Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900

**Issue 4/2021**

# POSSIBILITIES FOR DEVELOPING AND IMPLEMENTING A MOBILE APPLICATION FOR RECOGNIZING THE SHAPE OF THE ENVIRONMENT, TEXT, AND READING QR CODES USING THE ANDROID CAMERAX FRAMEWORK AND THE MACHINE LEARNING KIT

***Miljan PELEŠ[1], Svetlana JEVREMOVIĆ[1], Aleksandar SIMOVIĆ[1], Aleksandra HADŽIĆ[1]***
***[1] Information Technology School, Cara Dušana 34, 11070 Belgrade, Serbia, Emails: miljan4718@its.edu.rs; svetlana.jervremovic@its.edu.rs; aleksandar.simovic@its.edu.rs; aleksandra44820@its.edu.rs***

**How to cite:** PELEŠ, M., JEVREMOVIĆ, S., SIMOVIĆ, A., & HADŽIĆ, A. (2021). "Possibilities for Developing and Implementing a Mobile Application for Recognizing the Shape of the Environment, Text, and Reading QR Codes Using the Android Camerax Framework and the Machine Learning KIT." *Annals of Spiru Haret University. Economic Series*, *21*(4), 163-179, doi: https://doi.org/10.26458/2148

**Abstract**

*The advancement and development of digital technologies have resulted in the need to network various devices at the application level. Wireless communication between devices via the Internet has opened a plethora of possibilities for enhancing user capabilities. We are witnessing dizzying changes in computer technology, and we can conclude that the device's purpose is no longer narrowly defined. The mobile phone is evolving into a personal computer, innovative features are being added to today's televisions, and cameras can process and send photos. These are merely a few examples of universal electronic devices. Of course, for the device to perform all these functions, adequate hardware infrastructure integrated into the device itself is required, as is the fundamental software component that connects user*

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900

**Issue 4/2021**

*operations and the components themselves - the operating system. This paper's operating system under consideration is the Android operating system, which is currently the most popular operating system for smart devices.*

**Keywords:** *machine learning; Android; QR codes.*

**JEL Classification:** O14

## Introduction

The paper aims to analyze the possibilities of combining the Android CameraX framework with the Machine Learning Kit to create and implement a mobile application for recognizing the shape of the environment, text, and QR codes. To fully describe the technology that will be used for this project, the concept of machine learning (*MLKit*) and *CameraX* software must be introduced. The first section, i.e., the second chapter of this paper, will cover the theory of the Android operating system and a description of the functionalities and libraries used in the project itself. The second section of the paper will cover a theoretical introduction to machine learning (*Machine Learning Kit*), machine learning applications within the Android operating system, and an explanation of all machine learning tools. The third section will explain *CameraX* Framework support in Android systems and introduce the understanding of the solution's implementation. It discusses how to use an Android device's camera and explains how it works.

## 1. *Android Applications and Operating System*

Google's Android is an open-source operating system based on the Linux kernel. It is intended for various devices, ranging from mobile phones, where it is most prevalent, to smartwatches, fitness devices, home appliances, and televisions. This applicability on a wide range of devices is due to the Android operating system's ability to separate the hardware from its software. Android is based on the direct manipulation of objects on the screen through touch inputs, as evidenced by the appearance of touch screen mobile phones. This chapter will provide a brief overview of Android's history, the structure of an Android application, the operating system's architecture, and system applications. Finally, a brief overview of the libraries used in this project will be provided, followed by a more detailed explanation in Chapters Four and Five.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



Using the language, you can create applications for the Android platform in a variety of ways: C/C++, Java/XML, Basin/XML and HTML5.

When the C/C++ programming languages are used for application development, we refer to them as "native applications" (native applications). These are the most fundamental applications. To bring this up to speed, consider the software stack architecture. The software stack is divided into three basic levels:

- HAL (Hardware Abstraction Layer)
- PAL (Platform Abstraction Layer)
- Application Layer (AL)

Native applications run on the platform layer or the layer between the PAL and AL, known as middleware, and serve to translate a user application into a machine-readable language. This approach is required when creating the performance or managing the resources so that only a small number of developers deal with this method of developing applications.

The most common type of Android application is written in the Java / XML or Kotlin / XML language, and it is programmed on the application layer of the software stack. Java is a virtual machine-required language, and Android employs the Android Runtime (ART) virtual machine for this purpose. This virtual machine has been in use since the Android operating system's version 5.0 (Lollipop) replaced the Dalvik virtual machine. The Dalvik virtual machine does not accept standard Java class files and instead uses its format: Dalvik Executable (DEX). In contrast to standard Java applications, which contain multiple class files, DEX consolidates all class files into a single. dex file. The main distinction between the ART virtual machine and the Dalvik VM is that ART translates a substantial portion of the executable byte code into machine native language during installation, so restarting the application does not necessitate a complete translation of the applications.dex executable byte code into machine language. ML (*Extensible Markup Language*) is used to create resource files as well as files that contain some functional data and contracts within the application.

The Android platform is made up of a variety of components that can be divided into six categories:

- Linux kernel
- HAL - A hardware abstraction layer
- Native Libraries (source libraries)
- Android Runtime (executable environment)
- Java API framework

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



This layered structure is what sets the Android operating system apart. This structure implies that the operating system comprises several layers that build on top of one another. Each layer has specific functions described through its interface to the higher layer; more specifically, each layer implements the higher layer's coupling.

**2. *Technology Used for Model Development***

When it is discussed about the programming language for developing Android applications on which this paper is based, it is referred to the Java programming language, which is used to write the source code. It is written in text files with the.java extension, compiled and translated into byte code, then executed using the previously mentioned virtual machine. The difference between Android and regular Java applications is that ART virtual machines do not read traditional bytecode but instead require compiled binary code in the DEX format. .jar Java archives contain both classic.java and compiled. class files. The Android executable environment does not recognize Java archives but bundles all files from a single application in a particular format known as an Android Application Package (APK) file with the extension.

A Java Development Kit (JDK) must be installed on the computer for an Android application to be successfully created. Following that, the Java source code must be translated into a form understandable by ART, which necessitates using a special compiler. Other tools, such as linkers, debuggers, and libraries with built-in functions, are required to run the code. All these components are bundled together under the banner of the Android Software Development Kit (SDK), which consists of the following units: SDK Platform, SDK Tools, Sample Apps, Documentation and Android Support.

Following these prerequisites comes the Android platform's ultimate development environment, an Android Studio application.

Android Studio is an integrated development environment that manages the complexities of developing Android applications. It is built on the IntelliJ IDEA development platform. Android Studio makes it simple to test, debug, and, most importantly, build applications. It does all the work of compiling code into DEX binary format and connecting the code to the JDK and Android API libraries. It includes predefined functions and special libraries for developing Android apps for a variety of devices. It is distinguished by a detailed overview of the possibilities for interaction with the developer, a presentation of the application's structure, and

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



a comprehensive tool for testing applications. It supports the entire application development process, from the design of the user interface to the software solution and the ability to oversee databases. Android Studio is now regarded as one of the most comprehensive software environments for developing user applications.

This project will use the most recent version of the environment, Android Studio Arctic Fox 2020.3.1, as illustrated in Figure 1.

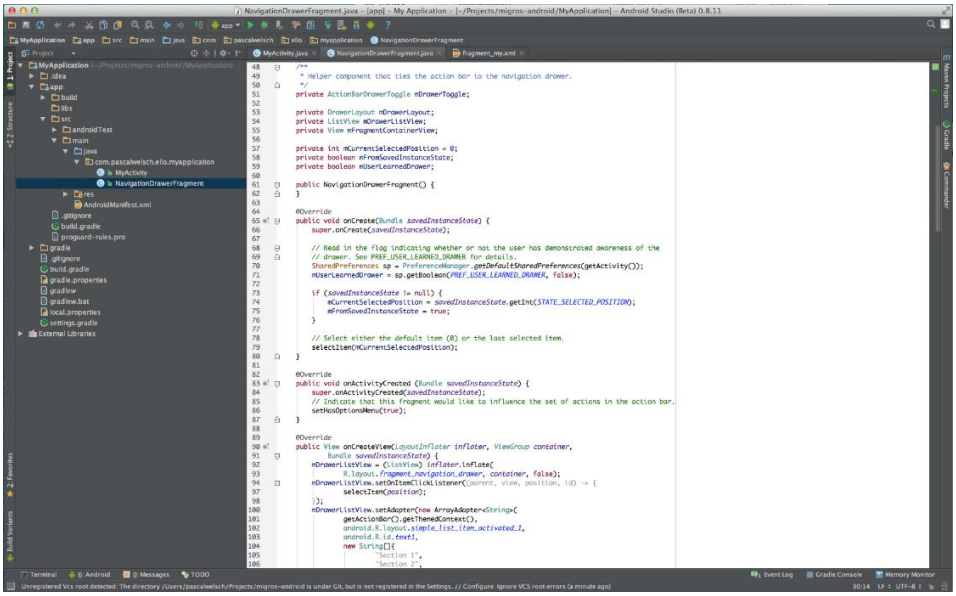

**Fig. 1. Android Studio Development Framework**

### 3. *Machine Learning*

At the outset, it will be explained what machine learning is, how it works, and what applications it has. Machine Learning (ML) is a branch of statistics and computer science that enables a computer to learn how to perform a specific task without being pre-programmed.

Machine learning is based on the idea that generic algorithms can show something interesting about a set of data without the need to write custom code for the problem. Rather than writing code, data is fed into the generic algorithm, generating its logic based on the data.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



The classification algorithm, for example, is one type of such algorithm. It can store data in various groups. Without changing a single line of code, the same classification algorithm used to recognize handwritten numbers could be used to classify emails as "spam" or "not spam." The algorithm is the same, but different training data are fed into it, resulting in different classification logic.

The machine learning algorithm is a black box, creating its logic based on data. Many of these types of classification algorithms are covered under the umbrella term "machine learning." Machine learning is divided into two categories: supervised learning and unsupervised learning. The main distinction is whether the samples are labelled, whether the computer is told which (for example) images contain the terms we want to recognize or whether we let it try to understand the structure of the input.

Machine learning has a wide range of applications in a variety of industries.

- Text categorization based on topic, expressed feelings and/or attitudes, and the like
- Text machine translation Understanding has spoken language
- Image face recognition
- Segmenting the market
- Observing the use of various applications
- Autonomous vehicles (*self-driving cars*, for example) and many more.

*ML Kit* is a mobile *SDK* that enables Google's machine learning tool within Android applications. Google's machine learning tool is free, giving developers maximum flexibility in its application.

*The ML Kit API* is compatible with all devices with an *API* level greater than sixteen, specifically all Android devices running Android version 4.1 or higher. This range enables Google machine learning tools to be used on 99.8% of current devices, effectively covering the entire market.

The ML Kit comprises ten completed APIs that allow you to work with various areas of machine learning. This paper will go over the following APIs: Recognition of text, Face recognition, Detection of body position, Image recognition for selfies, Scannable barcodes, Image tagging, Detecting and tracking objects and Text recognition in digital form.

Android applications that read data encoded in standard bar code formats can be easily created by using the bar code scanning API. Scanning bar codes is done on the user's device and does not require an internet connection.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900
Issue 4/2021

Bar codes are a quick and effortless way to transfer data from the real world to a mobile application. For example, a 2D format such as a QR code, contact information, or Wi-Fi data can be encoded. When the user scans the bar code, the ML Kit automatically processes the scanned data, allowing the application being developed to provide answers intelligently.

The following reading formats are supported: Codabar, Code 39, Code 93, Code 128, EAN-8, EAN-13, ITF, UPC-A, UPC-E, Aztec, Data Matrix, PDF417 and QR Code.

The barcode scanning API supports scanning all barcode formats simultaneously without specifying which format is required. Marking a specific format, on the other hand, speeds up the scanning process. Figure 2 depicts a processing example.

The following barcode formats are not supported:

- 1D bar codes with only one character ITF bar codes with fewer than six characters
- FNC2, FNC3, or FNC4 bar code formats
- In ECI mode, QR codes are generated

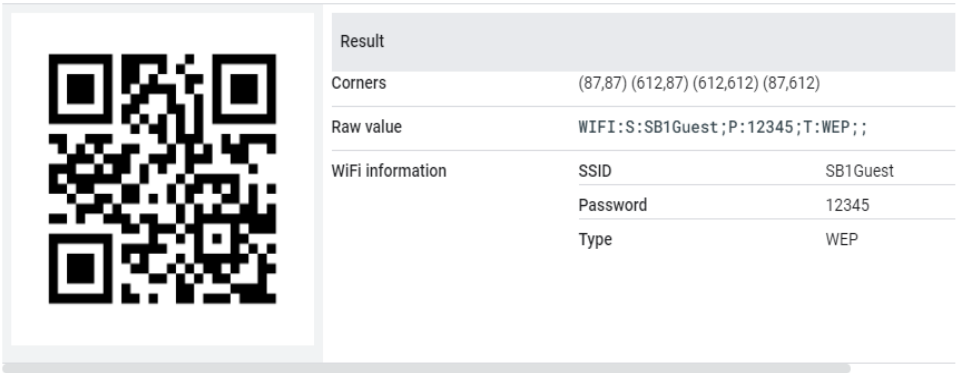


**Fig. 2. Example of generation**

**4. *Camerax Framework***

*CameraX Framework* is a collection of libraries designed to make working with Android devices' cameras easier. Developing applications that use a camera has always been demanding, but these applications are also the most dynamic on the user side. CameraX allows us to use the API that runs on most Android devices, with background support up to Android 5.0. (API Level 21).

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900
Issue 4/2021

CameraX extends the capabilities of the old Camera2 API, which has been updated to collaborate with the Android camera. Prior to the Camera2 API, the original Camera API was used, which is now deprecated and replaced by the Camera2 API, which is still in use.

CameraX makes all functionalities easily accessible and allows for their implementation in a small number of lines of code. In addition, the old API's compatibility issue with older generation devices has been resolved.

In the following ways, this framework improves the developer experience:

1. Simple to use - CameraX has introduced a new feature in Android called use cases, which allows developers to focus on the tasks at hand rather than configuring the device. There are two fundamental applications:

a. Preview - This function displays the image on the screen (camera view)

b. Image Analysis - Gain access to the processing of individual camera frames and send them to custom algorithms. The ML Kit will be used in the upcoming Android application.

c. Image Capture - Maintains high-quality images.

These use cases are compatible with all Android devices running version 5.0 or higher.

2. Device consistency - Managing the consistency of camera behaviour in an app is a challenging task. Numerous functionalities must be considered, including aspect ratio, orientation, rotation, image preview size, and high-resolution image size. CameraX makes it simple to work with all the cameras, as mentioned earlier behaviours.

3. New camera experience - CameraX includes an optional add-on called "Extensions," which gives you access to the same features and capabilities as the built-in camera app on a specific Android device. In other words, devices with HDR, Night, Portrait, Beauty, and other camera modes can be used with the CameraX API

### 4.1 Architecture

The architecture of the CameraX Framework will be described in detail below, including its structure, how to work with the API, how to use the lifecycle, and how to combine use cases.

Developers use CameraX to access the device's camera via a "use case." Currently, the following use cases are available:

- Preview - Uses the PreviewView class to display the image on the screen (camera view).

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



- Image Analysis - Gain access to the processing of individual camera frames and send them to custom algorithms such as ML Kit.
- Image Capture - Maintains high-quality images.

Usage scenarios can be combined. For example, an application can use the Preview case to show a user an image that the camera sees, the Image Analysis case to determine if people in the picture are laughing, and the Image Capture case to take a picture all simultaneously time.

A few things must be specified when working with the *CameraX* library:

- Use case with the necessary configuration
- How to manage feedback
- Camera execution flow, such as when to activate the camera and when to capture and process data

The set () method is used to configure the usage case, followed by the *build* () method. Each use case has its API, such as the *takePicture* () method in the "Image Capture" use case.

*CameraX* uses life cycles to determine when a camera should open, when an image capture session should be started, and when to steal to interrupt and shut down camera execution.

When working with an Android camera, the CAMERA permission must be added to the application. Except for devices running Android version 10.0 and higher, *WRITE_EXTERNAL_STORAGE* is required to save images to files. During application execution, permissions must be requested from the user (requesting runtime permissions).

*CameraX* must meet the following minimum requirements to run on the device:

Level 21 of the Android API

• *FragmentActivity* or *AppCompatActivity* is required for life cycle activities in Android Architecture Components 1.1.1.

*CameraX* must be included in the *build.gradle* files to be used as part of an Android project.

The following line of code must be added to the project's *build.gradle* file:

```
allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

**Fig. 3. Code added to the project's build.gradle file**

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



The following code must be inserted into the "Android" block:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    // For Kotlin projects
    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

**Fig. 4. Code inserted into the „Android" block**

Each module that is used must be added to the application build.gradle file:

```
dependencies {
  // CameraX core library using the camera2 implementation
  def camerax_version = "1.0.1"
  // The following line is optional, as the core library is included indirectly by camera-camera2
  implementation "androidx.camera:camera-core:${camerax_version}"
  implementation "androidx.camera:camera-camera2:${camerax_version}"
  // If you want to additionally use the CameraX Lifecycle library
  implementation "androidx.camera:camera-lifecycle:${camerax_version}"
  // If you want to additionally use the CameraX View class
  implementation "androidx.camera:camera-view:1.0.0-alpha27"
  // If you want to additionally use the CameraX Extensions library
  implementation "androidx.camera:camera-extensions:1.0.0-alpha27"
}
```

**Fig. 5. Each module that is added to the application build**

### 4.2 *Configuration*

Each use case must be configured to control various aspects of the use case's operations. For example, in the "Image Capture" use case, you can change the aspect ratio and the flash mode.

CameraX automatically configures configuration parameters based on the device on which the application is running. For example, if no resolution has been

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



specified manually previously, or if the resolution defined by the developer is unsupported, CameraX will automatically determine which resolution is best to use. The library provides all these features, removing the developer's need to write custom code for each device.

### 4.3 *Preview* use case

When adding a camera preview to an application, the PreviewView class, a View that can be cut, scaled, or rotated depending on the screen on which it is displayed, should be used.

### 4.4 *Image analysis* use case

The "Image Analysis" use case (case for image analysis) provides the application with a processor image that is used to process the image (e.g., in combination with the ML Kit). The application performs image analysis on each frame that the Preview use case displays.

Images are processed in such a way that they are sent to the executor, where image analysis begins.

The example code in Figures 6 and 7 demonstrates how to implement image analysis as well as how to connect the use cases for image analysis and camera display with the camera life cycle.

Image analysis can be performed in two modes [14]:

1. Mode of blocking
2. Mode of non-blocking

Calling the *setBackgroundStrategy* () method and passing the *STRATEGY_BLOCK_PRODUCER* parameter activates the lock mode. The image processor receives camera frames in sequential order in this mode. This means that if the analyze () method takes longer than the latency of one frame in the current array of frames, the frames may be out of date because new frames are not forwarded for processing until the method returns a return value.

Calling the *setBackgroundStrategy* () method and passing the *STRATEGY_KEEP_ONLY_LATEST* parameter activates the non-blocking mode. The image processor receives the last available frame that the camera recorded when the analyze () method was called in this mode. Some frames may be ignored if the method lasts longer than the latency of a single frame in the current sequence of frames.

It is necessary to call the image before returning a value from the analyze () method.

To avoid memory filling, use the close () method.

*CameraX* produces images in the YUV 420 888 format.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900
Issue 4/2021

### 4.5. *Image capture* use case

The image use case was created to take high quality and high-resolution images. There are two ways to invoke the image capture method:

• *takePicture* (Executor, *OnImageCapturedCallback*) - this method saves the downloaded image to the memory buffer

• *takePicture* (*OutputFileOptions*, Executor, *OnImageSavedCallback*) - this method saves the downloaded image in the forwarded file location

Basic image download functionalities are available for use as part of the case for downloading images. For the best image optimization, you need to set the *ImageCapture.CaptureMode* parameter to *CAPTURE_MODE_MINIMIZE_LATENCY*, while for the best image quality, you need to set it to *CAPTURE_MODE_MAXIMIZE_QUALITY*.

The following code shows an example of how to configure an image download application:

```
ImageCapture imageCapture =
    new ImageCapture.Builder()
        .setTargetRotation(view.getDisplay().getRotation())
        .build();
```

**Fig. 6. Configuring an image download application**

After configuring the camera, the following code generates an image in response to a user action (in this case, a *onClick ()* action):

```
public void onClick() {
    ImageCapture.OutputFileOptions outputFileOptions =
            new ImageCapture.OutputFileOptions.Builder(new File(...)).build();
    imageCapture.takePicture(outputFileOptions, cameraExecutor,
        new ImageCapture.OnImageSavedCallback() {
            @Override
            public void onImageSaved(ImageCapture.OutputFileResults outputFileResults) {
                // insert your code here.
            }
            @Override
            public void onError(ImageCaptureException error) {
                // insert your code here.
            }
        }
    );
}
```

**Fig. 7. action onClick()**

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900

**Issue 4/2021**

**5.** ***Practical Application of the Developed Model***

The designed and implemented model for recognizing objects, text, barcodes and faces with the results of successful reading is shown in Figure 8.

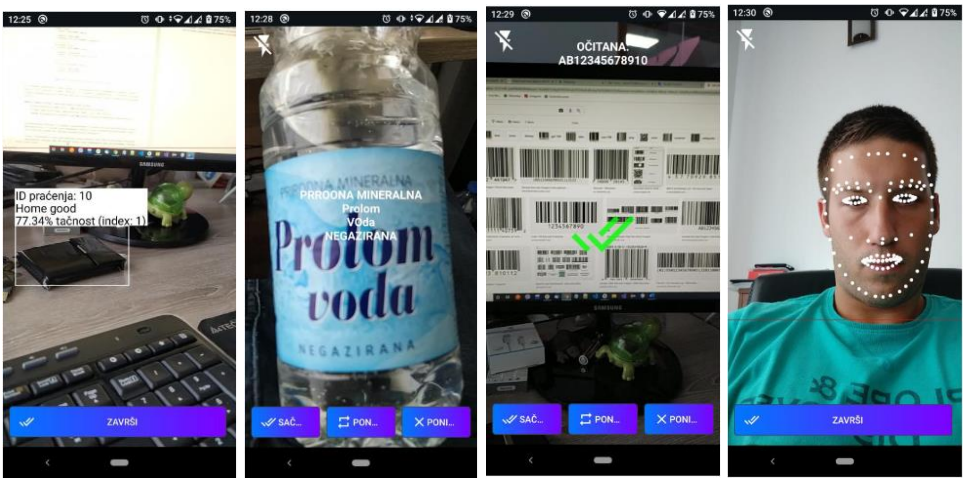


**Fig. 8. Execution of key functionalities of the developed model (from left to right: Object recognition; Text recognition; Barcode recognition; Face recognition)**

Given the complexity of the realized model, the description of the logic of the realized functionalities will be presented with selected elements of importance. The logic of the application is defined by the AndroidManifest.xml file, which serves to describe the basic information related to the application itself, related to the Android operating system and Google Play. It also defines the permissions that the application uses and registers all external and internal services. The manifest file also serves to register all the activities used in the application, and this is shown below.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900
Issue 4/2021

```
<activity android:name=".activities.LauncherActivity"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".activities.IstorijaActivity"
    android:parentActivityName=".activities.MainActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".activities.FaceActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".activities.ObjectActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".activities.TextActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".activities.BarcodeActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".activities.MainActivity"
    android:screenOrientation="portrait"/>
```

LauncherActivity contains a nested tag <intent-filter> in which it is defined that LauncherActivity is the entry point of the application, i.e. when the application is launched, this activity will be the first to be displayed. LauncherActivity has one basic function - starting the animation and, after the successful completion of the animation, starting the MainActivity activity. Animations are defined as special XML files located in the anim resource folder. In the onCreate () method of the activity that is called when the activity is created, the animation itself is started, as well as all the necessary variables are initialized. The following is the starting code of the LauncherActivity java class.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



```
public class LauncherActivity extends AppCompatActivity {

    private Animation fadeOut, fadeIn;
    private LinearLayout root;
    private boolean fade;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_launcher);

        root = findViewById(R.id.root);

        fadeOut = AnimationUtils.loadAnimation( context: this, R.anim.fade_out);
        fadeIn = AnimationUtils.loadAnimation( context: this, R.anim.fade_in);

        root.startAnimation(fadeOut);

        fadeOut.setAnimationListener(fadeListener);
        fadeIn.setAnimationListener(fadeListener);
    }
```

The demanding logic of the realized model is the logic of reading and processing itself frames from the camera. Each activity that works with the camera is defined separately, and each analyzes the image in its own way. The logics of all activities that work with the camera are very similar so that a characteristic part of the logic for scanning bar codes will be shown, as well as the differences found in other activities.

All parameters used for activities are defined as class fields BarcodeActivity. Objects scanner_view, imagine, flash and last_read after initializations in the onCreate () method point to equivalent View elements within the activity_barcode.xml file. Variables soundManager, foundBarcode, capturedBitmap, DateRead, hasFlash and cameraRunning are used for tracking and maintaining the various conditions required during the execution of activities. Finally, the variables camera, previewView, cameraProviderFuture, and cameraProvider serve to work with the camera. The following are the declarations of the basic variables as well as the onCreate method.

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900

**Issue 4/2021**

```
    // camera
    private Camera camera;
    private PreviewView previewView;
    private ListenableFuture<ProcessCameraProvider> cameraProviderFuture;
    private ProcessCameraProvider cameraProvider;

    // view
    private ScannerView scanner_view;
    private ImageView imageDone;
    private ImageView flash;
    private TextView poslednja_ocitana;

    // units
    private SoundManager soundManager;

    private String pronadjenBarkod;
    private Bitmap capturedBitmap;
    private String datumOcitavanja;

    private boolean hasFlash;
    private boolean cameraRunning;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_barcode);

    soundManager = new SoundManager(getApplicationContext());

    previewView = findViewById(R.id.previewView);
    poslednja_ocitana = findViewById(R.id.poslednja_ocitana);
    scanner_view = findViewById(R.id.scanner_view);
    imageDone = findViewById(R.id.image_done);
    Button sacuvaj = findViewById(R.id.sacuvaj);
    Button ponovi = findViewById(R.id.ponovi);
    Button ponisti = findViewById(R.id.ponisti);
    flash = findViewById(R.id.flash);

    sacuvaj.setOnClickListener(onSacuvajClick);
    ponovi.setOnClickListener(onPonoviClick);
    ponisti.setOnClickListener(onPonistiClick);
    flash.setOnClickListener(onFlashClick);

    cameraProviderFuture = ProcessCameraProvider.getInstance( context: this);
    startCamera();
}
```

Annals of Spiru Haret University.
Economic Series
Since 2000
ISSN:2393-1795 ISSN-L:2068-6900



**Conclusion**

This scientific paper addressed both theoretical and practical aspects of the development and use of Android-based applications.

The theoretical section described how the Android software stack functions and how the application layer communicates with the lower layers. It is demonstrated how Android libraries function and how the Android Studio development environment integrates all of this into a single project. The theoretical part of the CameraX library and the ML Kit and its implementation are then explained.

In this paper's practical section, an application was created to demonstrate how the ML Kit works in conjunction with the CameraX library. Scanning bar codes, detecting environmental objects, scanning digital text, scanning faces, creating a database, storing data and images, and so on were all performed. This paper is a functional application incorporating all the above features.

**References**

[1] Darwin I. F., (2017) *Android Cookbook: Problems and Solutions for Android Developers*, Second Edition, O'Reilly Media, Inc.

[2] Griffiths, D. and Griffiths, D. (2017) *Head First Android Development: A Brain-Friendly Guide*, Second Edition, O'Reilly Media, Inc.

[3] Talbot, J. and McLean J. (2014) *Programiranje Android aplikacija*, CET, Beograd.

[4] ITAcademy course - Android application development; PDF documents from the lecture. (visited 22.08.2021.)

[5] https://appinventiv.com/blog/google-play-store-statistics/; The total number of apps in the Google Play Store. (Accessed 23.08.2021.)

[6] https://source.android.com/; About the Android Open-Source project, as well as a complete website for various data. (Accessed 23.08.2021.)

[7] https://developer.android.com/training/camerax; CameraX <br> CameraX architecture, configuration options, preview implementation, image analysis, image capture. (Accessed 25.08.2021.)

[8] https://developers.google.com/ml-kit/guides; ML kit, text recognition, face recognition, pose Detection, selfie segmentation, bar code scanning, image tagging, object detection and tracking, digital ink recognition. (Accessed August 27, 2021.)

[9] https://developers.google.com/; Using Android documentation while creating a complete application. . (Accessed August 27, 2021.)