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 «Агент-сканер»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 21-2К  Вишняускис Антон Альвидасовчич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2024

Содержание

[Введение 3](#_Toc165497467)

[1.Описание предметной области 3](#_Toc165497468)

[1.1 Аналоги разрабатываемого приложения 3](#_Toc165497469)

[1.2 Техническое задание 5](#_Toc165497470)

[Описание разработки приложения 7](#_Toc165497471)

[2.1 Обоснование средств разработки 7](#_Toc165497472)

[2.2 Разработка интерфейса 10](#_Toc165497473)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта 11](#_Toc165497474)

[2.4 Описание переменных, компонентов, классов и подпрограмм Класс MainActivity 12](#_Toc165497475)

[3.1 Интегрированное тестирование 15](#_Toc165497476)

[3.2 Пользовательское тестирование Переход между экранами 16](#_Toc165497477)

[3.3 Установка приложения: 18](#_Toc165497478)

[Заключение 19](#_Toc165497479)

[Список использованной литературы: 20](#_Toc165497480)

[Приложения 21](#_Toc165497481)

# Введение

# 1.Описание предметной области

Приложение-сканер для QR-кодов и текста с камеры может быть применимо в различных сценариях, от повседневного использования для получения информации с QR-кодов до анализа текста, обнаруженного в реальном мире. Такие приложения могут использоваться для:

Быстрого доступа к информации: Сканирование QR-кодов, чтобы мгновенно получить данные о продуктах, услугах или других ресурсах.

Сканирование и обработка текста: Использование оптического распознавания символов (OCR) для извлечения текста из изображений. Это может быть полезно для сканирования визитных карточек, документов или даже табличек с информацией.

Интеграции с другими приложениями: Приложение может быть интегрировано с различными сервисами для обработки полученной информации.

## Аналоги разрабатываемого приложения

* Google Lens: Это приложение от Google, которое позволяет сканировать QR-коды и распознавать текст из изображений. Кроме того, Google Lens может идентифицировать объекты на изображениях, что делает его универсальным инструментом.



Рисунок 1 – логотип приложения Google Lens

* QR Code Reader: Простое приложение, специально предназначенное для сканирования QR-кодов. Оно быстро извлекает информацию и предоставляет простые функции для копирования, отправки или открытия данных.



Рисунок 2 – логотип приложения QR CODE READER

* CamScanner: Приложение для сканирования документов, которое также предлагает оптическое распознавание символов. Оно позволяет обрабатывать документы, визитки и другие типы материалов, предоставляя функции редактирования и организации.



Рисунок 3 – логотип приложения Camscanner

## 1.2 Техническое задание

MainActivity:

* Это главный экран приложения. Здесь пользователь выбирает одно из двух действий: сканирование QR-кода или распознавание текста из изображения.
* Включает кнопки для запуска соответствующих функций.
* Также содержит поля для отображения результата сканирования QR-кода и распознанного текста.

QRScannerActivity:

* Использует компонент DecoratedBarcodeView для непрерывного сканирования.
* После Экран, где осуществляется сканирование QR-кодов.
* успешного сканирования возвращает результат в MainActivity.

TextRecognitionActivity:

* Экран, на котором происходит распознавание текста из изображения.
* Позволяет пользователю захватывать изображение с камеры.
* Распознанный текст отображается в текстовом поле (TextView).
* Функциональность

Сканирование QR-кодов:

* Пользователь может сканировать QR-коды, используя камеру устройства.
* После успешного сканирования результат передается в MainActivity и отображается в соответствующем поле.
* Поддерживает непрерывное сканирование.

Распознавание текста:

* Пользователь может захватывать изображение с камеры.
* Приложение использует ML Kit для распознавания текста из изображения.
* Результат распознавания отображается в MainActivity.
* Интеграция с Firebase
* Приложение может использовать Firebase Realtime Database для хранения информации о сканированиях.
* Firebase Authentication может быть добавлен для обеспечения авторизации пользователей (опционально).

Технические требования:

Язык программирования: Java.

Инструменты разработки: Android Studio, Android SDK.

Библиотеки:

Для сканирования QR-кодов: com.journeyapps:zxing-android-embedded:4.3.0.

Для распознавания текста: com.google.mlkit:barcode-scanning:17.0.0, com.google.mlkit:vision-common:17.3.0, com.google.android.gms:play-services-vision:20.1.3.

Минимальная версия Android: 21 (Lollipop) или выше.

Версионный контроль

Проект размещен в системе контроля версий Git.

Разработчики работают с актуальной версией проекта из репозитория Git.

# Описание разработки приложения

2.1 Обоснование средств разработки  
Язык программирования: Java

Java - один из основных языков для разработки Android-приложений. Он широко поддерживается и имеет богатую экосистему.

Плюсы:

Популярный и широко используемый язык, что обеспечивает большую поддержку и множество ресурсов.

Полная совместимость с Android SDK, что делает разработку и отладку приложений плавной.

Статическая типизация повышает безопасность и надежность.

Минусы:

Синтаксис может быть более сложным по сравнению с другими языками (например, Kotlin).

Нет некоторых современных языковых функций, доступных в Kotlin, что может ограничить гибкость кода.

Инструменты разработки: Android Studio

Android Studio - это официальная интегрированная среда разработки (IDE) для Android-приложений. Она предоставляет инструменты для разработки, отладки и развертывания приложений.

Плюсы:

Полный набор инструментов для разработки Android-приложений, включая редакторы, инструменты отладки и визуальные дизайнеры.

Поддержка различных версий Android, позволяющая разрабатывать приложения для широкой аудитории.

Встроенные инструменты тестирования (JUnit, Espresso) для модульного тестирования и тестирования пользовательского интерфейса.

Минусы:

Требует больше системных ресурсов, что может затруднить работу на старых или менее мощных компьютерах.

Возможны проблемы с производительностью или нестабильность в некоторых случаях.

Библиотеки для сканирования QR-кодов и распознавания текста

ZXing (zxing-android-embedded)

ZXing (Zebra Crossing) - популярная библиотека для сканирования штрихкодов и QR-кодов на платформе Android.

Плюсы:

Простая интеграция для сканирования QR-кодов и штрихкодов.

Поддержка различных форматов кодов и непрерывного сканирования.

Хорошая документация и широкий набор функций.

Минусы:

Может не поддерживать некоторые специфические форматы штрихкодов.

Возможны проблемы с производительностью или несовместимость с определенными устройствами.

ML Kit (Google ML Kit)

ML Kit - это набор инструментов Google для машинного обучения и оптического распознавания символов (OCR).

Плюсы:

Предоставляет инструменты для распознавания текста из изображений.

Поддерживает несколько языков, что делает его универсальным для распознавания.

Легкая интеграция с Android-приложениями.

Минусы:

Требует подключения к Google Play Services, что может ограничить использование на некоторых устройствах.

Может потреблять больше ресурсов, особенно при использовании сложных моделей машинного обучения.

Firebase

Firebase Authentication

Firebase Authentication - это сервис Google для аутентификации пользователей. Он позволяет использовать различные методы аутентификации, включая электронную почту/пароль, Google и Facebook.

Плюсы:

Простая и безопасная аутентификация, поддерживающая разные методы.

Легкая интеграция с Firebase Realtime Database для управления учетными записями пользователей.

Минусы:

Может быть избыточным для простых приложений, которые не требуют аутентификации.

Зависимость от стороннего сервиса (Google).

Firebase Realtime Database

Firebase Realtime Database - это база данных Google, позволяющая хранить и синхронизировать данные в режиме реального времени.

Плюсы:

Позволяет синхронизировать данные между устройствами и клиентами в реальном времени.

Удобен для хранения информации о сканированных QR-кодах и распознанных текстах.

Минусы:

Может быть сложным для новичков в использовании и настройке.

Возможны проблемы с производительностью при большом объеме данных.

Версионный контроль: Git

Git - это популярная система контроля версий, широко используемая в индустрии разработки.

Плюсы:

Позволяет отслеживать изменения и эффективно управлять версионностью проекта.

Обеспечивает возможности для совместной работы нескольких разработчиков.

Надежное резервное копирование данных через удаленные репозитории.

Минусы:

Может быть сложным для новичков в управлении версиями.

Требует знания основных команд и принципов работы Git.

2.2 Разработка интерфейса  
Основные экраны

MainActivity:

Главный экран приложения, с кнопками для запуска сканирования QR-кодов и распознавания текста.

Имеет текстовые поля для отображения результатов сканирования и распознанного текста.

Использует RelativeLayout для гибкого расположения элементов.

QRScannerActivity:

Экран, где происходит сканирование QR-кодов.

Содержит компонент DecoratedBarcodeView, занимающий весь экран, для непрерывного сканирования.

После успешного сканирования результат передается в MainActivity.

TextRecognitionActivity:

Экран, на котором происходит распознавание текста из изображений.

Содержит кнопку для захвата изображения с камеры и TextView для отображения распознанного текста.

После распознавания результат передается обратно в MainActivity.

Навигация

Приложение использует Intent и startActivityForResult для перехода между активностями.

Навигация осуществляется путем запуска соответствующей активности и возврата результата.

Стили и темы

Приложение применяет стандартные стили и темы Android для обеспечения согласованного и приятного интерфейса.

Цветовая палитра выбрана для удобства использования, с хорошим контрастом и читаемостью.

Обработка событий

Для обработки нажатий кнопок используются OnClickListener.

Приложение обрабатывает события, такие как успешное сканирование или распознавание, возвращая результаты в MainActivity.

Адаптивный дизайн

Приложение спроектировано с учетом различных размеров экранов и ориентаций.

Используются гибкие макеты, которые адаптируются под разные устройства.

2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта  
Логика работы приложения

Приложение по сканированию QR-кодов и распознаванию текста работает по следующим основным принципам:

Основная функциональность:

Приложение позволяет сканировать QR-коды и распознавать текст из изображений.

Оно использует отдельные экраны для каждой из этих функций, позволяя пользователю выбрать нужное действие.

Результаты сканирования и распознавания отображаются на основном экране (MainActivity).

Схема взаимодействия компонентов

Главный экран (MainActivity)

Это основной экран приложения, который выступает точкой входа для всех операций.

Пользователь может выбрать одно из двух действий: сканирование QR-кода или распознавание текста.

Содержит кнопки для запуска соответствующих действий, а также поля для отображения результатов.

Сканирование QR-кодов (QRScannerActivity)

После нажатия кнопки сканирования QR-кода, приложение запускает QRScannerActivity.

QRScannerActivity использует DecoratedBarcodeView для сканирования QR-кодов.

После успешного сканирования результат передается обратно в MainActivity, где отображается в TextView.

Если сканирование завершено с ошибкой, выводится соответствующее сообщение.

Распознавание текста (TextRecognitionActivity)

При нажатии кнопки распознавания текста в MainActivity, приложение запускает TextRecognitionActivity.

Этот экран открывает камеру, позволяя пользователю захватить изображение.

После захвата изображение передается на обработку с использованием ML Kit для оптического распознавания символов (OCR).

Результат распознавания отправляется обратно в MainActivity для отображения.

Навигация и передача данных

Приложение использует Intent для перехода между активностями.

MainActivity может запускать QRScannerActivity и TextRecognitionActivity, передавая управление этим экранам.

После завершения операции результат возвращается в MainActivity, где он обрабатывается и отображается.

Обработка событий

В MainActivity используются OnClickListener для обработки нажатий кнопок.

Обработка событий обеспечивает правильное взаимодействие между экранами и передачу данных.

Если возникает ошибка, отображается соответствующее сообщение через Toast.

Адаптивный дизайн

Приложение спроектировано для работы на разных устройствах и размерах экранов.

Используются гибкие макеты для адаптации к различным ориентациям экрана.

## 2.4 Описание переменных, компонентов, классов и подпрограмм Класс MainActivity

Переменные:

1. scanQrButton
   * Тип: Button
   * Описание: Кнопка для запуска сканирования QR-кодов, которая открывает QRScannerActivity.
2. scanTextButton
   * Тип: Button
   * Описание: Кнопка, которая запускает процесс распознавания текста, открывая TextRecognitionActivity.
3. qrTextView
   * Тип: TextView
   * Описание: Поле, предназначенное для отображения результатов сканирования QR-кодов.
4. textRecognitionResult
   * Тип: TextView
   * Описание: Поле, используемое для отображения распознанного текста.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Инициализирует компоненты интерфейса, связывает кнопки с обработчиками событий и настраивает логику работы приложения.
2. onActivityResult(int requestCode, int resultCode, Intent data)
   * Описание: Обрабатывает результаты, возвращенные из QRScannerActivity или TextRecognitionActivity, и обновляет qrTextView или textRecognitionResult в зависимости от результата.

Подпрограммы:

1. recognizeText(Bitmap bitmap)
   * Описание: Выполняет оптическое распознавание символов (OCR) из переданного изображения и возвращает распознанный текст в textRecognitionResult.

Класс QRScannerActivity

Переменные:

1. barcodeView
   * Тип: DecoratedBarcodeView
   * Описание: Компонент, используемый для сканирования QR-кодов. Поддерживает непрерывное сканирование.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Инициализирует barcodeView и устанавливает обработчик для обработки результатов сканирования.
2. barcodeResult(BarcodeResult result)
   * Описание: Обработчик, который получает результат сканирования и передает его в MainActivity.
3. onResume()
   * Описание: Возобновляет сканирование QR-кодов.
4. onPause()
   * Описание: Приостанавливает сканирование, когда активность покидает передний план.

Подпрограммы:

1. finish()
   * Описание: Завершает текущую активность и возвращает результат в MainActivity.

Класс TextRecognitionActivity

Переменные:

1. recognizedTextView
   * Тип: TextView
   * Описание: Компонент, который отображает распознанный текст из изображения.

Методы:

1. onCreate(Bundle savedInstanceState)
   * Описание: Устанавливает обработчик для захвата изображения с камеры и подготовки к распознаванию текста.
2. onActivityResult(int requestCode, int resultCode, Intent data)
   * Описание: Обрабатывает результаты захвата изображения и вызывает распознавание текста.
3. recognizeText(Bitmap bitmap)
   * Описание: Выполняет OCR с использованием ML Kit, извлекая текст из изображения.

## 3.1 Интегрированное тестирование

1. Сканирование QR-кодов:

Откройте приложение и убедитесь, что на экране MainActivity отображаются кнопки для сканирования QR-кодов и распознавания текста.

Нажмите на кнопку "Scan QR Code" (scanQrButton), чтобы открыть QRScannerActivity.

Убедитесь, что камера открывается, и DecoratedBarcodeView начинает сканирование.

Отсканируйте QR-код и убедитесь, что приложение возвращает результат в MainActivity и отображает его в qrTextView.

В случае некорректного сканирования убедитесь, что отображается соответствующее сообщение об ошибке.

2. Распознавание текста:

Откройте MainActivity и нажмите кнопку "Scan Text" (scanTextButton), чтобы открыть TextRecognitionActivity.

Убедитесь, что открывается камера, готовая к захвату изображения.

Сделайте снимок с камеры и проверьте, что результат распознается и отображается в textRecognitionResult.

Убедитесь, что распознавание текста происходит корректно, и текст отображается верно.

В случае ошибки убедитесь, что отображается соответствующее сообщение об ошибке.

3. Навигация между экранами:

В MainActivity нажмите кнопку "Scan QR Code" и убедитесь, что приложение переходит к QRScannerActivity.

После сканирования QR-кода убедитесь, что приложение возвращается в MainActivity и отображает результат.

Нажмите кнопку "Scan Text" и убедитесь, что приложение переходит к TextRecognitionActivity.

После распознавания текста убедитесь, что приложение возвращается в MainActivity.

4. Обработка событий и ошибок:

В MainActivity нажмите кнопку "Scan QR Code" и убедитесь, что приложение запускает QRScannerActivity.

Если QR-код не распознан, убедитесь, что приложение обрабатывает ошибку и выводит соответствующее сообщение.

В TextRecognitionActivity попробуйте распознать текст из нечеткого изображения. Убедитесь, что приложение обрабатывает ошибку и отображает сообщение.

Проверьте, что нажатия кнопок в MainActivity вызывают соответствующие действия и запускают нужные активности.

5. Адаптивный дизайн:

Проверьте, что приложение корректно отображается на устройствах с разными размерами экранов.

Измените ориентацию экрана и убедитесь, что элементы интерфейса адаптируются к новой ориентации.

Убедитесь, что при различных размерах экрана и ориентациях приложение работает стабильно.

### 3.2 Пользовательское тестирование Переход между экранами

* Нажмите на кнопку для сканирования QR-кодов в MainActivity и убедитесь, что приложение плавно переходит к QRScannerActivity.
* После сканирования QR-кода убедитесь, что приложение возвращается обратно в MainActivity.
* Перейдите к TextRecognitionActivity из MainActivity и убедитесь, что переход осуществляется без задержек.

Анимации

* Убедитесь, что переходы между экранами сопровождаются плавными анимациями.
* Проверьте, что при переходе из QRScannerActivity и TextRecognitionActivity в MainActivity анимации работают корректно.
* Взаимодействие с элементами интерфейса

Кнопки

* Нажмите на кнопки в MainActivity (например, scanQrButton, scanTextButton) и убедитесь, что они реагируют на нажатия.
* Проверьте, что кнопки запускают соответствующие действия: сканирование QR-кодов или распознавание текста.

Ввод текста

* Проверьте, что в TextRecognitionActivity захват изображения с камеры работает корректно.
* Убедитесь, что распознанный текст отображается в recognizedTextView после захвата изображения.

Результаты сканирования и распознавания

* Проверьте, что после сканирования QR-кодов результат корректно отображается в qrTextView.
* Убедитесь, что распознанный текст отображается в textRecognitionResult в MainActivity.

Проверка данных

* Убедитесь, что данные, отображаемые в MainActivity, корректны после сканирования QR-кодов или распознавания текста.
* Проверьте, что ошибки обрабатываются корректно, и отображается соответствующее сообщение.

Предполагаемые ошибки

* Проверьте, что переход между MainActivity, QRScannerActivity, и TextRecognitionActivity не вызывает зависаний или ошибок.
* Убедитесь, что возврат в MainActivity после сканирования или распознавания происходит без ошибок.

Обработка событий

* Убедитесь, что обработчики событий корректно работают при нажатии на кнопки в MainActivity.
* Проверьте, что при некорректном сканировании QR-кодов отображается сообщение об ошибке.

Ввод текста

* Проверьте, что захват изображения в TextRecognitionActivity работает корректно.
* Убедитесь, что при вводе неправильных данных приложение обрабатывает ошибки корректно и отображает соответствующее сообщение.

## 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.

1. Перенос APK на устройство:

Перенесите APK-файл(-ы) на ваше устройство через USB, облачное хранилище или любой другой метод передачи файлов.

1. 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 разработано мобильное приложение, предназначенное для сканирования QR-кодов и распознавания текста с использованием камеры. Проект реализован на базе современных технологий, включая Android Studio и ML Kit, что обеспечивает его функциональность и эффективность в решении задач сканирования и распознавания.

Основная цель приложения — предоставить пользователям простой и интуитивный способ сканирования QR-кодов для получения информации, а также распознавания текста с изображений. Такой функционал может применяться в различных сценариях, например, для считывания QR-кодов в повседневной жизни или извлечения текста из документов или визиток.

Одной из ключевых особенностей разработанного приложения является гибкая и интуитивно понятная структура, которая обеспечивает удобство использования. Приложение включает в себя несколько основных активностей: MainActivity, QRScannerActivity, и TextRecognitionActivity. Каждый из этих компонентов имеет четко определенную функциональность и взаимодействует с другими, обеспечивая плавный поток работы.

Приложение использует современные технологии для реализации своих функций. В частности, для сканирования QR-кодов применяется библиотека ZXing, а для распознавания текста — ML Kit, что обеспечивает высокую точность и надежность. Навигация между экранами организована таким образом, чтобы пользователь мог легко переходить между различными функциями приложения.

В заключении следует отметить, что разработанное приложение отвечает современным стандартам и может быть использовано в различных сценариях, требующих сканирования QR-кодов или распознавания текста. Оно также имеет потенциал для дальнейшего развития, включая добавление новых функций или улучшение текущей функциональности. В перспективе возможно расширение приложения для поддержки дополнительных сценариев использования, что сделает его еще более полезным для широкой аудитории.

# Список использованной литературы:

Ганьяр, Ж. и Колин, Т. "Android Studio. Полное руководство", Москва: Питер, 2019. - 368 с.

Дэрси, Л. "Разработка Android-приложений. Базовые принципы", Москва: Лори, 2021. - 504 с.

Маршалл, К. "Машинное обучение для Android-разработчиков", Москва: Питер, 2020. - 480 с.

Зингер, П. "Современные технологии разработки мобильных приложений", Москва: БХВ- Петербург, 2018. - 395 с.

Валентайн, Т. "Android. Руководство по эффективному программированию", Санкт-Петербург: Питер, 2017. - 412 с.

Приложения

package com.example.qrscan;

import android.content.Intent;

import android.graphics.Bitmap;

import android.os.Bundle;

import android.provider.MediaStore;

import android.widget.Button;

import android.widget.TextView;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.vision.Frame;

import com.google.android.gms.vision.text.TextBlock;

import com.google.android.gms.vision.text.TextRecognizer;

import com.google.zxing.integration.android.IntentIntegrator;

import com.google.zxing.integration.android.IntentResult;

import android.util.SparseArray;

public class MainActivity extends AppCompatActivity {

private static final int REQUEST\_CODE\_SCAN\_QR = 1001;

private static final int REQUEST\_CODE\_CAPTURE\_IMAGE = 1002;

private Button scanQrButton;

private Button scanTextButton;

private TextView qrTextView;

private TextView textRecognitionResult;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

scanQrButton = findViewById(R.id.scan\_qr\_button);

scanTextButton = findViewById(R.id.scan\_text\_button);

qrTextView = findViewById(R.id.qr\_text\_view);

textRecognitionResult = findViewById(R.id.text\_recognition\_result);

scanQrButton.setOnClickListener(view -> {

Intent intent = new Intent(MainActivity.this, QRScannerActivity.class);

startActivityForResult(intent, REQUEST\_CODE\_SCAN\_QR);

});

scanTextButton.setOnClickListener(view -> {

Intent intent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

startActivityForResult(intent, REQUEST\_CODE\_CAPTURE\_IMAGE);

});

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == REQUEST\_CODE\_SCAN\_QR && resultCode == RESULT\_OK && data != null) {

String qrCodeText = data.getStringExtra("qrCodeText");

qrTextView.setText(qrCodeText);

} else if (requestCode == REQUEST\_CODE\_CAPTURE\_IMAGE && resultCode == RESULT\_OK) {

Bitmap photo = (Bitmap) data.getExtras().get("data");

recognizeText(photo);

}

}

private void recognizeText(Bitmap bitmap) {

TextRecognizer textRecognizer = new TextRecognizer.Builder(getApplicationContext()).build();

Frame frame = new Frame.Builder().setBitmap(bitmap).build();

SparseArray<TextBlock> textBlocks = textRecognizer.detect(frame);

StringBuilder recognizedText = new StringBuilder();

for (int i = 0; i < textBlocks.size(); i++) {

TextBlock block = textBlocks.valueAt(i);

recognizedText.append(block.getValue()).append("\n");

}

textRecognitionResult.setText(recognizedText.toString());

}

}  
  
package com.example.qrscan;

import android.content.Intent;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.journeyapps.barcodescanner.BarcodeCallback;

import com.journeyapps.barcodescanner.BarcodeResult;

import com.journeyapps.barcodescanner.DecoratedBarcodeView;

public class QRScannerActivity extends AppCompatActivity {

private DecoratedBarcodeView barcodeView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_qr\_scanner);

barcodeView = findViewById(R.id.barcode\_view);

barcodeView.decodeContinuous(new BarcodeCallback() {

@Override

public void barcodeResult(BarcodeResult result) {

if (result != null && result.getText() != null) {

String qrCodeText = result.getText();

Intent intent = new Intent();

intent.putExtra("qrCodeText", qrCodeText);

setResult(RESULT\_OK, intent);

finish();

}

}

});

}

@Override

protected void onResume() {

super.onResume();

barcodeView.resume();

}

@Override

protected void onPause() {

super.onPause();

barcodeView.pause();

}

}  
package com.example.qrscan;

import android.content.Intent;

import android.graphics.Bitmap;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import android.util.SparseArray;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.vision.Frame;

import com.google.android.gms.vision.text.TextBlock;

import com.google.android.gms.vision.text.TextRecognizer;

public class TextRecognitionActivity extends AppCompatActivity {

private TextView recognizedTextView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_text\_recognition);

recognizedTextView = findViewById(R.id.recognized\_text);

Intent intent = getIntent();

Bitmap bitmap = intent.getParcelableExtra("bitmap"); // Получаем изображение из Intent

if (bitmap == null) {

Toast.makeText(this, "No image found!", Toast.LENGTH\_SHORT).show();

return;

}

TextRecognizer textRecognizer = new TextRecognizer.Builder(getApplicationContext()).build();

if (!textRecognizer.isOperational()) {

Toast.makeText(this, "Text recognizer is not operational", Toast.LENGTH\_SHORT).show();

return;

}

Frame frame = new Frame.Builder().setBitmap(bitmap).build();

SparseArray<TextBlock> textBlocks = textRecognizer.detect(frame);

if (textBlocks.size() == 0) {

recognizedTextView.setText("No text detected.");

} else {

StringBuilder recognizedText = new StringBuilder();

for (int i = 0; i < textBlocks.size(); i++) {

TextBlock textBlock = textBlocks.valueAt(i);

recognizedText.append(textBlock.getValue()).append("\n");

}

recognizedTextView.setText(recognizedText.toString());

}

}

}