

**Facultatea de Automatica și Calculatoare**

**Program de analiză a trăsăturilor morfologice ale feței**

Student: Mocoi Ioan-Victor

Grupa: 30234

1. **Introducere**

**Context:**

Recunoașterea facială devine tot mai relevantă în domeniul securității, automatizării și interacțiunii om-mașină, în această eră a tehnologiei avansate. Proiectul se concentrează pe dezvoltarea unei aplicații de recunoaștere facială în limbajul de programare Java, care poate identifica ochii, nasul și gura unei persoane în imagini sau în timp real. Această tehnologie are potențialul de a revoluționa domenii precum securitatea, accesul la dispozitive și asistența medicală.

**Motivație:**

Motivația pentru acest proiect este contribuirea la îmbunătățirea eficienței și securității prin intermediul recunoașterii faciale. Cu creșterea continuă a infracțiunilor și a nevoii de securitate, precum și cu cerințele în creștere de a simplifica autentificarea și autorizarea în diferite domenii, recunoașterea facială devine o soluție promițătoare. Prin acest proiect, se vor explora tehnologiile de ultimă oră pentru a dezvolta un sistem de recunoaștere facială robust și precis.

**Obiective:**

Obiectivele principale ale acestui proiect sunt:

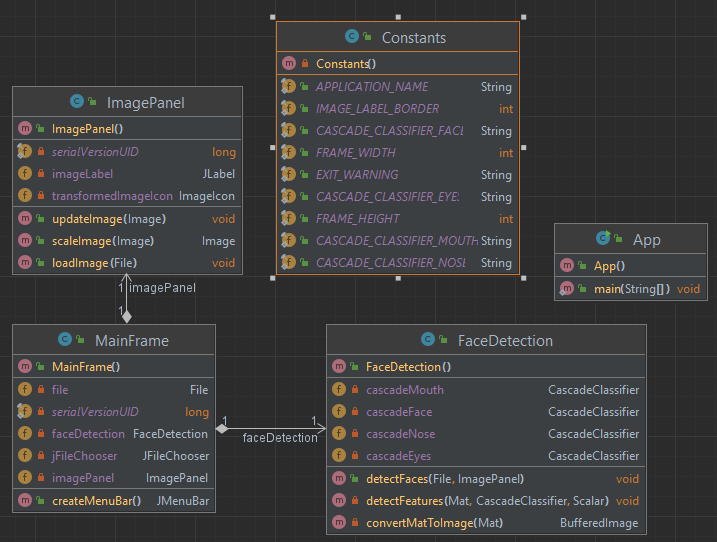
* dezvoltarea unei aplicații de recunoaștere facială în Java care să poată identifica și analiza morfologia trăsăturilor faciale;
* realizarea unei interfețe prietenoase pentru a permite utilizatorilor să folosească și să testeze aplicația cu ușurință;
* optimizarea metodei de recunoaștere facială pentru a asigura precizie și performanță;

Prin aceste obiective, se propune contribuirea la progresul tehnologic și aducerea de beneficii semnificative în domenii precum securitatea, accesul la dispozitive și asistența medicală, într-un mod care respectă principiile etice și legale ale confidențialității datelor personale.

1. **Studiu bibliografic**

Recunoașterea facială a devenit o componentă esențială a tehnologiei moderne și a fost subiectul unor cercetări extinse în ultimii ani. Acest studiu bibliografic are ca scop să ofere o privire de ansamblu asupra resurselor și cercetărilor relevante care vor fi de ajutor în dezvoltarea proiectului nostru de recunoaștere a trăsăturilor faciale.

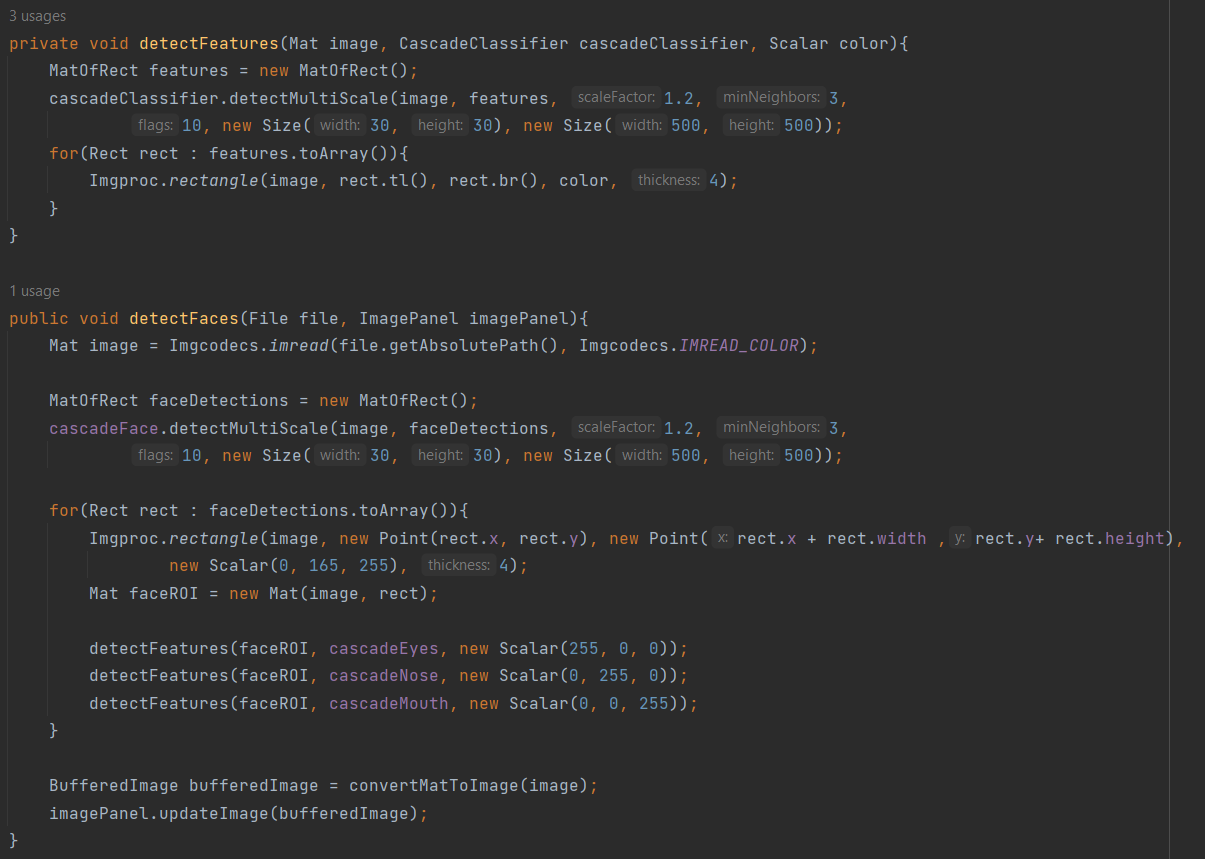
1. **"FaceNet: A Unified Embedding for Face Recognition and Clustering" (2015)** - Acest articol scris de Florian Schroff, Dmitry Kalenichenko și James Philbin prezintă o abordare revoluționară de recunoaștere facială bazată pe o rețea neuronală convoluțională.
2. **"Facial Landmark Detection by Deep Multi-task Learning" (2016)** - Autorii Zhanpeng Zhang, Ping Luo, Chen Change Loy și Xiaoou Tang explorează o tehnică de detectare a punctelor cheie ale feței, inclusiv ochii, nasul și gura, cu ajutorul rețelelor neuronale profunde.
3. **"OpenCV: A general computer vision library" (2000)** - Biblioteca OpenCV ne oferă un set extins de funcționalități pentru procesarea imaginilor și recunoașterea facială. Vom utiliza această resursă pentru a facilita procesul nostru de dezvoltare în Java.
4. **Design și analiză**

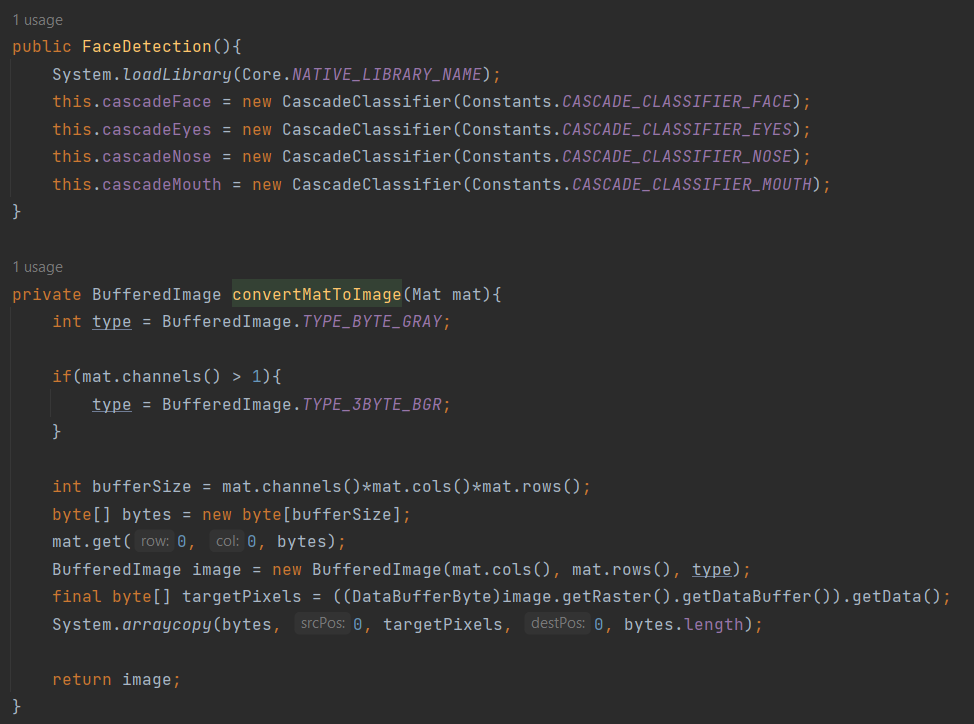
****

**diagrama UML a claselor**

* **Clasa App:**
  + reprezintă clasa cu funcția **main** și apelul de **run** pentru pornirea aplicației
* **Clasa Constants:**
  + un Singleton ce conține constante importante folosite in multiple locuri in proiect
* **Clasa ImagePanel**
  + cuprinde funcțiile pentru încărcarea unei imagini folosind un File Path, scalarea imaginii pentru a putea fi pusă pe panou, și update-ul panoului cu noua imagine procesată
* **Clasa FaceDetection**
  + cele 4 atribute ale clasei reprezintă cascader-ele ce folosesc algoritmii haar cascade pentru identificarea feței, ochilor, nasului și gurii
  + funcția de **detectFeatures** folosește cascader-ele pentru identificarea trăsăturilor feței, iar funcția **detectFaces** apelează **detectFeatures** și unește rezultatele algoritmilor pentru a determina fețele și trăsăturile
  + funcția **convertMatToImage** convertește matricea generată de algoritm într-o imagine ce poate fi încărcată pe panou
* **Clasa MainFrame**
  + face legătura între clasele **FaceDetection** și **ImagePanel**
  + creează meniul aplicației și atribuie funcționalități elementelor din meniu

1. **Implementare**





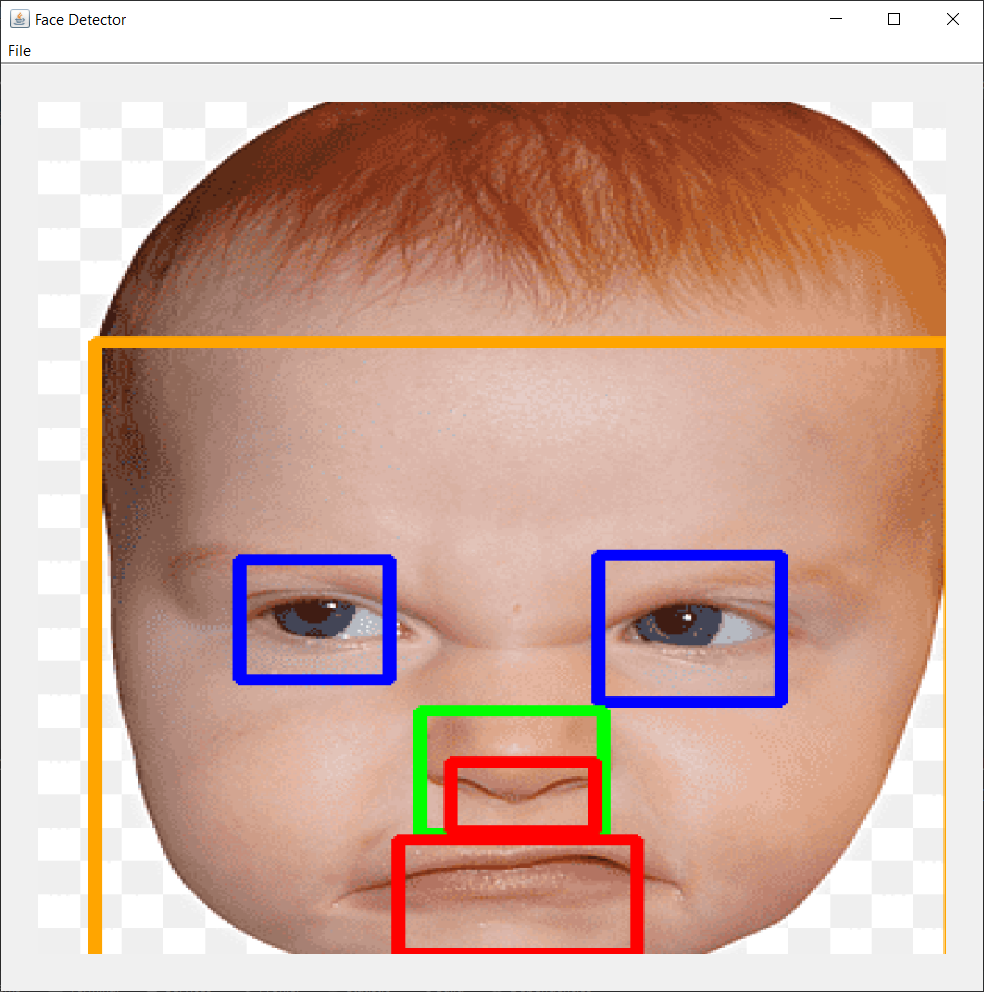
**clasa FaceDetection, ce conține implementarea algoritmului de detectare a fețe**

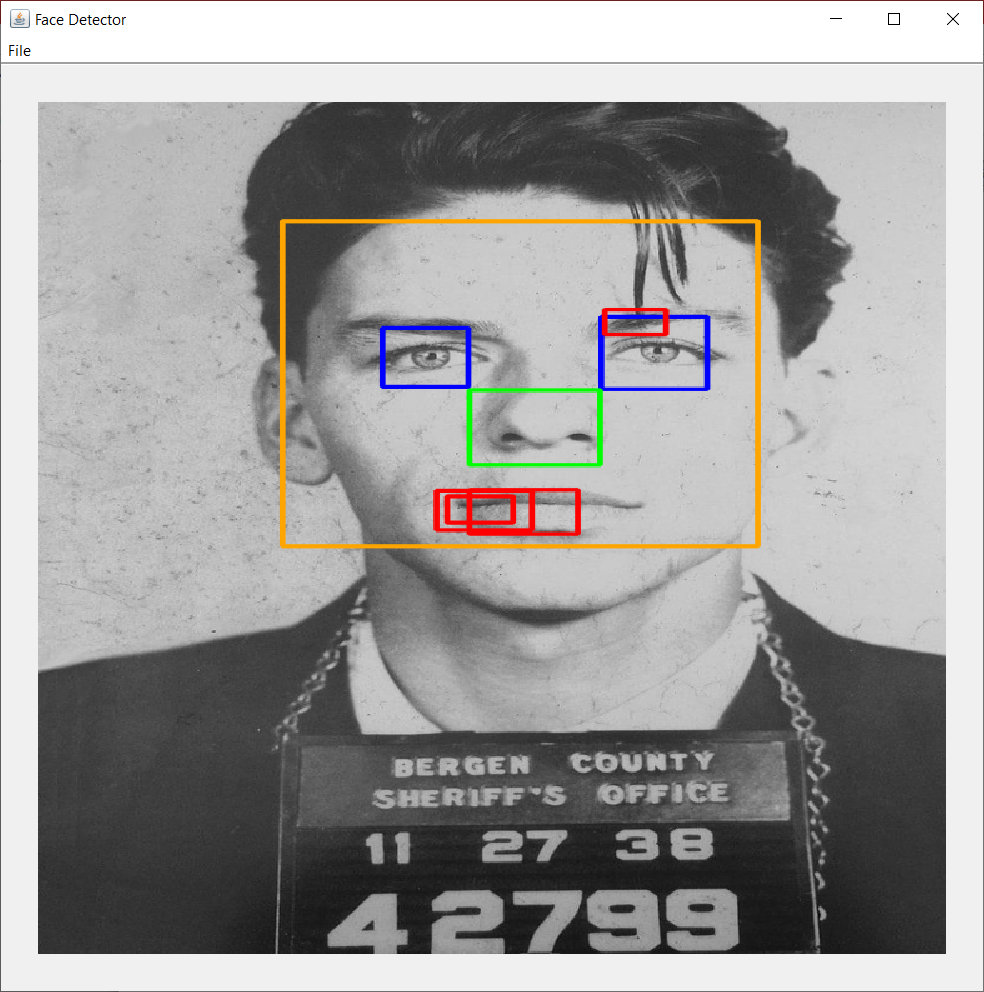
* **Constructor:**
  + În constructorul clasei se încarcă biblioteca nativă OpenCV **System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME)**
  + Se inițializează obiectele CascadeClassifier pentru detectarea fețelor, ochilor, nasului și gurii folosind căile specificate în clasa de constante Constants
* **Funcția convertMatToImage:**
  + Convertește un obiect **Mat** (o imagine OpenCV) într-un obiect **BufferedImage** Java
  + Se determină tipul de imagine (alb-negru sau color) și se alocă un buffer corespunzător
  + Se extrag valorile de pixeli din obiectul **Mat** și se copiază în obiectul **BufferedImage**
* **Funcția detectFeatures:**
  + Primește o imagine de tip **Mat**, un obiect **CascadeClassifier** și o culoare **Scalar**.
  + Detectează caracteristici folosind algoritmul specificat de cascader și evidențiază aceste caracteristici pe imagine
  + Caracteristicile detectate sunt evidențiate cu un dreptunghi de culoarea specificată
* **Funcția detectFaces:**
  + Primește un obiect **File** și un obiect **ImagePanel**.
  + Citește imaginea de la calea specificată în **File**.
  + Detectează fețele în imagine folosind **cascadeFace** și evidențiază aceste fețe cu un dreptunghi portocaliu pe imagine.
  + Pentru fiecare față detectată, se extrage regiunea de interes (**ROI**) și se detectează ochii, nasul și gura folosind metoda **detectFeatures**.
  + Imaginea rezultată este convertită într-un obiect **BufferedImage** și este actualizată în **imagePanel**.

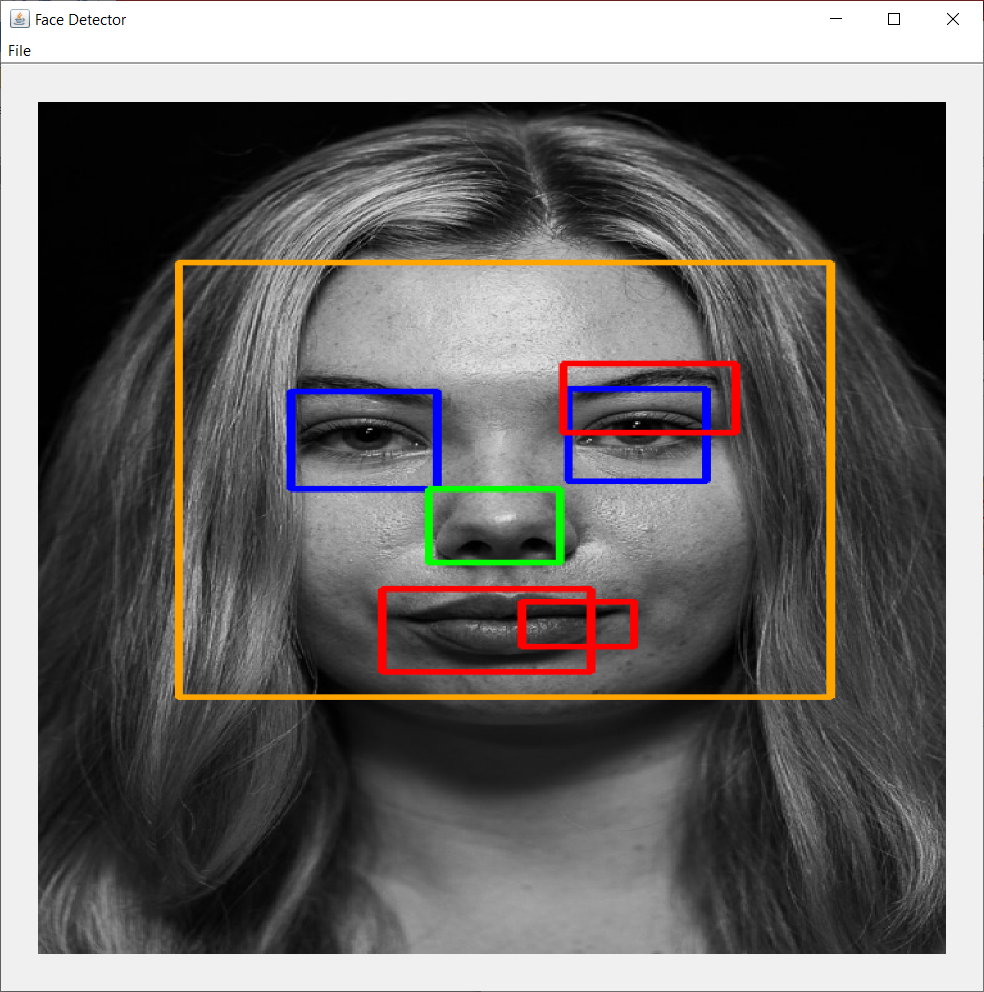
1. **Testare și validare**

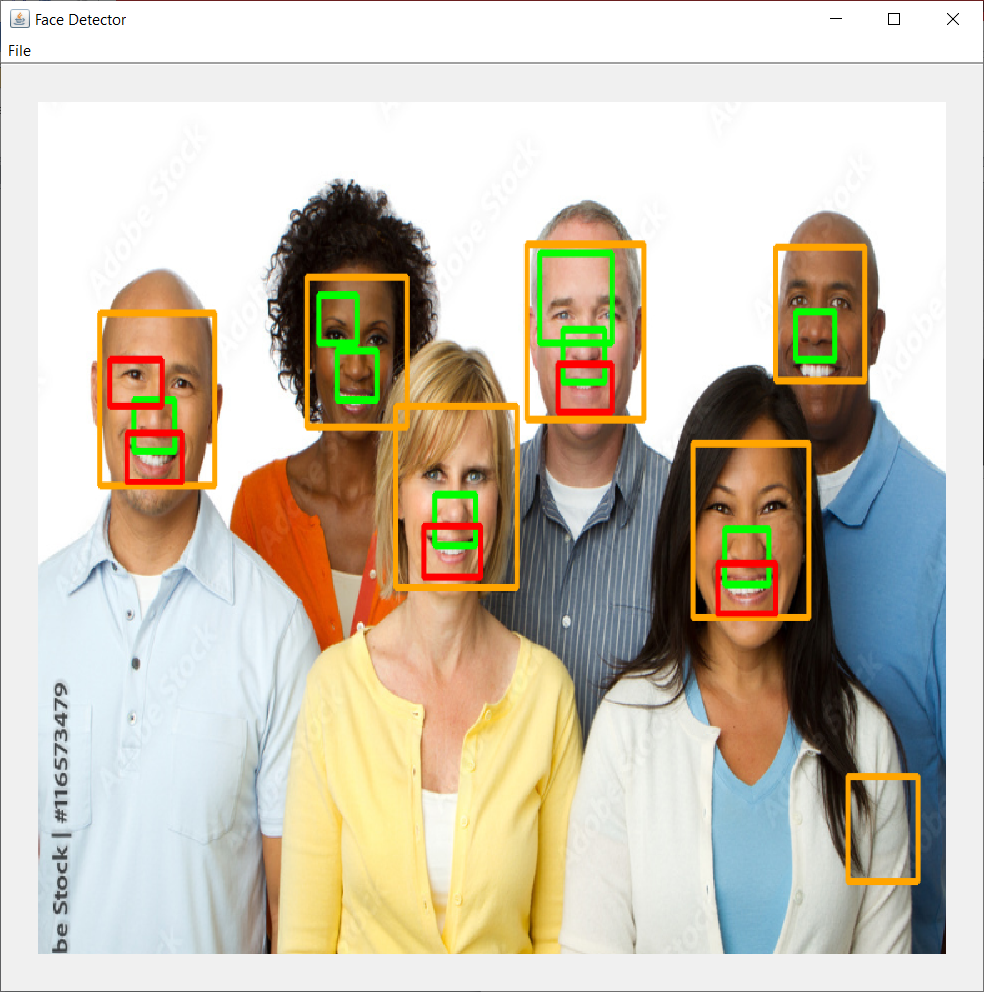
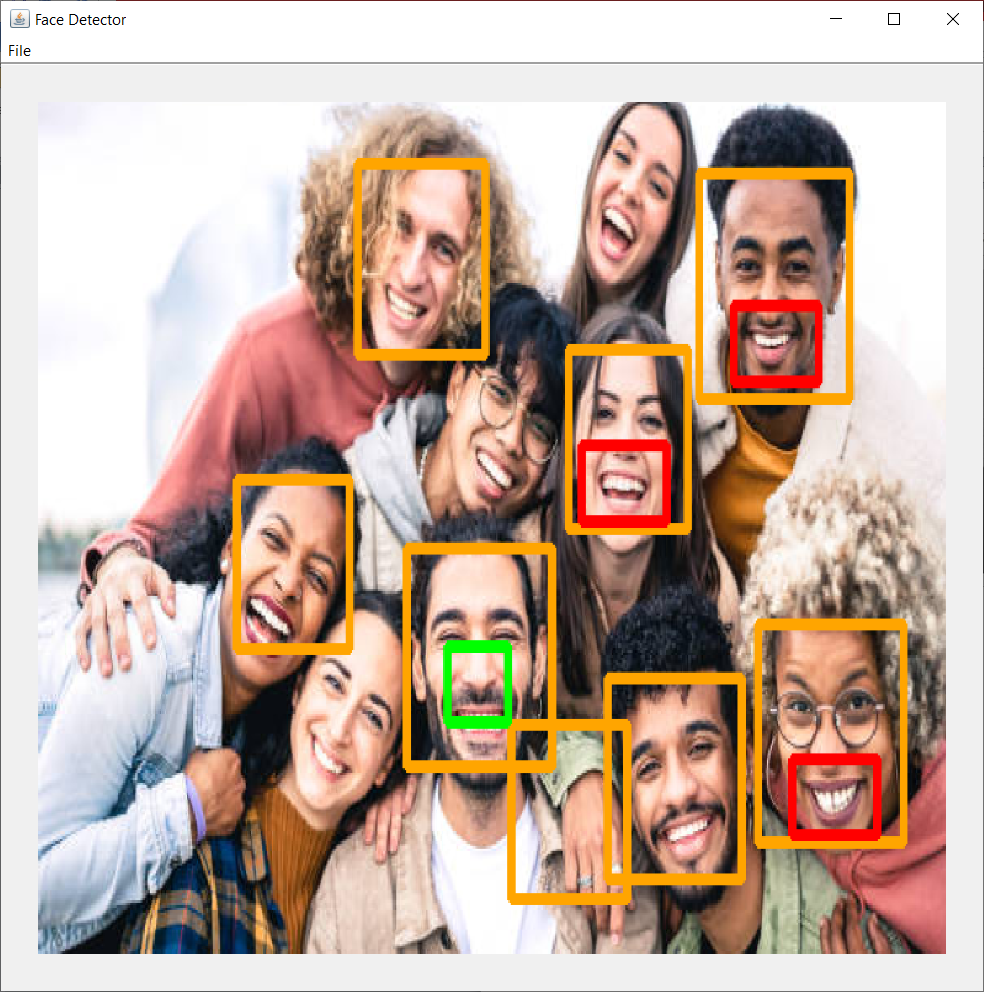
Algoritmii haar cascade utilizați în proiect pentru detectarea feței, ochilor, nasului și gurii au fost pre-antrenați pe un set de 5000 de pozitive și 3000 de negative, iar dupa implementarea proiectului am ales câteva imagini pentru verificarea acestora.

Rezultatele au fost următoarele:



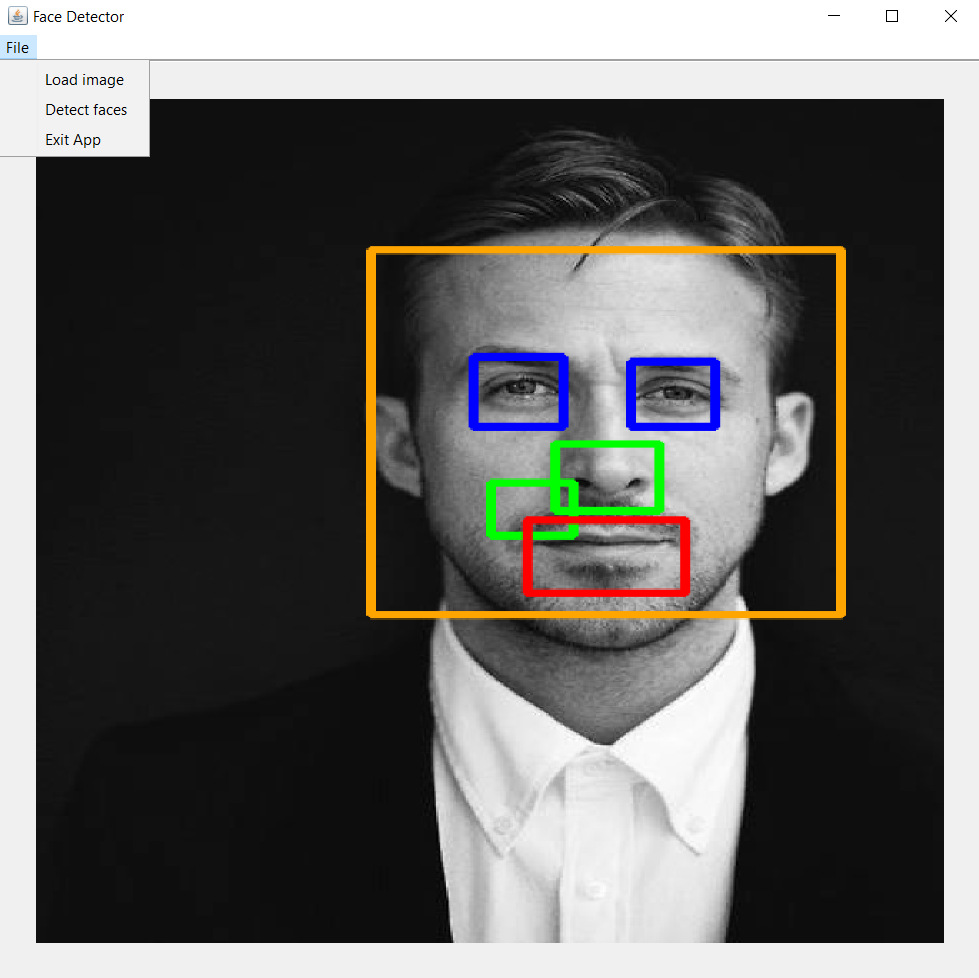






* **se observă acuratețea crescută a recunoașterii ochilor și nasului în pozele cu persoane singure comparativ cu cele cu grupuri de persoane, unde precizia algoritmilor pentru recunoașterea ochilor și nasului scade considerabil**

1. **Concluzii**

****

Interfața grafică a aplicației, cu o poză incărcată și algoritmul rulat

Scopul proiectului a fost crearea unei aplicații pentru recunoașterea fețelor și a trăsăturilor morfologice ale acestora (ochi, nas, gură). Acest lucru a fost îndeplinit mulțumită **OpenCV**, o bibliotecă de funcții informatice specializată pe vedere computerizată și a **haar cascade,** un algoritm care poate detecta obiecte în imagini, dimensiunea și poziția în imagine fiind neimportante.

Posibile îmbunătățiri ce pot fi aduse proiectului cuprind:

* o interfață grafică mai dezvoltată, spre exemplu folosind JavaFX în loc de Swing
* introducerea posibilității de a aplica algoritmul pe imaginile generate în timp real de webcam-ul pc-ului
* antrenarea mai îndelungată și pe o bază de date mai mare a algoritmului haar cascade, pentru rezultate mai precise

1. **Bibliografie**
   * [**https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/**](https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/)
   * [**https://stackoverflow.com/questions/4440283/how-to-choose-the-cascade-file-for-face-detection**](https://stackoverflow.com/questions/4440283/how-to-choose-the-cascade-file-for-face-detection)
   * [**https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade\_eye.xml**](https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade_eye.xml)
   * [**https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade\_eye\_tree\_eyeglasses.xml**](https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade_eye_tree_eyeglasses.xml)
   * [**https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade\_mcs\_mouth.xml**](https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade_mcs_mouth.xml)
   * [**https://github.com/adobe/SimpleSensor/blob/master/simplesensor/collection\_modules/demographic\_camera/classifiers/haarcascades/haarcascade\_eye.xml**](https://github.com/adobe/SimpleSensor/blob/master/simplesensor/collection_modules/demographic_camera/classifiers/haarcascades/haarcascade_eye.xml)
   * [**https://github.com/oreillymedia/Learning-OpenCV-3\_examples/blob/master/haarcascade\_frontalface\_alt.xml**](https://github.com/oreillymedia/Learning-OpenCV-3_examples/blob/master/haarcascade_frontalface_alt.xml)
   * [**https://github.com/facebookarchive/caffe2/issues/755**](https://github.com/facebookarchive/caffe2/issues/755)