализирует интерфейс активности и устанавливает обработчик нажатия на кнопку регистрации.
   * Параметры:
     + savedInstanceState: Объект типа Bundle, содержащий данные о предыдущем состоянии активности.
   * Возвращаемое значение: void
2. onClick(View v)
   * Описание: Метод обработчика нажатия на кнопку регистрации. Проверяет введенные пользователем данные, добавляет нового пользователя в базу данных и перенаправляет пользователя на экран входа.
   * Параметры:
     + v: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void

Класс: UserDatabase

* Тип: Не представлен в предоставленном коде, предположительно это пользовательский класс.
* Описание: Этот класс, вероятно, представляет базу данных пользователей и содержит методы для добавления новых пользователей, проверки существующих пользователей и другие операции.

DrawingActivity

Класс: DrawingActivity

* Тип: AppCompatActivity
* Описание: Этот класс представляет активность для рисования, где пользователь может выбирать цвет, толщину кисти и тип кисти для рисования.

Переменные:

1. drawingView
   * Тип: DrawingView
   * Описание: Объект класса DrawingView, представляющий виджет для рисования на экране.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод жизненного цикла активности, вызывается при ее создании. Инициализирует интерфейс активности и устанавливает обработчики событий для кнопок выбора цвета, толщины кисти и типа кисти.
   * Параметры:
     + savedInstanceState: Объект типа Bundle, содержащий данные о предыдущем состоянии активности.
   * Возвращаемое значение: void
2. showColorPicker()
   * Описание: Метод для отображения диалога выбора цвета. Создает диалог с списком цветов, когда пользователь выбирает цвет, устанавливает выбранный цвет для кисти в виджете для рисования.
   * Параметры: Нет
   * Возвращаемое значение: void

DrawingView

Класс: DrawingView

* Тип: View
* Описание: Этот класс представляет пользовательский виджет для рисования на экране.

Переменные:

1. currentPaint
   * Тип: Paint
   * Описание: Объект класса Paint, используется для текущих настроек кисти.
2. paths
   * Тип: List<Path>
   * Описание: Список всех путей, которые были нарисованы на экране.
3. paints
   * Тип: List<Paint>
   * Описание: Список всех кистей, которые использовались для рисования соответствующих путей.

Методы:

1. DrawingView(Context context, AttributeSet attrs)
   * Описание: Конструктор класса. Вызывает метод init() для инициализации переменных.
   * Параметры:
     + context: Контекст приложения.
     + attrs: Набор атрибутов из XML.
   * Возвращаемое значение: Нет
2. init()
   * Описание: Метод инициализации переменных. Создает новый объект Paint для текущей кисти, инициализирует списки paths и paints.
   * Параметры: Нет
   * Возвращаемое значение: void
3. setBrushType(String type)
   * Описание: Метод для установки типа кисти (карандаш, кисть, маркер).
   * Параметры:
     + type: Строка, представляющая тип кисти.
   * Возвращаемое значение: void
4. setBrushColor(int color)
   * Описание: Метод для установки цвета кисти.
   * Параметры:
     + color: Цвет в формате int.
   * Возвращаемое значение: void
5. setBrushThickness(float thickness)
   * Описание: Метод для установки толщины кисти.
   * Параметры:
     + thickness: Толщина кисти в пикселях.
   * Возвращаемое значение: void
6. onDraw(Canvas canvas)
   * Описание: Метод, вызываемый для отрисовки виджета. Отрисовывает все пути на канве с соответствующими кистями.
   * Параметры:
     + canvas: Объект Canvas, на котором происходит отрисовка.
   * Возвращаемое значение: void
7. onTouchEvent(MotionEvent event)
   * Описание: Метод для обработки событий прикосновений к экрану. Создает новый путь при касании экрана, добавляет линии к пути при перемещении пальца и завершает путь при отпускании пальца.
   * Параметры:
     + event: Объект MotionEvent, представляющий событие касания экрана.
   * Возвращаемое значение: boolean

LoginActivity

Класс: LoginActivity

* Тип: AppCompatActivity
* Описание: Этот класс представляет активность входа в приложение.

Переменные:

1. edtLogin
   * Тип: EditText
   * Описание: Поле ввода для логина пользователя.
2. edtPassword
   * Тип: EditText
   * Описание: Поле ввода для пароля пользователя.
3. btnLogin
   * Тип: Button
   * Описание: Кнопка для входа в систему.
4. btnRegister
   * Тип: Button
   * Описание: Кнопка для перехода к активности регистрации.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод жизненного цикла активности, вызывается при ее создании. Инициализирует интерфейс активности и устанавливает обработчики нажатия на кнопки входа и регистрации.
   * Параметры:
     + savedInstanceState: Объект типа Bundle, содержащий данные о предыдущем состоянии активности.
   * Возвращаемое значение: void
2. onClick(View v) для btnLogin
   * Описание: Метод обработчика нажатия на кнопку входа. Проверяет введенные пользователем логин и пароль. Если они верны, перенаправляет пользователя на главный экран. В противном случае выводит сообщение об ошибке.
   * Параметры:
     + v: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void
3. onClick(View v) для btnRegister
   * Описание: Метод обработчика нажатия на кнопку регистрации. Перенаправляет пользователя на активность регистрации.
   * Параметры:
     + v: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void

MainActivity

Класс: MainActivity

* Тип: AppCompatActivity
* Описание: Этот класс представляет главную активность приложения.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод жизненного цикла активности, вызывается при ее создании. Устанавливает макет (layout) для активности, находит кнопки на этом макете и устанавливает для них обработчики нажатия.
   * Параметры:
     + savedInstanceState: Объект типа Bundle, содержащий данные о предыдущем состоянии активности.
   * Возвращаемое значение: void
2. onClick(View v) для btnLogin
   * Описание: Метод обработчика нажатия на кнопку "Войти". Создает интент для перехода на активность входа (LoginActivity) и запускает этот интент.
   * Параметры:
     + v: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void
3. onClick(View v) для btnContinueWithoutLogin
   * Описание: Метод обработчика нажатия на кнопку "Продолжить без входа". Создает интент для перехода на главное меню (MainMenuActivity) и запускает этот интент. Если кнопка "Продолжить без входа" не существует, данный метод не выполняется.
   * Параметры:
     + v: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void

MainMenuActivity

Класс: MainMenuActivity

* Тип: AppCompatActivity
* Описание: Этот класс представляет главное меню приложения.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод жизненного цикла активности, вызывается при ее создании. Устанавливает макет (layout) для активности.
   * Параметры:
     + savedInstanceState: Объект типа Bundle, содержащий данные о предыдущем состоянии активности.
   * Возвращаемое значение: void
2. onLearningMaterialsClick(View view)
   * Описание: Метод обработчика нажатия на кнопку "Обучающие материалы". Создает интент для перехода на активность с видеоуроками (VideoLinksActivity) и запускает этот интент.
   * Параметры:
     + view: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void
3. onExercisesClick(View view)
   * Описание: Метод обработчика нажатия на кнопку "Упражнения и задания". Создает интент для перехода на активность с упражнениями (WorkActivity) и запускает этот интент.
   * Параметры:
     + view: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void
4. onDrawingToolsClick(View view)
   * Описание: Метод обработчика нажатия на кнопку "Инструменты для рисования". Создает интент для перехода на активность с инструментами для рисования (DrawingActivity) и запускает этот интент.
   * Параметры:
     + view: Объект типа View, представляющий элемент интерфейса, на который было совершено нажатие.
   * Возвращаемое значение: void

UserEntry.java

Класс: User

* Описание: Представляет запись в базе данных, содержащую информацию о пользователе (логин и пароль).

Переменные:

1. username
   * Тип: String
   * Описание: Логин пользователя.
2. password
   * Тип: String
   * Описание: Пароль пользователя.

Методы:

1. User(String username, String password)
   * Описание: Конструктор класса User. Создает новый объект пользователя с заданным логином и паролем.
   * Параметры:
     + username: Логин пользователя.
     + password: Пароль пользователя.
2. getUsername()
   * Описание: Возвращает логин пользователя.
   * Возвращаемое значение: String - логин пользователя.
3. getPassword()
   * Описание: Возвращает пароль пользователя.
   * Возвращаемое значение: String - пароль пользователя.

Класс: UserDatabase

* Описание: Класс базы данных для хранения логинов и паролей пользователей.

Переменные:

1. users
   * Тип: List<User>
   * Описание: Список пользователей, хранящихся в базе данных.

Методы:

1. UserDatabase()
   * Описание: Конструктор класса UserDatabase. Инициализирует список пользователей.
2. addUser(String username, String password)
   * Описание: Добавляет нового пользователя в базу данных, если пользователь с таким логином еще не существует.
   * Параметры:
     + username: Логин нового пользователя.
     + password: Пароль нового пользователя.
3. isUserExist(String username)
   * Описание: Проверяет, существует ли пользователь с заданным логином в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
   * Возвращаемое значение: boolean - true, если пользователь существует, иначе false.
4. validateUser(String username, String password)
   * Описание: Проверяет, совпадает ли логин и пароль пользователя с данными в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
     + password: Пароль пользователя для проверки.
   * Возвращаемое значение: boolean - true, если логин и пароль совпадают, иначе false.
5. getAllUsers()
   * Описание: Получает список всех пользователей в базе данных (используется для тестирования).
   * Возвращаемое значение: List<User> - список всех пользователей.

User

Класс: User

* Описание: Представляет запись в базе данных, хранящую информацию о пользователе (логин и пароль).

Переменные:

1. username
   * Тип: String
   * Описание: Логин пользователя.
2. password
   * Тип: String
   * Описание: Пароль пользователя.

Методы:

1. User(String username, String password)
   * Описание: Конструктор класса User. Создает новый объект пользователя с указанным логином и паролем.
   * Параметры:
     + username: Логин пользователя.
     + password: Пароль пользователя.
2. getUsername()
   * Описание: Возвращает логин пользователя.
   * Возвращаемое значение: String - логин пользователя.
3. getPassword()
   * Описание: Возвращает пароль пользователя.
   * Возвращаемое значение: String - пароль пользователя.

Класс: UserDatabase

* Описание: Класс базы данных для хранения логинов и паролей пользователей.

Переменные:

1. users
   * Тип: List<User>
   * Описание: Список объектов User, представляющих пользователей в базе данных.

Методы:

1. UserDatabase()
   * Описание: Конструктор класса UserDatabase. Инициализирует список пользователей.
2. addUser(String username, String password)
   * Описание: Добавляет нового пользователя в базу данных, если пользователь с указанным логином еще не существует.
   * Параметры:
     + username: Логин нового пользователя.
     + password: Пароль нового пользователя.
3. isUserExist(String username)
   * Описание: Проверяет, существует ли пользователь с указанным логином в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
   * Возвращаемое значение: boolean - true, если пользователь существует, иначе false.
4. validateUser(String username, String password)
   * Описание: Проверяет, совпадает ли логин и пароль пользователя с данными в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
     + password: Пароль пользователя для проверки.
   * Возвращаемое значение: boolean - true, если логин и пароль совпадают, иначе false.
5. getAllUsers()
   * Описание: Получает список всех пользователей в базе данных (используется для тестирования).
   * Возвращаемое значение: List<User> - список всех пользователей.

UserEntry

Класс: UserDatabase

* Описание: Класс базы данных для хранения логинов и паролей пользователей.

Переменные:

1. users
   * Тип: List<User>
   * Описание: Список объектов User, представляющих пользователей в базе данных.

Методы:

1. UserDatabase()
   * Описание: Конструктор класса UserDatabase. Инициализирует список пользователей.
2. addUser(String username, String password)
   * Описание: Добавляет нового пользователя в базу данных, если пользователь с указанным логином еще не существует.
   * Параметры:
     + username: Логин нового пользователя.
     + password: Пароль нового пользователя.
3. isUserExist(String username)
   * Описание: Проверяет, существует ли пользователь с указанным логином в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
   * Возвращаемое значение: boolean - true, если пользователь существует, иначе false.
4. validateUser(String username, String password)
   * Описание: Проверяет, совпадает ли логин и пароль пользователя с данными в базе данных.
   * Параметры:
     + username: Логин пользователя для проверки.
     + password: Пароль пользователя для проверки.
   * Возвращаемое значение: boolean - true, если логин и пароль совпадают, иначе false.
5. getAllUsers()
   * Описание: Получает список всех пользователей в базе данных (используется для тестирования).
   * Возвращаемое значение: List<User> - список всех пользователей.

VideoLinksActivity

Класс: VideoLinksActivity

* Описание: Активность, отображающая список видеоуроков и открывающая выбранный видеоурок в браузере при клике на него.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод, вызываемый при создании активности. Инициализирует интерфейс и устанавливает обработчик нажатий на элементы списка.
   * Параметры:
     + savedInstanceState: Объект Bundle, содержащий состояние активности.
2. (ListView.OnItemClickListener)
   * Описание: Обработчик нажатий на элементы списка видеоуроков. При клике на элемент списка открывает соответствующий видеоурок в браузере.
   * Параметры:
     + parent: Родительский объект AdapterView.
     + view: Нажатый элемент списка.
     + position: Позиция нажатого элемента в списке.
     + id: Идентификатор нажатого элемента в списке.

Переменные:

1. videoTitles
   * Тип: String[]
   * Описание: Массив с названиями видеоуроков.
2. videoLinks
   * Тип: String[]
   * Описание: Массив с ссылками на видеоуроки.
3. listView
   * Тип: ListView
   * Описание: Объект ListView для отображения списка видеоуроков.
4. adapter
   * Тип: ArrayAdapter<String>
   * Описание: Адаптер для отображения списка видеоуроков в ListView.

Действия:

1. Создается адаптер (ArrayAdapter), использующий макет android.R.layout.simple\_list\_item\_1 для отображения списка видеоуроков в ListView.
2. Устанавливается адаптер для ListView.
3. Устанавливается обработчик нажатий на элементы списка. При выборе элемента списка формируется интент для открытия ссылки на соответствующий видеоурок в браузере.

WorkActivity

Класс: WorkActivity

* Описание: Активность для отображения списка упражнений и заданий для рисования.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Метод, вызываемый при создании активности. Инициализирует интерфейс и устанавливает адаптер для списка упражнений и заданий.
   * Параметры:
     + savedInstanceState: Объект Bundle, содержащий состояние активности.

Переменные:

1. videoTitles
   * Тип: String[]
   * Описание: Массив с названиями упражнений и заданий.
2. videoLinks
   * Тип: String[]
   * Описание: Массив с ссылками на упражнения и задания.
3. listView
   * Тип: ListView
   * Описание: Элемент пользовательского интерфейса, используемый для отображения списка упражнений и заданий.
4. adapter
   * Тип: ArrayAdapter<String>
   * Описание: Адаптер для связывания данных (названий упражнений и заданий) с элементами списка упражнений и заданий.

Действия:

1. Создается адаптер (ArrayAdapter), который использует макет android.R.layout.simple\_list\_item\_1 для отображения списка упражнений и заданий.
2. Адаптер устанавливается для ListView, чтобы отобразить список упражнений и заданий.
3. Устанавливается обработчик нажатий на элементы списка. При выборе элемента списка формируется интент для открытия ссылки на соответствующее упражнение или задание во внешнем браузере.

Тестирование и установка приложения

## 3.1 Интеграционное тестирование:

1. Вход в приложение:
   * Откройте приложение и убедитесь, что экран входа отображается.
   * Введите неверный адрес электронной почты или пароль и нажмите кнопку входа. Убедитесь, что приложение отображает сообщение об ошибке.
   * Введите верный адрес электронной почты и пароль и нажмите кнопку входа. Убедитесь, что пользователь успешно входит в приложение и перенаправляется на главный экран (MainActivity).
2. Главное меню:
   * Перейдите на главный экран (MainActivity) и убедитесь, что все кнопки-ссылки работают
3. Обучающие материалы:
   * Перейдите на экран материалов. Проверьте, что все видеоуроки работают
   * Нажмите на одну из лекций и убедитесь, что открывается соответствующий экран.
4. Упражнения и задания:
   * Перейдите на экран заданий и убедитесь, что задания отображаются корректно.
5. Инструменты рисования:
   * Перейдите на экран и убедитесь, что открывается холст для рисования
   * Убедитесь, что все инструменты рисования работают

## 3.2 Пользовательское тестирование:

1. Оценка интерфейса:
   * Проверьте, насколько интуитивно понятен интерфейс. Пользователь должен легко находить основные функции, такие как выбор кисти, изменение цвета и толщины линии.
   * Проверьте, насколько хорошо работают элементы управления, такие как кнопки, панели инструментов и ползунки.
2. Тестирование функциональности:
   * Убедитесь, что все основные функции работают корректно: рисование, выбор инструментов и т. д.
   * Проверьте работу всех инструментов: кисти, маркера и карандаша.
3. Тестирование совместимости:
   * Протестируйте приложение на разных устройствах (с разными размерами экрана и версиями операционных систем).
   * Убедитесь, что приложение работает корректно на Android.
4. Тестирование производительности:
   * Оцените скорость работы приложения на разных устройствах.
   * Проверьте, не возникают ли задержки при использовании различных инструментов или при работе с большими изображениями.
5. Обратная связь пользователя:
   * Проведите опрос пользователей после использования приложения. Узнайте, что им нравится, а что нет, и получите предложения по улучшению.
   * Организуйте фокус-группы, чтобы получить более подробную обратную связь от целевой аудитории.
6. Тестирование на устойчивость:
   * Проверьте, как приложение работает в условиях ограниченного доступа к интернету или при сбоях соединения.
   * Проверьте, как приложение обрабатывает непредвиденные ситуации, такие как отсутствие памяти на устройстве или неожиданное завершение работы.
7. Тестирование безопасности:
   * Убедитесь, что приложение не передает или не сохраняет конфиденциальную информацию пользователя без его разрешения.
   * Проверьте, что приложение не подвержено известным уязвимостям безопасности.

## 3.3 Установка приложения:

1. Создание APK:
   * В Android Studio выберите "Build" -> "Build Bundle(s) / APK(s)" -> "Build APK(s)".
   * Это создаст APK-файл(-ы) вашего приложения, который(-ые) вы можете использовать для установки.
2. Перенос APK на устройство:
   * Перенесите APK-файл(-ы) на ваше устройство через USB, облачное хранилище или любой другой метод передачи файлов.
3. Установка приложения:
   * На вашем устройстве перейдите в настройки безопасности и разрешите установку приложений из неизвестных источников, если это ещё не сделано.
   * Используйте файловый менеджер на устройстве, чтобы найти и открыть APK-файл(-ы) приложения.
   * Следуйте инструкциям по установке, чтобы завершить установку приложения.

Заключение

В данной работе было успешно разработано мобильное приложение "Основы рисования", предназначенное для людей, желающих освоить основы искусства рисования непосредственно на своих мобильных устройствах. Основываясь на современных технологиях Android Studio, приложение обеспечивает высокую функциональность и доступность на различных платформах, что делает его удобным инструментом как для начинающих, так и для опытных художников.

Основная цель приложения — предоставить пользователям интуитивно понятный и эффективный инструмент для обучения рисованию.

Кроме того, приложение предлагает инновационные решения, такие как интеграция с облачными сервисами для сохранения данных и механизмы безопасности, обеспечивающие защиту персональной информации пользователей.

# Список использованной литературы:

* 1. Макдональд М. "Android. Разработка приложений для смартфонов и планшетов". Перевод с английского. Издательство "ДМК Пресс", 2019.
  2. Филлипс Б., Харди К., Козгроув Дж. "Программирование под Android. Руководство Big Nerd Ranch". 4-е издание. Издательство Big Nerd Ranch, 2019.
  3. Моррис С., Шоу Л. "Программирование под Android: Руководство по языку Java для нового поколения мобильных устройств". Издательство O'Reilly Media, 2015.
  4. Энтони Лоуренс. "Java для начинающих. Изучайте язык программирования за 24 часа". Издательство "ДМК Пресс", 2020.
  5. Кэтти Сьерра, Берт Бейтс. "Изучаем Java". Издательство "Питер", 2019.
  6. Владимир Иванов, Юлия Норицына. "Разработка Android-приложений для чайников". Издательство "Питер", 2019.

7. Евгений Масленников. "Разработка приложений для Android на Java". Издательство "Питер", 2018.

Приложения

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.example.d\_osnovi"> <!-- Правильное имя пакета -->  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.D\_osnovi"> <!-- Элемент application -->  
  
 <!-- Главная активность -->  
 <activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:label="Основы рисования"  
 android:theme="@style/Theme.D\_osnovi"> <!-- Элемент activity -->  
 <intent-filter> <!-- Intent-filter должен быть внутри activity -->  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity> <!-- Закрываем activity -->  
  
 <!-- Активность входа -->  
 <activity  
 android:name=".LoginActivity"  
 android:exported="true"  
 android:label="Основы рисования"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
  
 <!-- Главное меню -->  
 <activity  
 android:name=".MainMenuActivity"  
 android:exported="true"  
 android:label="Основы рисования"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
  
 <!-- Активность регистрации -->  
 <activity  
 android:name=".Activity\_Reg"  
 android:exported="true"  
 android:label="Основы рисования"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
  
 <!-- Активность рисования -->  
 <activity  
 android:name=".DrawingActivity"  
 android:exported="true"  
 android:label="Основы рисования"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:name=".VideoLinksActivity"  
 android:exported="true"  
 android:label="Обучающие материалы"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:name=".WorkActivity"  
 android:exported="true"  
 android:label="Упражнения и задания"  
 android:theme="@style/Theme.D\_osnovi">  
 <intent-filter>  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

package com.example.d\_osnovi;  
  
import android.content.Intent; // Для перехода к другим активити  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class Activity\_Reg extends AppCompatActivity {  
 private EditText etUsername;  
 private EditText etPassword;  
 private Button btnRegister;  
 private UserDatabase userDatabase; // Объект базы данных  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_reg*);  
  
 // Инициализация элементов интерфейса  
 EditText etUsername = findViewById(R.id.*etNewLogin*);  
 EditText etPassword = findViewById(R.id.*etNewPassword*);  
 Button btnRegister = findViewById(R.id.*btnRegister*);  
  
 userDatabase = new UserDatabase(); // Инициализация базы данных  
  
 btnRegister.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String username = etUsername.getText().toString().trim();  
 String password = etPassword.getText().toString().trim();  
  
 // Проверка ввода  
 if (username.isEmpty() || password.isEmpty()) {  
 Toast.*makeText*(Activity\_Reg.this, "Введите логин и пароль", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // Проверка, существует ли пользователь  
 if (userDatabase.*isUserExist*(username)) {  
 Toast.*makeText*(Activity\_Reg.this, "Пользователь уже существует!", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 // Добавление нового пользователя  
 userDatabase.*addUser*(username, password);  
 Toast.*makeText*(Activity\_Reg.this, "Регистрация успешна!", Toast.*LENGTH\_SHORT*).show();  
  
 // Переход к активити логина  
 Intent loginIntent = new Intent(Activity\_Reg.this, LoginActivity.class);  
 startActivity(loginIntent);  
 }  
 }  
 });  
 }  
}

package com.example.d\_osnovi;  
  
import android.graphics.Color;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.\*;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.content.ContextCompat;  
import androidx.appcompat.app.AlertDialog;  
  
public class DrawingActivity extends AppCompatActivity {  
 private DrawingView drawingView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*drawing\_activity*);  
  
 drawingView = findViewById(R.id.*drawing\_view*);  
  
 // Обработка выбора цвета  
 Button btnChooseColor = findViewById(R.id.*btnChooseColor*);  
 btnChooseColor.setOnClickListener(v -> {  
 showColorPicker(); // Показать диалог выбора цвета  
 });  
  
 // Обработка выбора толщины кисти  
 SeekBar seekBarThickness = findViewById(R.id.*seekBarThickness*);  
 seekBarThickness.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 drawingView.setBrushThickness((float) progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {}  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {}  
 });  
  
 // Обработка выбора карандаша  
 Button btnPencil = findViewById(R.id.*btnPencil*);  
 btnPencil.setOnClickListener(v -> {  
 drawingView.setBrushType("Pencil"); // Установка типа кисти  
 });  
  
 // Обработка выбора кисти  
 Button btnBrush = findViewById(R.id.*btnBrush*);  
 btnBrush.setOnClickListener(v -> {  
 drawingView.setBrushType("Brush"); // Установка типа кисти  
 });  
  
 // Обработка выбора маркера  
 Button btnMarker = findViewById(R.id.*btnMarker*); // Исправленный идентификатор  
 btnMarker.setOnClickListener(v -> {  
 drawingView.setBrushType("Marker"); // Установка типа кисти  
 });  
 }  
  
 private void showColorPicker() {  
 // 20 цветов  
 String[] colors = {  
 "Красный", "Зеленый", "Синий", "Желтый", "Фиолетовый", "Черный", "Белый", "Бирюзовый",  
 "Серый", "Лавандовый", "Оливковый", "Золотой", "Бронзовый", "Серебряный",  
 "Аквамариновый", "Шоколадный"  
 };  
  
 int[] colorValues = {  
 Color.*RED*, Color.*GREEN*, Color.*BLUE*, Color.*YELLOW*, Color.*MAGENTA*,  
 Color.*BLACK*, Color.*WHITE*, Color.*CYAN*, // Бирюзовый  
 Color.*GRAY*, 0xFFB57EDC, // Лаванда  
 0xFF808000, // Оливковый  
 0xFFFFD700, // Золотой  
 0xFFCD7F32, // Бронзовый  
 0xFFC0C0C0, // Серебряный  
 0xFF7FFFD4, // Аквамариновый  
 0xFF8B4513 // Шоколадный  
 };  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Выберите цвет");  
  
 builder.setItems(colors, (dialog, which) -> {  
 int selectedColor = colorValues[which];  
 drawingView.setBrushColor(selectedColor); // Установка выбранного цвета  
 });  
  
 builder.create().show();  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Context;  
import android.graphics.\*;  
import android.util.AttributeSet;  
import android.view.MotionEvent;  
import android.view.View;  
import java.util.ArrayList;  
import java.util.List;  
  
public class DrawingView extends View {  
 private Paint currentPaint; // Используем для текущих настроек  
 private List<Path> paths; // Список всех путей  
 private List<Paint> paints; // Список всех кистей  
  
 public DrawingView(Context context, AttributeSet attrs) {  
 super(context, attrs);  
 init();  
 }  
  
 private void init() {  
 currentPaint = new Paint();  
 currentPaint.setAntiAlias(true);  
 currentPaint.setStyle(Paint.Style.*STROKE*);  
 currentPaint.setStrokeWidth(5f); // Толщина по умолчанию  
  
 paths = new ArrayList<>();  
 paints = new ArrayList<>();  
 }  
  
 public void setBrushType(String type) { // Установка типа кисти  
 switch (type) {  
 case "Pencil":  
 currentPaint.setAlpha(128); // Полупрозрачный карандаш  
 break;  
 case "Brush":  
 currentPaint.setAlpha(255); // Полная насыщенность кисти  
 break;  
 case "Marker":  
 currentPaint.setAlpha(20); // Начальная насыщенность маркера  
 break;  
 }  
 }  
  
 public void setBrushColor(int color) {  
 currentPaint.setColor(color); // Установка цвета  
 }  
  
 public void setBrushThickness(float thickness) {  
 currentPaint.setStrokeWidth(thickness); // Установка толщины  
 }  
  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 // Проходим по всем путям и рисуем с соответствующей кистью  
 for (int i = 0; i < paths.size(); i++) {  
 canvas.drawPath(paths.get(i), paints.get(i));  
 }  
 }  
  
 @Override  
 public boolean onTouchEvent(MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 // Создаем новый Path и добавляем в список  
 Path newPath = new Path();  
 newPath.moveTo(event.getX(), event.getY());  
 paths.add(newPath);  
  
 // Добавляем копию текущего Paint в список, чтобы сохранить настройки  
 paints.add(new Paint(currentPaint));  
 return true;  
  
 case MotionEvent.*ACTION\_MOVE*:  
 paths.get(paths.size() - 1).lineTo(event.getX(), event.getY());  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 break;  
 }  
  
 invalidate(); // Перерисовать View  
 return true;  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.TextUtils;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class LoginActivity extends AppCompatActivity {  
 private EditText edtLogin;  
 private EditText edtPassword;  
 private Button btnLogin;  
 private Button btnRegister;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
  
 // Инициализация элементов интерфейса  
 edtLogin = findViewById(R.id.*etLogin*);  
 edtPassword = findViewById(R.id.*etPassword*);  
 btnLogin = findViewById(R.id.*btnLogin*);  
 btnRegister = findViewById(R.id.*btnRegister1*);  
  
 // Слушатель для кнопки "Вход"  
 btnLogin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String login = edtLogin.getText().toString().trim();  
 String password = edtPassword.getText().toString().trim();  
  
 // Проверка ввода  
 if (TextUtils.*isEmpty*(login) || TextUtils.*isEmpty*(password)) {  
 Toast.*makeText*(LoginActivity.this, "Пожалуйста, введите логин и пароль", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // Проверка логина и пароля  
 if (UserDatabase.*validateUser*(login, password)) {  
 Toast.*makeText*(LoginActivity.this, "Вы вошли в систему", Toast.*LENGTH\_SHORT*).show();  
  
 // Переход на главный экран  
 Intent mainActivityIntent = new Intent(LoginActivity.this, MainActivity.class);  
 startActivity(mainActivityIntent);  
  
 // Завершить текущую активити, чтобы не возвращаться при нажатии кнопки "Назад"  
 finish();  
 } else {  
 Toast.*makeText*(LoginActivity.this, "Ошибка логина или пароля", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 // Слушатель для кнопки "Регистрация"  
 btnRegister.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Переход к активити регистрации  
 Intent registerActivityIntent = new Intent(LoginActivity.this, Activity\_Reg.class);  
 startActivity(registerActivityIntent);  
 }  
 });  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_start\_screen*);  
  
 // Find the login button  
 Button btnLogin = findViewById(R.id.*btnLogin*);  
  
 // Set click listener for the login button  
 btnLogin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Implement your login logic here (e.g., open a login activity)  
 Intent loginIntent = new Intent(MainActivity.this, LoginActivity.class); // Assuming you have a LoginActivity  
 startActivity(loginIntent);  
 }  
 });  
  
 // Find the continue without login button (if it exists)  
 Button btnContinueWithoutLogin = findViewById(R.id.*btnContinueWithoutLogin*); // Adjust ID as needed  
  
 // Check if button exists before setting listener (optional)  
 if (btnContinueWithoutLogin != null) {  
 btnContinueWithoutLogin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Navigate to the main menu  
 Intent intent = new Intent(MainActivity.this, MainMenuActivity.class);  
 startActivity(intent);  
 }  
 });  
 }  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainMenuActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
  
 public void onLearningMaterialsClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, VideoLinksActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
  
 public void onExercisesClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, WorkActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
  
 public void onDrawingToolsClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, DrawingActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainMenuActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
  
 public void onLearningMaterialsClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, VideoLinksActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
  
 public void onExercisesClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, WorkActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
  
 public void onDrawingToolsClick(View view) {  
 Intent intent = new Intent(MainMenuActivity.this, DrawingActivity.class); // Переход к активности рисования  
 startActivity(intent);  
 }  
}

package com.example.d\_osnovi;  
  
import java.util.ArrayList;  
import java.util.List;  
  
// Класс, представляющий запись в базе данных  
class User {  
 private String username;  
 private String password;  
  
 public User(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
}  
  
// Класс базы данных для хранения логинов и паролей  
class UserDatabase {  
 private static List<User> *users*;  
  
 public UserDatabase() {  
 *users* = new ArrayList<>();  
 }  
  
 // Добавление нового пользователя  
 public static void addUser(String username, String password) {  
 if (!*isUserExist*(username)) {  
 *users*.add(new User(username, password));  
 }  
 }  
  
 // Проверка, существует ли пользователь с данным логином  
 public static boolean isUserExist(String username) {  
 for (User user : *users*) {  
 if (user.getUsername().equals(username)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Проверка, совпадает ли логин и пароль  
 public static boolean validateUser(String username, String password) {  
 for (User user : *users*) {  
 if (user.getUsername().equals(username) && user.getPassword().equals(password)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Получение списка всех пользователей (для тестирования)  
 public List<User> getAllUsers() {  
 return *users*;  
 }  
}

package com.example.d\_osnovi;  
  
import java.util.ArrayList;  
import java.util.List;  
  
// Класс, представляющий запись в базе данных  
class User {  
 private String username;  
 private String password;  
  
 public User(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
}  
  
// Класс базы данных для хранения логинов и паролей  
class UserDatabase {  
 private static List<User> *users*;  
  
 public UserDatabase() {  
 *users* = new ArrayList<>();  
 }  
  
 // Добавление нового пользователя  
 public static void addUser(String username, String password) {  
 if (!*isUserExist*(username)) {  
 *users*.add(new User(username, password));  
 }  
 }  
  
 // Проверка, существует ли пользователь с данным логином  
 public static boolean isUserExist(String username) {  
 for (User user : *users*) {  
 if (user.getUsername().equals(username)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Проверка, совпадает ли логин и пароль  
 public static boolean validateUser(String username, String password) {  
 for (User user : *users*) {  
 if (user.getUsername().equals(username) && user.getPassword().equals(password)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Получение списка всех пользователей (для тестирования)  
 public List<User> getAllUsers() {  
 return *users*;  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.ListView;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class VideoLinksActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_video\_links*);  
  
 // Массив с именами видеоуроков  
 String[] videoTitles = {  
 "Видеоурок 1",  
 "Видеоурок 2",  
 "Видеоурок 3",  
 "Видеоурок 4"  
 };  
  
 // Массив с ссылками на видеоуроки (замените на свои ссылки)  
 String[] videoLinks = {  
 "https://youtu.be/MdOLyGxwav4?si=dTfdjun7P2N-smTI",  
 "https://youtu.be/tDGwFqHZng4?si=Vzr1xdYXdUKz4dSU",  
 "https://youtu.be/bZRaZCSqWWc?si=rjxMxevqUQAYJlSu",  
 "https://youtu.be/wC1BFaV3W78?si=v0Dkx9lEzAEA45V5"  
 };  
  
 ListView listView = findViewById(R.id.*video\_links\_list*);  
  
 // Создаем адаптер для отображения списка видеоуроков  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(  
 this, android.R.layout.*simple\_list\_item\_1*, videoTitles);  
  
 listView.setAdapter(adapter);  
  
 // Обработчик нажатий, который открывает ссылку в браузере  
 listView.setOnItemClickListener((parent, view, position, id) -> {  
 String link = videoLinks[position]; // Ссылка по позиции  
 Intent browserIntent = new Intent(Intent.*ACTION\_VIEW*, Uri.*parse*(link));  
 startActivity(browserIntent); // Открываем браузер  
 });  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
import android.widget.ArrayAdapter;  
import android.widget.ListView;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class WorkActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_video\_links*);  
  
 // Массив с именами видеоуроков  
 String[] videoTitles = {  
 "11 упражнений и техник чтобы начать рисовать с нуля"  
 };  
  
 // Массив с ссылками на видеоуроки (замените на свои ссылки)  
 String[] videoLinks = {  
 "https://bool.dev/blog/detail/kak-nachat-risovat-s-nulya-11-uprazhneniy"  
  
 };  
  
 ListView listView = findViewById(R.id.*video\_links\_list*);  
  
 // Создаем адаптер для отображения списка видеоуроков  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(  
 this, android.R.layout.*simple\_list\_item\_1*, videoTitles);  
  
 listView.setAdapter(adapter);  
  
 // Обработчик нажатий, который открывает ссылку в браузере  
 listView.setOnItemClickListener((parent, view, position, id) -> {  
 String link = videoLinks[position]; // Ссылка по позиции  
 Intent browserIntent = new Intent(Intent.*ACTION\_VIEW*, Uri.*parse*(link));  
 startActivity(browserIntent); // Открываем браузер  
 });  
 }  
}

package com.example.d\_osnovi;  
  
import android.content.Context;  
  
import androidx.test.platform.app.InstrumentationRegistry;  
import androidx.test.ext.junit.runners.AndroidJUnit4;  
  
import org.junit.Test;  
import org.junit.runner.RunWith;  
  
import static org.junit.Assert.\*;  
  
*/\*\*  
 \* Instrumented test, which will execute on an Android device.  
 \*  
 \* @see <a href="http://d.android.com/tools/testing">Testing documentation</a>  
 \*/*@RunWith(AndroidJUnit4.class)  
public class ExampleInstrumentedTest {  
 @Test  
 public void useAppContext() {  
 // Context of the app under test.  
 Context appContext = InstrumentationRegistry.*getInstrumentation*().getTargetContext();  
 *assertEquals*("com.example.d\_osnovi", appContext.getPackageName());  
 }  
}

package com.example.d\_osnovi;  
  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
*/\*\*  
 \* Example local unit test, which will execute on the development machine (host).  
 \*  
 \* @see <a href="http://d.android.com/tools/testing">Testing documentation</a>  
 \*/*public class ExampleUnitTest {  
 @Test  
 public void addition\_isCorrect() {  
 *assertEquals*(4, 2 + 2);  
 }  
}