Universitatea Politehnică Timișoara
Facultatea de Automatică și Calculatoare, specializarea Informatică, anul II

PROIECT SINCRETIC

SCRABBLE COMPUTER VISION

proiect susținut de către:
Balea Andrei-Petru și Micloș Eduard-Pavel

prof. coordonator: Bocan Valer

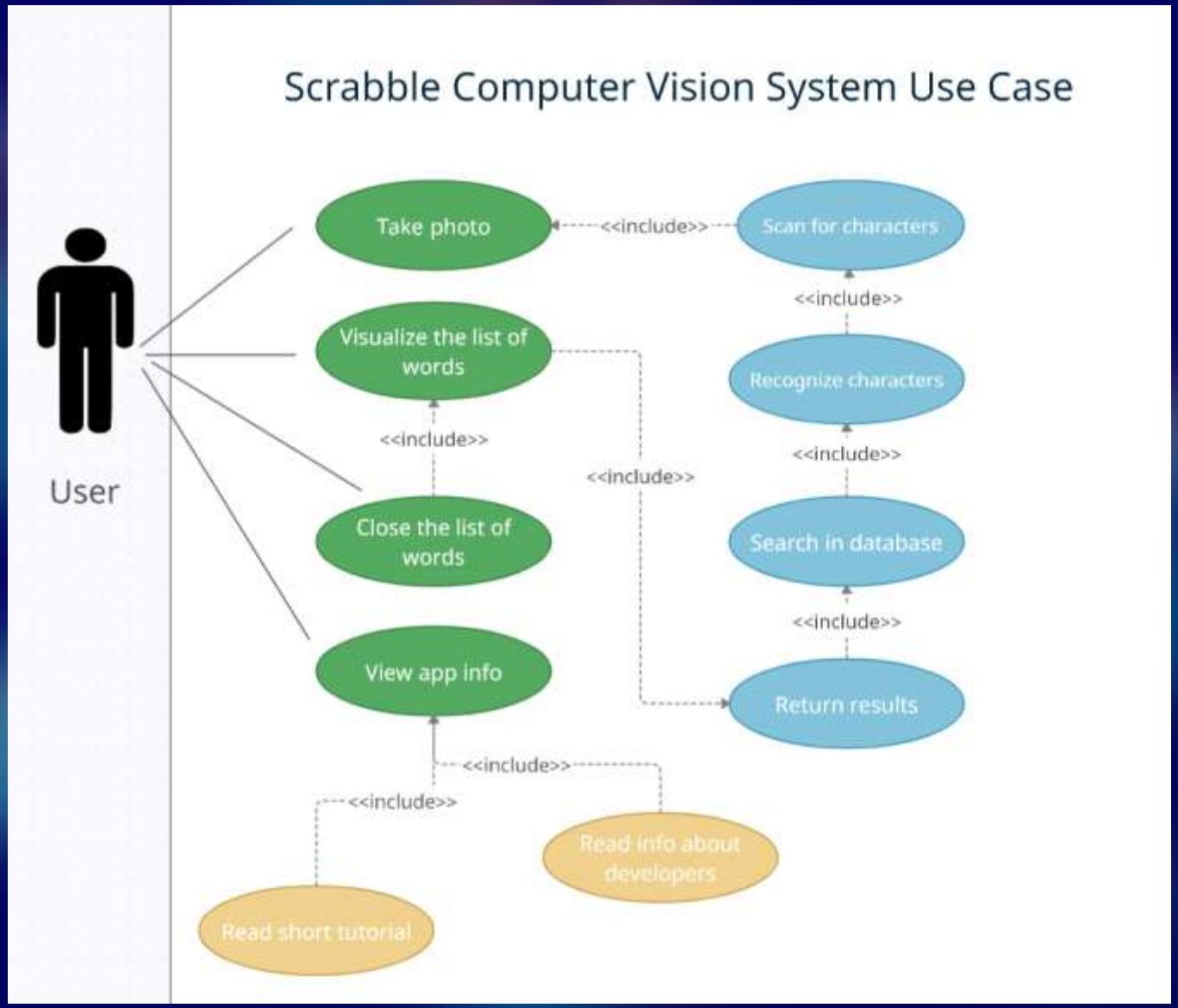
Universitatea Politehnica Timișoara

VIZIUNEA NOASTRĂ

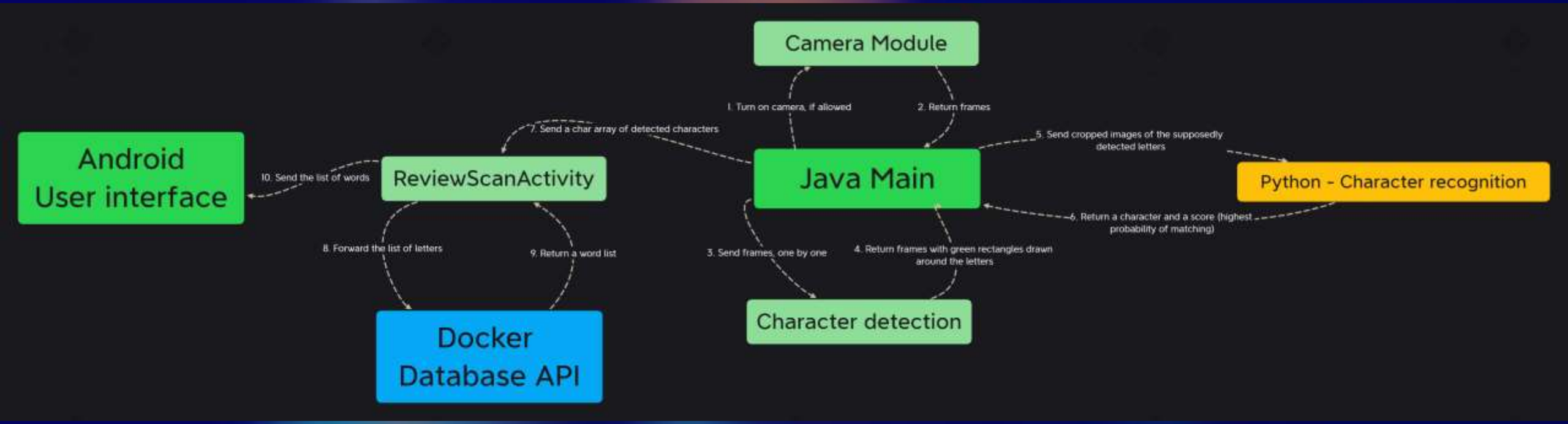
Aplicația **SCRABBLE COMPUTER VISION** a fost construită cu scopul de a ne determina să învățăm lucruri noi din domeniile care ne pasionează: **mobile app development**, **computer vision**, **algoritmică**, **baze de date în cloud**.

Aplicația noastră detectează literele în contextul unui joc de **Scrabble**, prin scanare folosind camera frontală, și returnează utilizatorului **o listă cu toate cuvintele care se pot forma** folosind acele litere.

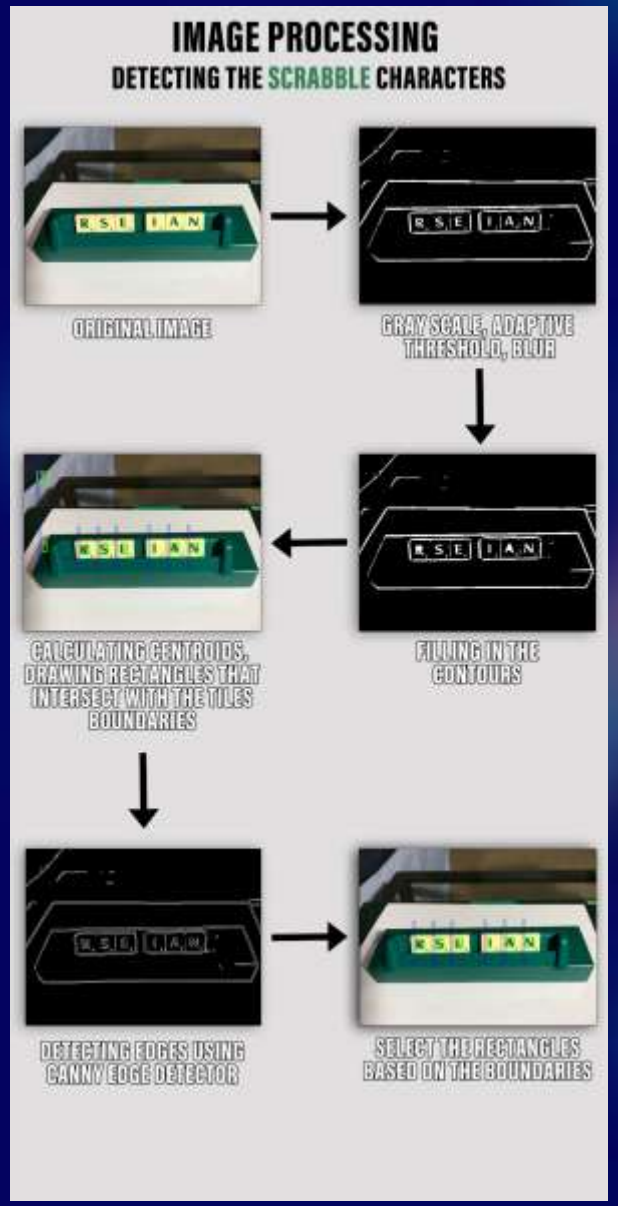
DIAGRAMA USE-CASE

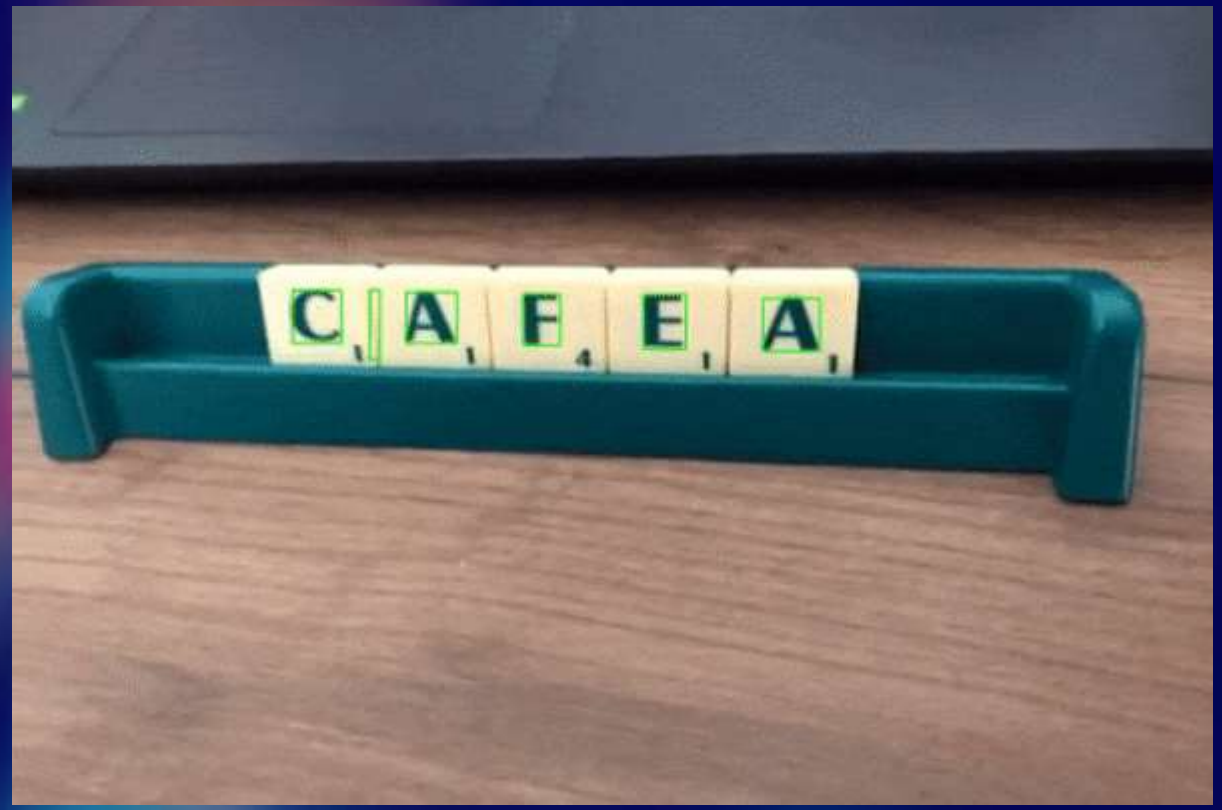


WORKFLOW


Camera Module
1. Turn on camera, if allowed
2. Return frames
7. Send a char array of detected characters
5. Send cropped images of the supposedly detected letters
Android User interface
10. Send the list of words
ReviewScanActivity
Java Main
Python - Character recognition
6. Return a character and a score (highest probability of matching)
8. Forward the list of letters
9. Return a word list
3. Send frames, one by one
4. Return frames with green rectangles drawn around the letters
Character detection
Docker Database API

CUM FUNCȚIONEAZĂ DETECTAREA CARACTERLELOR





*PENTRU CHARACTER RECOGNITION, AM FOLOSIT ALGORITMUL SSI (STRUCTURAL SIMILARITY INDEX)

FRÂNTURI DE COD - IMAGE TRANSFORMATION

```java
package com.andyedy.scrabble_computer_vision.Util;

import androidx.annotation.NonNull;

import org.opencv.android.CameraBridgeViewBase;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.Moments;

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

public class ImageTransform implements IdealConstants {

    private static Mat frame;

    private static final Scalar boundingRectangleColor = new Scalar(0, 255, 0);
    private static final int boundingRectangleThickness = 2;

    @NonNull
    private static Mat applyAdaptiveThreshold(Mat grayImg){
        Mat threshImg = new Mat();
        Imgproc.adaptiveThreshold(grayImg, threshImg, 255, Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY_INV, blockSize, constantC);
        return threshImg;
    }

    @NonNull
    private static List<MatOfPoint> getContours(Mat threshImage){
        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
```

FRÂNTURI DE COD - IMAGE TRANSFORMATION

```
        /* This variable is not used. */
        Mat hierarchy = new Mat();

        Imgproc.findContours(threshImage, contours, hierarchy, Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_NONE);

        return contours;
    }


    public static void drawRectangle(Mat frame, @NonNull Rect boundingRect){
        /* Top left corner of the rectangle. */
        Point startPoint = new Point();

        /* Bottom right corner of the rectangle. */
        Point finishPoint = new Point();

        startPoint.x = boundingRect.x;
        startPoint.y = boundingRect.y;

        finishPoint.x = startPoint.x + boundingRect.width;
        finishPoint.y = startPoint.y + boundingRect.height;

        Imgproc.rectangle(frame, startPoint, finishPoint, boundingRectangleColor, boundingRectangleThickness);
    }

    public static Vector<Rect> transform (@NonNull CameraBridgeViewBase.CvCameraViewFrame inputFrame){
        /* Creating a matrix from current input frame. */
        frame = inputFrame.rgba();

        /* Getting image dimensions. */
        int imageHeight = frame.rows();
        int imageWidth = frame.cols();

        /* Getting gray-scale image such that we can apply the thresholding effects. */
        Mat grayFrame = inputFrame.gray();
```

FRÂNTURI DE COD - IMAGE TRANSFORMATION

```
71
72          /* Threshold image. */
73          Mat threshImage = applyAdaptiveThreshold(grayFrame);;
74
75          /* Getting the contours. */
76          List<MatOfPoint> contours = getContours(threshImage);
77
78          /* Vector of drawn rectangles to be returned by the function. */
79          Vector<Rect> drawnRectangles = new Vector<Rect>();
80
81          /* Iterating through all the contours. */
82          for(MatOfPoint contour : contours) {
83              Rect boundingRect = Imgproc.boundingRect(contour);
84
85              /* Checking for all the conditions that need to be met.
86              Take note that these conditions are the ones that determine:
87
88              - how far/close the camera should be.
89              - how likely it is to find and draw a rectangle on the screen.
90               */
91              if (boundingRect.height > rectangleMinHeight &&
92                      boundingRect.height < rectangleMaxHeight &&
93                      boundingRect.width > rectangleMinWidth &&
94                      boundingRect.width < rectangleMaxWidth) {
95
96                  Moments moments = Imgproc.moments(contour);
97
98                  double blockArea = moments.m00;
99                  double xMassDistribution = moments.m10;
100                 double yMassDistribution = moments.m01;
101
102                 if (blockArea != 0) {
103
104                     /* Determining the center of mass. */
105                     int centroidX = (int) (xMassDistribution / blockArea);
106                     int centroidY = (int) (yMassDistribution / blockArea);
```

FRÂNTURI DE COD - LETTER

```java
package com.andyedy.scrabble_computer_vision.Util;

import android.os.Parcel;
import android.os.Parcelable;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Locale;

public class Letter implements Comparable, Parcelable {
    private String value;
    private int frequency;

    public static ArrayList<Letter> getArrayFromString(String MyLetters) {
        ArrayList<Letter> letterList = new ArrayList<>();
        for (char c: MyLetters.toCharArray()) {
            boolean exists = false;
            for(Letter l: letterList) {
                //if (l.getValue().compareTo(String.format("%c", c)) == 0) {
                if(l.getValue().toUpperCase(Locale.ROOT).toCharArray()[0] == Character.toUpperCase(c)) {
                    l.frequency++;
                    exists = true;
                }
                if(exists) break;
            }
            if(!exists) letterList.add(new Letter(String.format("%c", c), 1));
        }

        Collections.sort(letterList);
```

FRÂNTURI DE COD - LETTER

```
31
32          return letterList;
33      }
34
35      public Letter(String value, int frequency) {
36          this.value = value.toUpperCase(Locale.ROOT);
37          this.frequency = frequency;
38      }
39
40      protected Letter(Parcel in) {
41          value = in.readString();
42          frequency = in.readInt();
43      }
44
45      public static final Creator<Letter> CREATOR = new Creator<Letter>() {
46          @Override
47          public Letter createFromParcel(Parcel in) {
48              return new Letter(in);
49          }
50
51          @Override
52          public Letter[] newArray(int size) {
53              return new Letter[size];
54          }
55      };
56
57      public String getValue() {
58          return value;
59      }
60
61      public int getFrequency() {
62          return frequency;
63      }
64
```

FRÂNTURI DE COD - LETTER

```
63      }
64
65      @Override
66      public int compareTo(Object o) {
67          Letter toLetter = (Letter)o;
68          return value.compareTo(toLetter.getValue());
69      }
70
71      @Override
72      public int describeContents() {
73          return 0;
74      }
75
76      @Override
77      public void writeToParcel(Parcel parcel, int i) {
78          parcel.writeString(value);
79          parcel.writeInt(frequency);
80      }
81  }
```