Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare și
Tehnologia Informației

DESIGNYOURROOM - APLICAȚIE MOBILĂ PENTRU SCHIMBAREA ASPECTULUI CAMEREI

Student:

Bakaity Lorena

Timișoara
Decembrie, 2020

Cuprins

Capitolul 1

Introducere

1.1 Context

Plecand de la ideea ca "trebuie sa existe o modalitate mai simpla sa pot sa schimb culoarea la camera decat sa fiu nevoita sa testez in Sims" si de la faptul ca sunt la CTI am zis ca trebuie sa fie o metoda mai eficienta si relativ mai simpla de a testa acest lucru.

1.2 Descrierea proiectului

Scopul principal al proiectului este de a rezolva o problema destul de intalnita si anume "ce culoare sa imi vopsesc camera" si de a evita un posibil esec. Trecand recent printr-o astfel de dilema si fiind sub presiunea "trebuie sa iti alegi o tema de proiect" am venit cu aceasta idee. Din perspectiva utilizatorului proiectul arata astfel: daca el isi doreste sa schimbe culoarea camerei face poza la camera sau incarca o poza, aleage dintr-o gama de culori, tapeturi, texturi si ca prin magie camera va avea alta culoare. Din perspectiva mea toata magia consta din doi mari pasi si anume: segmentare si "aplicarea" culorii. Poate ca aplicatia pare simpla, dar dupa cum Richard Branson afirma

> *“Nici o idee nu este prea mica, orice fel de idee are potentialul de a schimba lumea.”*

astfel eu am inceput de la o idee mai micuta care pe parcurs poate fi dezvoltata intr-o aplicatie mai complexa.

Capitolul 2

Analiza domeniului

2.1 Aplicații mobile asemănătoare

Dupa cum un profesor de la facultate spunea

"Daca ai venit cu o idee de proiect inseamna ca a fost deja facut."

am gasit tot felul de aplicatii pe Google Play si voi specifica doua aplicatii: Paint Tester si Paint My Room. Ambele aplicatii au ca scop principal schimbarea culorii peretelui doar ca Paint Tester nu dispune de wall detection astfel utilizatorul are la alegere o gama larga de brush tool-uri pentru a schimba culoarea, iar Paint My Room a introdus wall detection realtime cu o gama larga de culori. Ce as dori eu sa adaug aplicatiei mele este de a putea aplica nu doar culori simple ci si tapeturi.

Table 2.1: Analiză comparativă între aplicatii regasite in MagazinPlay. Funcționalitățile analizate regăsite într-o aplicație sunt marcate cu ✓, iar cele absente cu ✗

Caracteristici și funcționalități	**Paint Tester**	**Paint My Room**	**DesignYourRoom**
Sistem de operare	Android	Android	Android
Magazin de aplicații	Google Play	Google Play	✗
Nota din Google Play	3.00/5	2,6/5	✗
Număr de instalări	1,000,000+	100,000+	✗
Număr de ratinguri	2,276	290	✗
Colorarea peretilor	✓	✓	✓
Detectarea peretilor	✗	✓	✓
Gama larga de culor	✓	✓	✓
Aplicarea instanta a culorii	✗	✗	✓
Aplicare Real-time a culorii	✗	✓	✗
Object detection	✗	✗	✓

Capitolul 3

Proiectare

3.0.1 Diagra de Use-case-uri

In acest Capitol pe scurt am explicat functionalitatile aplicatiei si anume, utilizatorul, dupa ce a pornit aplicatia, poate sa aleaga doua modalitati de incarcare a unei poze si anume sa faca o poza sau sa incarce din galerie. Cand poza a fost incarcata utilizatorul o poate modifica alegand o culoare sau poate sa obtina o lista cu obiectele regasite in camera. De asemenea poate sa partajeze poza.

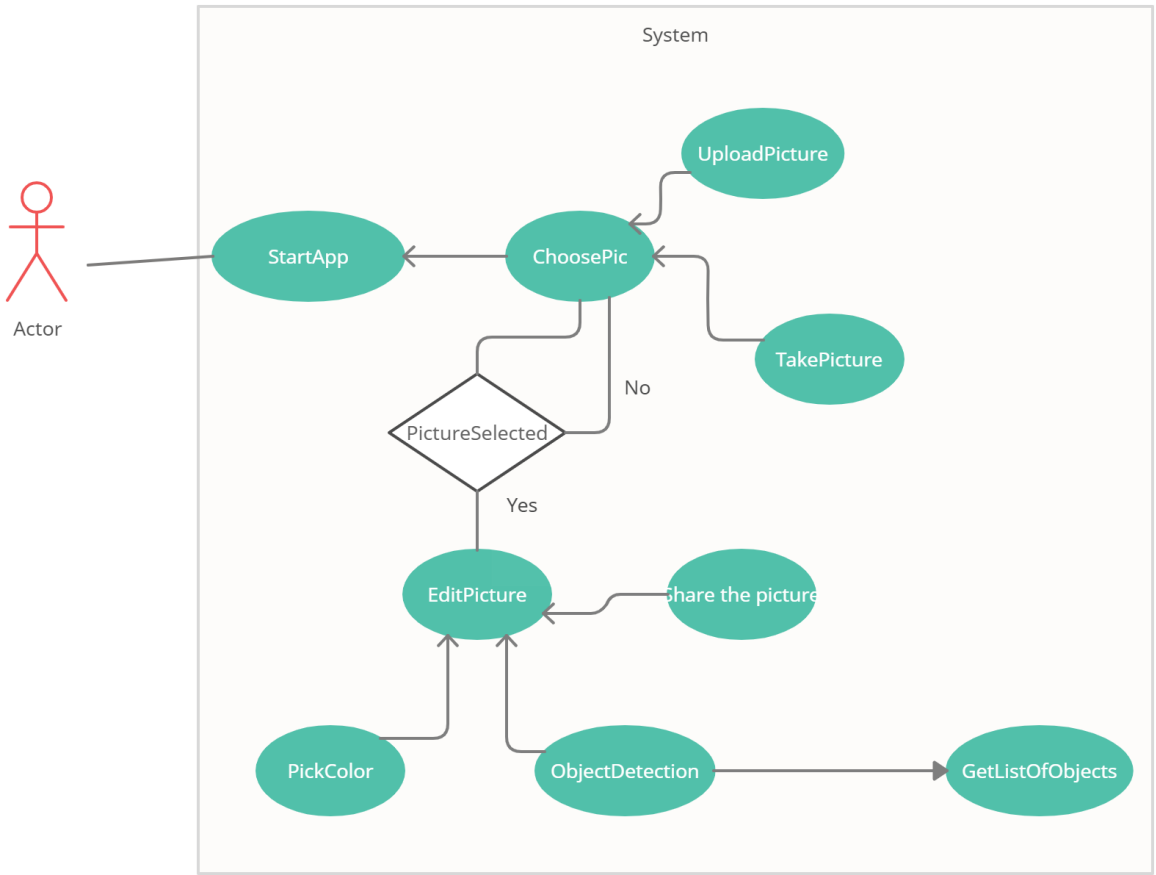

Figure 3.1: Diagrama de UseCase-uri.

Capitolul 4

Utilizare aplicatie

4.1 Structura

Pentru aplicatia mea am ales un design cat mai simplu, astfel ea este formata din 4 ecrane. Primul ecran este doar un ecran de start, introductiv, in care nu exista nicio functionalitate.

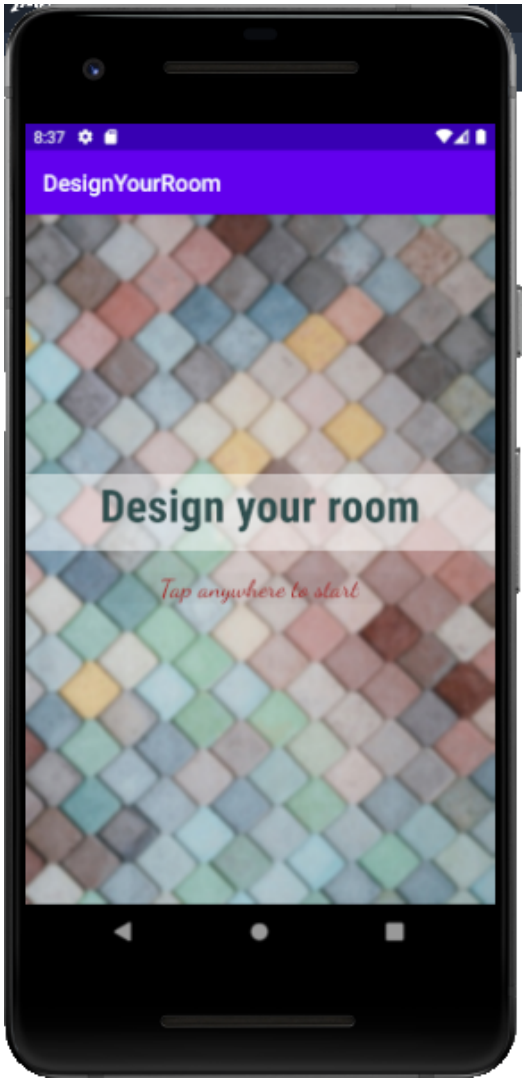

Figure 4.1: Ecranul de start.

In al doilea ecran utilizatorul poate sa aleaga metoda de incarcare a unei poze si anume sa faca o poza sau sa incarce din galerie o poza.

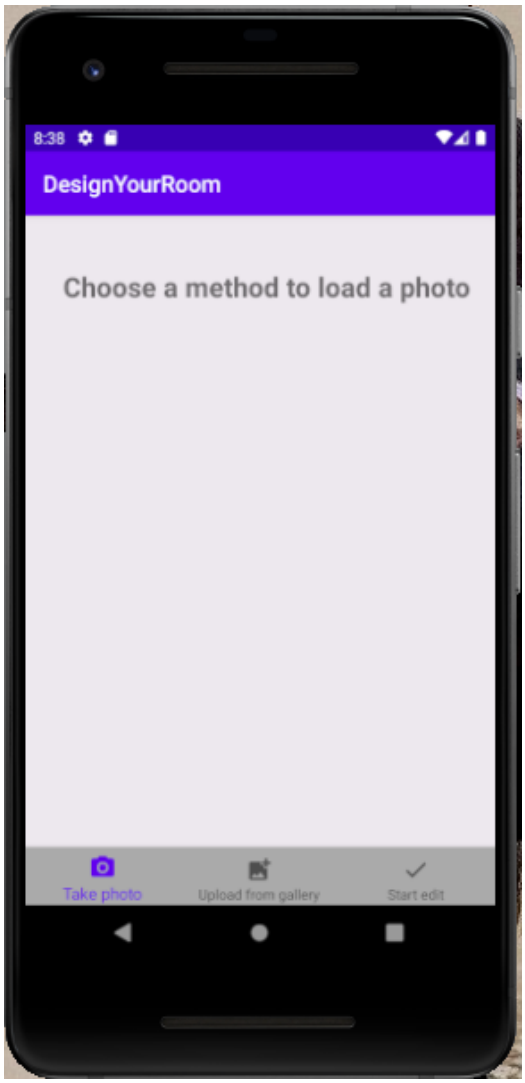

Figure 4.2: Al doilea ecran.

Dupa ce utilizatorul alege o metoda de incarcare, poza este afisata pe ecran, iar daca utilizatorul este multumit trece la al treilea ecran.

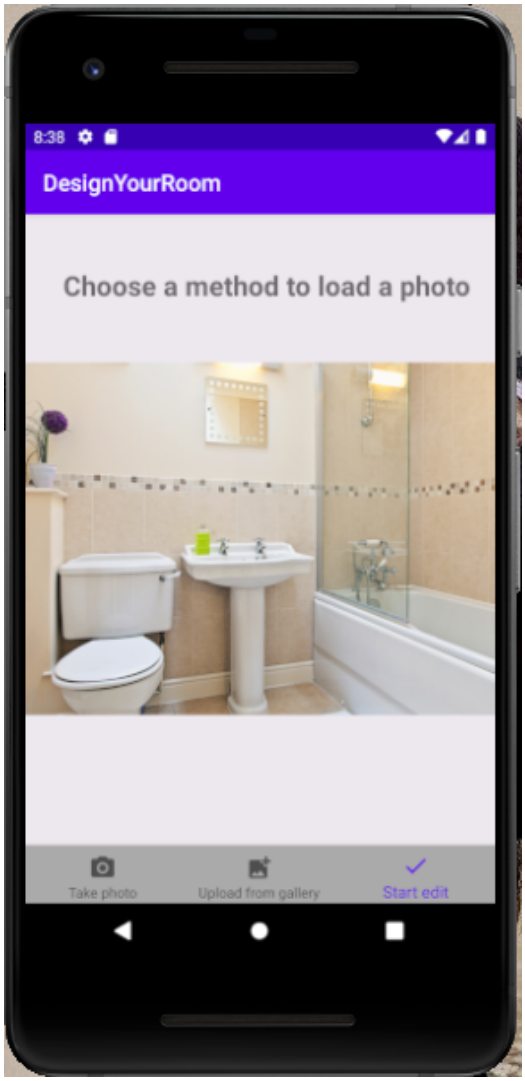

Figure 4.3: Poza incarcata.

Al treilea ecran este format din 2 butoane(doua functionalitati) si anume, poate sa schimbe culoarea din camera sau poate sa afiseze o lista cu obiectele din camera, lista care contine obiectul si link-ul la o categorie de obiecte.

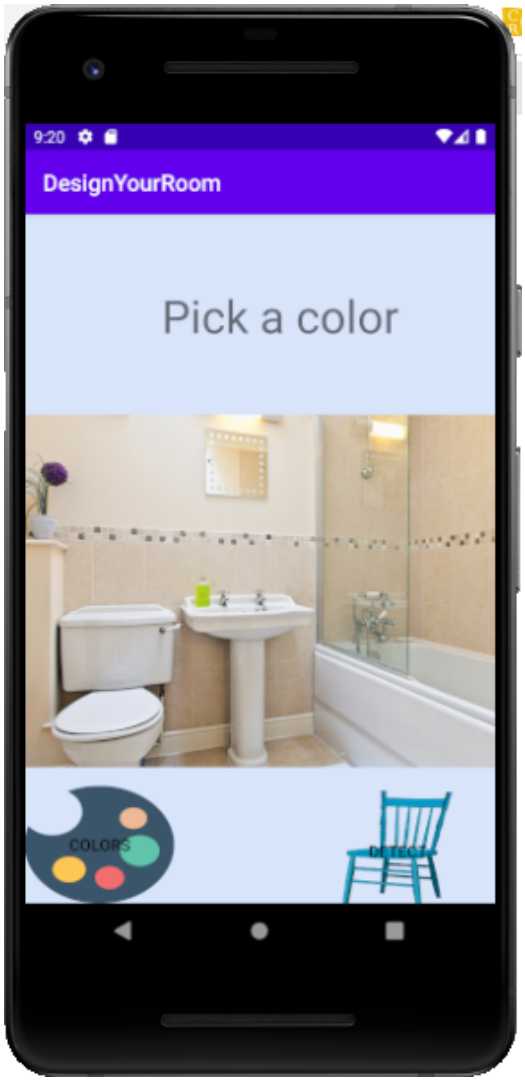

Figure 4.4: Ecranul de editare.

Utilizatorul poate sa aleaga dintr-o gama variata de culori

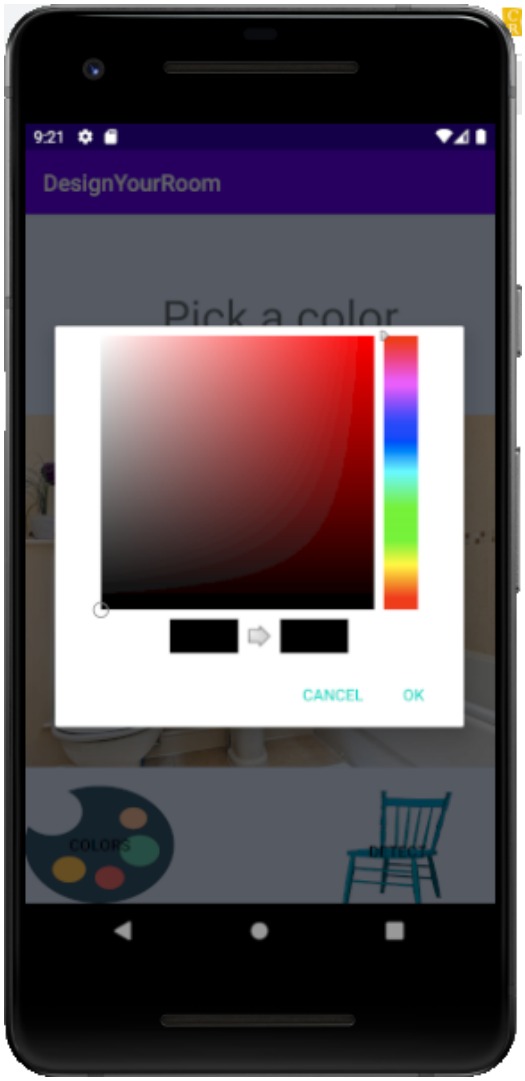

Figure 4.5: Paleta de culori.

A doua functionalitate presupune detectarea de obiecte din poza si afisarea unei liste cu obiectele detectate.

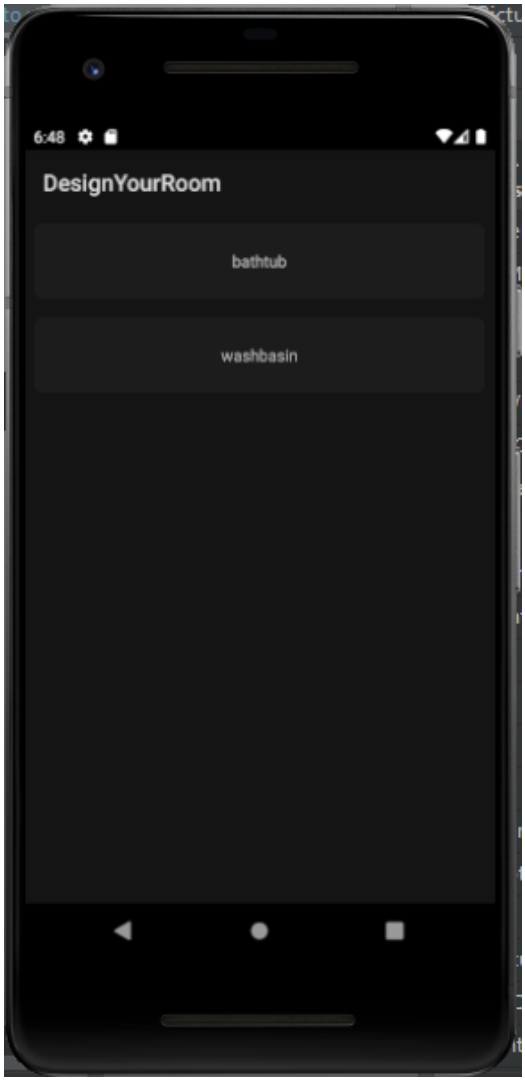

Figure 4.6: Lista obiecte.

Si de asemenea fiecare obiect gasit prin selecarea lui se va deschide o pagina cu obiecte din aceeasi gama.

Figure 4.7: Obiecte din aceeasi gama.

Capitolul 5

Tehnologii utilizate

5.1 Android

Android este un sistem de operare, bazat pe Linux, conceput initial pentru dispozitivele cu touch screen precum smarthphone-uri, tablete etc. Prin intermediul acestui sistem de operare dezvoltatorii pot sa construiasca aplicatii unice, adica aplicatia sa poate rula pe orice timp de dispozitiv care are ca sistem de operare android[1]

5.2 Android Studio

Android Studio este un IDE(integrated development environment) folosit in mod special pentru dezvoltarea aplicatiilor Android

5.3 Kotlin

De ce am ales Kotlin? In primul rand mi s-a impus sa dezvolt aplicatia in acest limbaj, si in al doilea rand am zis ca e o provocare si de asemenea un avantaj pentru mine pentru ca voi invata un limbaj nou. La prima privire Kotlin arata ca o versiune mai concisa si simplificata a Java.

5.4 Versionarea codului

5.4.1 Git

Git este cea mai utilizata metoda pentru versionarea codului. Este folosit de catre dezvoltatori pentru a incarca intr-un repository central de fiecare data atunci cand fac o anumita modificare la o aplicatie, proiect etc, pentru a urmari modificarile facute sau pentru a colabora cu alti dezvoltatori mai usor. [2]

5.4.2 Github

Eu am optat pentru GitHub pentru ca este usor de folosit, este o optiune buna pentru stocarea aplocatiei. Linkul catre repository-ul aplicatiei[3]

5.5 Baza de date

Pentru Baza de date am folosit Firebase care este o aplicatie atat mobile cat si web oferind utilizatorilor o multime de servicii si instrumente care ii ajuta in dezvoltarea unor aplicatii de inalta calitate. In 2011 Firebase, numit pe atunci Envolve, oferea utilizatorilor un API care permitea aestora integrarea unui chat online in site-ul web creat de ei. In 2014 Envolve a ajuns in posesia lui Google si a devenit Firebase-ul de azi cu o multime de functionalitati utile pentr dezvoltarea unei aplicatie[4]

5.5.1 Realtime Database

Am folosit Realtime Database pentru a stoca obiecte avand ca si cheie denumirea "obiectului", iar valoarea este un link catre un anumit site de exemplu :Dedeman,Ikea.

5.6 Tehnologii folosite pentru segmentare

5.6.1 OpenCV

Pentru segmentare am folosit cea mai cunoscuta librarie pentru Computer vision si anume OpenCv. Libraria are peste 2500 de algoritmi optimizati precum: detectarea si recunoasterea faciala, identificarea obiectelor etc.[5]

Capitolul 6

Implementare

6.1 Algoritmi

Dupa cum am precizat si in introducere algoritmul consta in doua mari etape: edge detection si colorare. In aplicatie am folosit o librarie destul de utilizata si anume OpenCv care mi-a oferit toate functionalitatile de care aveam nevoie[6].

6.1.1 Segmentare

Inainte ca sa aplic algoritmul CannyEdge a trebuit sa preprocesez inputul astfel primul pas in cadrul segmentarii a fost sa convertesc imaginea bitmap in MAT pentru ca este cel mai usor si eficient mod de a lucra cu imaginile, apoi al doilea pas a fost sa transform imaginea din RBGA in RGB iar in final am obtinut imaginea grayscale pentru a aplica Canny Edge.

```
val mRgbMat = Mat()
Utils.bitmapToMat(bitmap, mRgbMat)
Imgproc.cvtColor(mRgbMat,mRgbMat, Imgproc.COLOR_RGBA2RGB)
val img = Mat()
mRgbMat.copyTo(img)
val mGreyScaleMat = Mat()
Imgproc.cvtColor(mRgbMat, mGreyScaleMat, Imgproc.COLOR_RGB2GRAY, 3)
Imgproc.medianBlur(mGreyScaleMat,mGreyScaleMat,3)
```

Listing 6.1: Preprocesarea imaginii

De asemenea pentru a diferentia cat mai bine culorile, separarea acestora de intensitate, am tranfsormat imaginea din RBG in HSV, iar valorile au fost stocare intr-un array

```
 val hsvImage = Mat()
Imgproc.cvtColor(img,hsvImage, Imgproc.COLOR_RGB2HSV)
val list = ArrayList<Mat>(3)
Core.split(hsvImage, list)
val sChannelMat = Mat()
Core.merge(listOf(list[1]), sChannelMat)
Imgproc.medianBlur(sChannelMat,sChannelMat,3)
```

Listing 6.2: HSV Image

Apoi am aplicat functia CannyEdge

```
val cannyGreyMat = Mat()
Imgproc.Canny(mGreyScaleMat, cannyGreyMat, cannyMinTreshold, 90.0, 3)
```

Listing 6.3: Canny Edge

Pentru matricea obtinuta am aplicat din nou CannyEdge, iar rezultatul obtinut este folosit in cadrul functiei floodFill

```
val cannyMat = Mat()
Imgproc.Canny(sChannelMat, cannyMat, cannyMinTreshold,90.0, 3)
Core.addWeighted(cannyMat,0.6, cannyGreyMat,0.4 ,0.0,cannyMat)
Imgproc.dilate(cannyMat, cannyMat,mask, Point(0.0,0.0), 5)
```

Listing 6.4: Canny Edge

A doua etapa este flood filling care are scopul de a umple o masca binara formand o componenta conectata pornind de la un seed point. Iar a treia si ultima operatia este dilatarea

```
val mask = Mat(Size(mRgbMat.width().toDouble(), mRgbMat.height().toDouble()), CvType.CV_8UC1,
    Scalar(0.0))
val floodFillFlag = 8
Imgproc.floodFill(
        mRgbMat,
        cannyMat,
        seedPoint,
        Scalar(
                Color.red(chosenColor).toDouble(),
                Color.green(chosenColor).toDouble(),
                Color.blue(chosenColor).toDouble()),
        Rect(),
        Scalar(5.0, 5.0, 5.0),
        Scalar(5.0, 5.0, 5.0),
        floodFillFlag
        )

Imgproc.dilate(mRgbMat, mRgbMat, mask, Point(0.0,0.0), 5)
```

Listing 6.5: FloodFill

6.1.2 Alpha Blending

Alpha blending este etapa finala si presupune combinatia liniara intre imaginea originala si masca colorata in regiunea mastii si reproducerea imaginii originale in afara mastii

```
val rgbHsvImage = Mat()
        Imgproc.cvtColor(mRgbMat,rgbHsvImage, Imgproc.COLOR_RGB2HSV)
        val list1 = ArrayList<Mat>(3)
        Core.split(rgbHsvImage, list1)

        val result = Mat()
        Core.merge(listOf(list1[0], list1[1], list[2]), result)

        Imgproc.cvtColor(result, result, Imgproc.COLOR_HSV2RGB)
        Core.addWeighted(result,0.6, img,0.4 ,0.0,result )
```

Listing 6.6: FloodFill

Capitolul 7

Concluzii

O concluzie la care am ajuns este faptul ca nu am reusit sa pastrez ideea pe care am avut-o la inceput ci pe parcurs am schimbat-o in proportie de 50%. Am renuntat la diferite functionalitati precum aplicarea culorii real-time, sau realitatii augmentate din cauza ca nu am avut pe ce sa testez. Iar tot ce am facut pana acuma am facut din emulator.

Ce as mai imbunatatii la aplicatie, una dintre imbunatatiri ar fi setul de date sa fie mai divers, modelul pe care l-am utilizat cuprindea un numar mic de obiecte de casa, astfel as crea un model doar cu aceste obiecte. De asemenea algorimtul de edge detection sa aiba o acuratete mai mare si de asemenea as schimba design-ul.

Ar mai fi cateva functionalitati pe care le-as adauga, functionalitati pe care le-am aminiti mai sus precum : colorarea peretelui real-time.

Bibliografie

[1] *Despre Android*. https://www.elprocus.com/what-is-android-introduction-features-applications/.

[2] Limbaj de programare kotlin. https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html.

[3] Bakaity L. *Depozitul codului sursă al aplicației* ***DesignYourRoom***. https://github.com/llorenadenisa/DesignYourRoom-app, 2020. [Online; Accesat: 07.12.2020].

[4] Firebase. https://hackernoon.com/introduction-to-firebase-218a23186cd7.

[5] Despre opencv. https://opencv.org/about/.

[6] *Real-Time Segmentation, Tracking, and Coloring of Walls Using iOS* . https://stacks.stanford.edu/file/druid:yj296hj2790/Yeung_Piersol_Liu_Project_Imaggle.pdf/, 2012. [Online; Accesat: 07.12.2020].