



**Universitatea Tehnică a Moldovei**

**PROIECTAREA ȘI DEZVOLTAREA UNUI SISTEM DE EDITARE ȘI  
OPTIMIZARE A IMAGINILOR.**

**DESIGN AND DEVELOPMENT OF AN IMAGE EDITING  
AND OPTIMIZATION SYSTEM.**

**Student:**

**gr. TI-194,  
Ceban Vitalie**

**Coordonator:**

**Cebotari Daria  
asistent universitar**

**Chișinău, 2023**

**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**  
**Universitatea Tehnică a Moldovei**  
**Facultatea Calculatoare, Informatică și Microelectronică**  
**Departamentul Ingineria Software și Automatică**

**Admis la susținere**  
**Șef departament:**  
**FIODOROV Ion dr., conf.univ.**

-----  
„\_\_\_” \_\_\_\_\_ 2023

**Proiectarea și dezvoltarea unui sistem de editare și optimizare a imaginilor.**

**Proiect de licență**

**Student:** \_\_\_\_\_ **Ceban Vitalie, TI-194**  
**Coordonator:** \_\_\_\_\_ **Cebotari Daria, asist. univ.**  
**Consultant:** \_\_\_\_\_ **Cojocaru Svetlana, asist.univ.**

**Chișinău, 2023**

# Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

Programul de studii Tehnologia informației

Aprob  
Șef departament:  
Fiodorov Ion, dr., conf.univ.

„06” octombrie 2022

## CAIET DE SARCINI pentru proiectul de licență al studentului

*Ceban Vitalie*

*(numele și prenumele studentului)*

1. Tema proiectului de licență Proiectarea și dezvoltarea unui sistem de editare și optimizare a imaginilor

confirmată prin hotărârea Consiliului facultății nr. 2 din „06” octombrie 2022

2. Termenul limită de prezentare a proiectului de licență 20.05.2023

3. Date inițiale pentru elaborarea proiectului de licență *Sarcina pentru elaborarea proiectului de diplomă.*

4. Conținutul memoriului explicativ

*Introducere*

*1 Analiza domeniului de studiu*

*2 Modelarea și proiectarea sistemului*

*3 Realizarea sistemului*

*4 Documentarea produsului realizat*

*5 Estimarea costurilor proiectului*

*Concluzii*

## **5. Conținutul părții grafice a proiectului de licență**

Figura 2.1 – Cazurile de utilizare pentru interacțiunea cu sistemul

Figura 2.2 – Diagrama de activitate a salvării unui fișier

**6. Lista consultanților:**

Consultant	Capitol	Confirmarea realizării activității	
		Semnătura consultantului (data)	Semnătura studentului
<i>Cojocaru Svetlana</i>	<i>Standarde tehnologice, Controlul calității, Estimarea costului proiectului</i>		

**7. Data înmânării caietului de sarcini** 02.09.2022

**Coordonator** *Cebotari Daria* \_\_\_\_\_  
*semnătura*

**Sarcina a fost luată pentru a fi executată de către studentul** *Ceban Vitalie*  
02.09.2022  
*semnătura, data*

**PLAN CALENDARISTIC**

Nr. crt.	Denumirea etapelor de proiectare	Termenul de realizare a etapelor	Nota
<i>1</i>	<i>Elaborarea sarcinii, primirea datelor pentru sarcină</i>	<i>02.09.22– 30.09.22</i>	<i>10%</i>
<i>2</i>	<i>Analiza domeniului de studiu</i>	<i>06.10.21– 30.11.21</i>	<i>15%</i>
<i>3</i>	<i>Proiectarea sistemului</i>	<i>01.12.21 – 25.12.21</i>	<i>15%</i>
<i>4</i>	<i>Realizarea sistemului</i>	<i>10.01.22 – 05.03.22</i>	<i>35%</i>
<i>5</i>	<i>Descrierea sistemului</i>	<i>06.03.22– 01.04.22</i>	<i>10%</i>
<i>6</i>	<i>Estimarea costurilor sistemului</i>	<i>02.04.22– 15.04.22</i>	<i>15%</i>
<i>7</i>	<i>Finisarea proiectului</i>	<i>16.04.22– 14.05.22</i>	<i>5%</i>

**Student** \_\_\_\_\_ *Ceban Vitalie* ( \_\_\_\_\_ )

**Coordonator de proiect de licență** *Cebotari Daria* ( \_\_\_\_\_ )

# UNIVERSITATEA TEHNICĂ A MOLDOVEI

## FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ DEPARTAMENTUL INGINERIA SOFTWARE ȘI AUTOMATICĂ PROGRAMUL DE STUDII TEHNOLOGIA INFORMAȚIEI

### AVIZ

la proiectul de licență

**Titlul:** Proiectarea și dezvoltarea unui sistem de editare și optimizare a imaginilor.

Studentul Ceban Vitalie gr. TI-194

**1. Actualitatea temei:** tot mai mulți oameni utilizează camere digitale pentru a face fotografii și au nevoie de un instrument eficient pentru a le edita și îmbunătăți. În plus, popularitatea rețelelor de socializare și a altor platforme online de partajare a imaginilor face ca oamenii să dorească să aibă imagini mai bune și mai atrăgătoare pentru a le împărtăși cu alții.

**2. Caracteristica proiectului de licență:** Aplicația a fost creată pentru ușurarea lucrului cu imaginile.

**3. Analiza prototipului:** Aplicația data este formată pentru a importa imagini de pe calculator care pot fi editate și salvate cu modificări.

**4. Estimarea rezultatelor obținute:** Acest program este creat pentru editarea imaginilor. Este un program ușor în utilizare și intuitiv pentru utilizatori nefamiliarizați cu aceste tipuri de aplicații.

**5. Corectitudinea materialului expus:** Materialul expus este prezentat prin referințe ale unor surse ce au fost scrise de persoane ce dețin experiență în domeniul Tehnologiilor Informaționale.

**6. Calitatea materialului grafic:** Proiectul este prezentat prin: diagrame, tabele, interfețe ale aplicației.

**7. Valoarea practică a proiectului:** Este destinat pentru utilizatorilor ce dețin calculatoare persoanele cu sistemul de operare Windows. Aceasta aplicație poate fi utilizată de orice utilizator familiar cu un calculator.

**8. Observații și recomandări:** Cerințele față de teza de licență au fost îndeplinite în totalitate. Observații nu sunt.

**9. Caracteristica studentului și titlul conferit :** Studentul Ceban Vitalie a dat dovadă de profesionalism în elaborarea lucrării, a respectat cerințele impuse și a manifestat exigență în elaborarea și calitatea tezei de licență. Din cele relatate, urmează că lucrarea de licență poate fi admisă spre susținere, cu nota 10

*Lucrarea în forma electronică corespunde originalului prezentat către susținere publică.*

Coordonatorul proiectului de licență \_\_\_\_\_ Cebotari Daria, asist.univ

*semnătura, data*

## REZUMAT

Lucrarea dată este compusă din 5 capitole distincte, fiecare capitol deținând mai multe subcapitole cu descrierea mai detaliată a conținutului.

În primul capitol, intitulat "Analiza domeniului de studiu", vom explora în profunzime domeniul în care se încadrează aplicația de editare a imaginilor. Vor fi prezentate argumente solide care susțin importanța temei alese și se va realiza o comparație riguroasă cu alte sisteme deja existente pe piață. De asemenea, în acest capitol vor fi descrise cu mare atenție scopul, obiectivele și cerințele sistemului de editare a imaginilor. Aceste informații sunt cruciale pentru dezvoltarea unei aplicații eficiente și satisfăcătoare pentru utilizatori. O analiză amănunțită a domeniului de studiu este esențială pentru a înțelege pe deplin cerințele și nevoile utilizatorilor. În acest fel, se pot dezvolta soluții personalizate și eficiente pentru nevoile fiecărui client în parte. Prin compararea cu alte sisteme existente, putem identifica punctele slabe și punctele forte ale acestora și, astfel, putem dezvolta o aplicație care să ofere o experiență superioară utilizatorilor și să răspundă nevoilor lor în mod eficient. De asemenea, descrierea clară a scopului, obiectivelor și cerințelor sistemului va asigura o dezvoltare coerentă și eficientă a aplicației. Aceste informații sunt utile pentru a identifica prioritățile dezvoltării și pentru a asigura că produsul final îndeplinește cerințele utilizatorilor.

În al doilea capitol „Modelarea și proiectarea sistemului informatic” este realizată descrierea comportamentală și structurală a sistemului, se creează o imagine generală asupra sistemului și se descriu stările și tranzițiile sistemului. Pentru aceasta sunt create diagrame UML în care sunt arătate toate aceste stări și cum au loc tranzițiile între ele. Diagramele sunt de mai multe tipuri, în dependența de stările care sunt necesare spre vizualizare. Sunt create scenarii de utilizare a aplicației care descriu fluxurile de mesaje și legăturile dintre componentele sistemului.

Capitolul 3 „Realizarea sistemului” se axează pe realizarea aplicației în limbajul ales anterior. În acest capitol se efectuează descrierea la nivel de cod pe module a funcționalului. Funcțiile cele mai importante sunt arătate și explicate din punct de vedere a funcționalului. De asemenea, în acest capitol este descris cum are loc testarea sistemului, fiecare tip de test fiind explicat pe ce se bazează și ce testează.

Cel de al 4 capitol este „Documentarea produsului realizat”. În acest capitol este realizată descrierea cerințelor tehnice pentru aplicație, cum se găsește aplicația, cum se instalează. Se realizează descrierea utilizării aplicației la nivel de utilizator, fiind creat un manual de utilizare cu explicarea fiecărui component grafic și cum se utilizează el.

Ultimul capitol este „Estimarea costurilor proiectului”. În acest capitol este realizată estimarea costurilor totale pentru realizarea unei astfel de aplicații. Se efectuează descrierea fiecărei categorii care intră în costul total, pentru fiecare fiind arătate exemple reale cu prețuri actuale la momentul descrierii.

## **ABSTRACT**

The given work consists of 5 distinct chapters, each chapter having several sub-chapters with a more detailed description of the content.

In the first chapter, entitled "Analysis of the study domain", we will explore in depth the domain in which the image editing application is included. Solid arguments supporting the importance of the chosen topic will be presented, and a rigorous comparison will be made with other existing systems on the market. Also, in this chapter, the purpose, objectives, and requirements of the image editing system will be described with great attention. This information is crucial for developing an efficient and satisfactory application for users. A thorough analysis of the study domain is essential to fully understand the requirements and needs of users. In this way, personalized and efficient solutions can be developed for the needs of each client. By comparing with other existing systems, we can identify their weaknesses and strengths, and thus develop an application that provides a superior user experience and efficiently meets their needs. Also, a clear description of the purpose, objectives, and requirements of the system will ensure a coherent and efficient development of the application.

In the second chapter, "Modeling and designing the computer system", the behavioral and structural description of the system is created, an overview of the system is presented, and the states and transitions of the system are described. For this, UML diagrams are created, showing all these states and how transitions occur between them. The diagrams are of several types, depending on the states required for visualization. Usage scenarios of the application are created, describing the message flows and connections between system components.

Chapter 3, "Implementing the system," focuses on creating the application in the previously chosen language. In this chapter, the description of the functional modules is performed at the code level. The most important functions are shown and explained from a functional point of view. Also, in this chapter, it is described how system testing takes place, with each type of test being explained on what it is based and what it tests.

The fourth chapter is "Documentation of the created product." In this chapter, the technical requirements for the application are described, including how to find and install the application. The description of the application usage is done at the user level, with a user manual created explaining each graphical component and how to use it.

The final chapter is "Estimating project costs." In this chapter, the total costs for creating such an application are estimated. The description of each category that is included in the total cost is performed, and real examples with current prices at the time of description are shown.



## CUPRINS

INTRODUCERE .....	10
1 ANALIZA DOMENIULUI DE STUDIU.....	11
1.1 IMPORTANȚA TEMEI .....	13
1.2 SISTEME SIMILARE CU PROIECTUL REALIZAT .....	14
1.3 SCOPUL, OBIECTIVELE ȘI CERINȚELE SISTEMULUI .....	20
2 MODELAREA ȘI PROIECTAREA SISTEMUL INFORMATIC .....	22
2.1 DESCRIEREA COMPORTAMENTALĂ A SISTEMULUI.....	23
2.1.1 IMAGINEA GENERALĂ ASUPRA SISTEMULUI .....	24
2.1.2 MODELAREA VIZUALĂ A FLUXURILOR.....	25
2.1.3 STĂRILE DE TRANZACȚIE A SISTEMULUI .....	27
2.1.4 DESCRIEREA SCENARIILOR DE UTILIZARE A APLICAȚIEI.....	28
2.1.5 FLUXURILE DE MESAJE ȘI LEGĂTURILE DINTRE COMPONENTELE SISTEMULUI .....	29
2.2 DESCRIEREA STRUCTURALĂ A SISTEMULUI.....	30
2.2.1 DESCRIEREA STRUCTURII STATICE A SISTEMULUI .....	31
2.2.2 RELAȚIILE DE DEPENDENȚĂ ÎNTRE COMPONENTELE SISTEMULUI .....	32
2.2.3 MODELAREA ECHIPAMENTELOR MEDIULUI DE IMPLEMENTARE .....	33
3 REALIZAREA SISTEMULUI.....	34
3.1 DESCRIEREA LA NIVEL DE COD PE MODULE .....	35
3.2 TESTAREA SISTEMULUI .....	37
CONCLUZII .....	39
ANEXA A.....	42

## INTRODUCERE

În zilele noastre, fotografiile sunt o parte importantă a vieții noastre și le utilizăm pentru a ne reaminti de momente speciale, a comunica cu prietenii și familia și pentru a exprima creativitatea noastră. Cu toate acestea, adesea fotografiile noastre nu apar așa cum ne-am dori și de aceea, este necesar să apelăm la instrumente de editare a imaginilor. O aplicație de editare a imaginilor este un instrument esențial pentru orice persoană care dorește să îmbunătățească calitatea imaginilor sale sau să le transforme într-un mod unic. În plus, aceste aplicații sunt esențiale și pentru profesioniști din domeniul fotografiei și designului, care utilizează imagini pentru a-și crea produsele și serviciile. Obiectivul principal al unei astfel de aplicație de editare a imaginilor este de a furniza utilizatorilor o gamă largă de instrumente și funcții de editare pentru a edita imaginile în diferite moduri. O astfel de aplicație trebuie să fie ușor de utilizat și să ofere o gamă largă de instrumente și funcții de editare pentru a permite utilizatorilor să editeze imaginile în diferite moduri. Pentru a atinge acest obiectiv, o aplicație de editare a imaginilor trebuie să aibă o interfață prietenoasă cu utilizatorul care permite utilizatorilor să navigheze ușor și să acceseze diversele instrumente de editare disponibile cum ar fi decuparea, redimensionarea, rotirea, răsturnarea, ajustarea culorii și aplicarea unui filtru. Aceste caracteristici ar trebui să fie ușor accesibile și simple de utilizat, chiar și pentru utilizatorii care nu sunt familiarizați cu editarea imaginilor. Aplicația trebuie să poată gestiona, de asemenea, diferite formate de fișiere, inclusiv fișiere JPEG, PNG și GIF. De asemenea, aplicația trebuie să ofere opțiuni de securitate și confidențialitate, permițând utilizatorilor să-și stocheze imaginile în siguranță pe propriile dispozitive sau pe servere locale securizate.

O aplicație de editare a imaginilor face parte din domeniul tehnologiilor multimedia și este esențială pentru a crea conținut multimedia de înaltă calitate și pentru a-l edita în mod eficient și creativ. În această eră digitală, tehnologiile multimedia sunt din ce în ce mai importante și influențează în mod semnificativ modul în care comunicăm și consumăm informații. Prin urmare, cercetarea și dezvoltarea în domeniul tehnologiilor multimedia este esențială pentru a rămâne la curent cu tendințele și pentru a oferi soluții inovatoare și eficiente.

În aceasta lucrare privind dezvoltarea unei aplicații de editare a imaginilor, s-a adoptat o abordare de cercetare aplicată, în care s-au utilizat metode și tehnici de cercetare specifice pentru a dezvolta o aplicație eficientă și funcțională. Printre metodele utilizate se numără cercetarea bibliografică, interviurile cu utilizatori potențiali și testarea prototipurilor. Scopul acestei cercetări a fost de a dezvolta o aplicație de editare a imaginilor care să răspundă nevoilor utilizatorilor, într-un mod intuitiv și eficient.

În următorul raport, vom examina o astfel de aplicație de editare a imaginilor și vom discuta despre caracteristicile sale, precum și importanța utilizării acesteia în lumea noastră digitală în continuă schimbare.

# 1 ANALIZA DOMENIULUI DE STUDIU

Aplicația face parte din domeniul tehnologiilor multimedia. Aplicația este de tip desktop având un GUI ce permite activitățile de baza pentru o aplicație ce ar permite optimizarea unei imagini. Astfel de aplicații sunt utilizate pentru reducerea dimensiunilor imaginilor.

Multimedia se referă la conținut și media care utilizează o combinație de diferite forme de conținut, inclusiv date vizuale codificate, audio, text și formate lingvistice. Termenul poate fi folosit ca substantiv (mediu cu forme multiple de conținut) sau un adjectiv care descrie un mediu cu mai multe formate de conținut descriptiv. Termenul este folosit în contrast cu mediile care utilizează doar ecrane rudimentare de computer, cum ar fi: afișarea numai a textului sau a materialelor realizate manual sau tipărite în formate tradiționale. Multimedia include orice combinație de conținut sub formă de text, audio, imagine, animație, video sau formă interactivă. În format electronic, poate fi citit pe PC-uri, tablete, smartphone-uri, e-reader etc.

Multimedia este de obicei înregistrată, redată, afișată sau accesată prin intermediul dispozitivelor de procesare a conținutului informațional, cum ar fi dispozitivele electronice și de calcul prin Internet, dar poate face, de asemenea, parte dintr-un spectacol live. Multimedia (ca adjectiv) se referă și la dispozitivele media electronice utilizate pentru a stoca și a experimenta conținut multimedia. Multimedia este diferită de media amestecată din cadrul artelor frumoase; incluzând audio, spre exemplu, are o sferă mai vastă. Termenul „rich media” este sinonim cu media interactivă, iar „hipermedia” este o altă aplicație multimedia [1].

Cunoașterea elementelor fundamentale ale graficii pe computer este esențială pentru inginerii, oamenii de știință, artiștii vizuali, designerii, fotografi, animatorii și nu numai. Apariția unor noi cerințe a condus la dezvoltarea mai rapidă a unor aplicații software, făcându-le mai intuitive și mai structurate de utilizat. Datorită computerului, puteți avea diferite variații de culoare, forme, configurații etc. în câteva secunde. În baza tehnologiilor graficii computerizate s-au dezvoltat:

- interfața de utilizator, GUI;
- proiectarea digitală în arhitectură și grafica industrială;
- efecte vizuale specializate, cinematografia digitală;
- grafica de computer pentru filme, animație, televiziune;
- proiecte multimedia, proiecte interactive;
- fotografia digitală și posibilitățile avansate de prelucrare a fotografiei;
- grafica și pictura digitală (cu 2 laturi esențiale – imitarea materialelor tradiționale și noile instrumente de lucru digitale) [2].

Imaginile rastru sunt stocate pe PC sub forma unei grile de elemente de imagine sau pixeli. Acești pixeli conțin informații despre culoarea și luminozitatea imaginii. Editorii de imagini pot modifica pixelii pentru

a îmbunătăți imaginea în multe feluri. Pixelii pot fi modificați ca grup sau individual, de către algoritmi sofisticati din editorii de imagini. Este mai ușor să rasterizați o imagine vectorială decât să vectorizați o imagine rastru; modul de a proceda despre vectorizarea unei imagini rastru este punctul central al multor cercetări în domeniul vederii computerizate. Imaginile vectoriale pot fi modificate mai ușor, deoarece conțin descrieri ale formelor pentru o rearanjare ușoară. De asemenea, sunt scalabile, fiind rasterizabile la orice rezoluție [3].

Multe formate de fișiere imagine utilizează compresia datelor pentru a reduce dimensiunea fișierului și a economisi spațiu de stocare. Comprimarea digitală a imaginilor poate avea loc în camera foto sau poate fi făcută în computer cu editorul de imagini. Când imaginile sunt stocate în format JPEG, compresia a avut deja loc. Atât camerele foto, cât și programele de calculator permit utilizatorului să seteze nivelul de compresie.

Unii algoritmi de compresie, cum ar fi cei utilizați în format de fișier PNG, sunt fără pierderi, ceea ce înseamnă că nu se pierde nicio informație atunci când fișierul este salvat. În schimb, formatul de fișier JPEG mai popular folosește un algoritm de compresie cu pierderi (bazat pe codificarea cu transformare cosinus discretă) prin care cu cât compresia este mai mare, cu atât se pierde mai multă informație, reducând în cele din urmă calitatea imaginii sau detaliile care nu pot fi restaurate. JPEG folosește cunoștințele despre modul în care creierul și ochii umani percep culoarea pentru a face această pierdere de detalii mai puțin vizibilă [4].

Aplicația are să fie una desktop. Prin definiție, o aplicație desktop este un software care poate fi instalat pe un singur computer (laptop sau desktop) și utilizat pentru a îndeplini o anumită sarcină. Unele aplicații desktop pot fi utilizate și de mai mulți utilizatori într-un mediu de rețea [5].

Într-o lume în care totul se îndreaptă către aplicații ușoare și portabile, aplicațiile desktop încă mai au sens în unele situații. Acest lucru se datorează faptului că pot suporta funcții avansate în situații foarte specializate. De asemenea, este posibil ca afacerea dvs. să nu dorească o aplicație care să dețină date sensibile în cloud.

Iată o privire mai atentă asupra avantajelor și dezavantajelor aplicațiilor desktop:

- opțiuni de funcționalitate – puteți adăuga aproape orice caracteristică dorită la o aplicație desktop, deoarece aplicația rulează local și nu în cloud;
- mai ușor de lucrat offline – deși puteți adăuga funcționalitate offline la o aplicație web, vor exista limitări. Cu toate acestea, puteți proiecta o aplicație desktop pentru a funcționa exact la fel offline ca și online;
- potrivit în situațiile în care cloud-ul nu este dorit – există unele situații în care cloud-ul este considerat nepotrivit. În aceste situații, aplicațiile desktop sunt o soluție viabilă și eficientă [6].

## 1.1 IMPORTANȚA TEMEI

În urma analizei sistemelor similare cu sistemul propus am ajuns la concluzia că toate sistemele disponibile pe piața sunt contra plată. Pentru utilizarea pe deplin este necesară procurarea totală a aplicației sau a unui abonament pe un termen limitat. Un alt neajuns a sistemelor similare este faptul că 2/3 din ele necesită deținerea unei conexiuni la internet pentru utilizarea lor. Acest neajuns adesea înseamnă că deși utilizatorul v-a fi conectat nu se garantează utilizarea comodă din cauza vitezei. Pe lângă aceste probleme, am observat că majoritatea sistemelor ce utilizează imagini stochează acestea într-o bază de date, ceea ce poate duce la o nevoie crescută de spațiu de stocare. Prin urmare, aceste sisteme necesită costuri suplimentare pentru a putea fi utilizate, din cauza spațiului mare necesar de pe servere. În acest context, aplicația propusă are ca scop reducerea dimensiunilor imaginilor pentru economisirea spațiului. Ea va efectua acest lucru prin compresarea imaginilor sau prin utilizarea altor algoritmi adecvați. Acest aspect este important deoarece, prin reducerea dimensiunilor imaginilor, spațiul necesar pentru stocarea acestora poate fi diminuat, ceea ce poate reduce costurile de stocare necesare unui sistem de editare a imaginilor. Un alt funcțional prezent în aplicație sunt filtrele încorporate, printre filtrele prezente pot fi identificate următoarele:

- sepia;
- alb-negru;
- sharpen.

Un avantaj al utilizării filtrelor deja implementate este că acestea sunt de obicei ușor de utilizat, chiar și pentru cei fără experiență în editarea imaginilor. Majoritatea filtrelor pot fi aplicate cu doar câteva clicuri sau atingeri, iar utilizatorul poate vedea instantaneu rezultatele. Un alt beneficiu este că filtrele pot economisi timp și efort. În loc să petreacă ore ajustând luminozitatea, contrastul și alte setări manual, utilizatorul poate pur și simplu să aleagă un filtru care obține efectul dorit. În plus, aplicația este foarte intuitivă și ușor de folosit, ceea ce face ca editarea imaginilor să fie o activitate plăcută și relaxantă. Pe lângă acestea, aplicația propusă are și un design modern și atractiv, care oferă o experiență de utilizare plăcută și confortabilă.

De asemenea, aplicația propusă vine cu o soluție la problemele de securitate și confidențialitate ale utilizatorilor. Mulți utilizatori sunt îngrijorați cu privire la faptul că imaginile lor pot fi stocate pe servere terțe, ceea ce poate duce la riscuri de securitate și încălcări ale confidențialității datelor. Pentru a rezolva această problemă, aplicația propusă oferă opțiunea de stocare locală, permițând utilizatorilor să-și păstreze imaginile pe propriile dispozitive sau pe servere locale securizate. Acest lucru va oferi utilizatorilor un control mai mare asupra datelor lor.

## 1.2 SISTEME SIMILARE CU PROIECTUL REALIZAT

În urma analizei sistemelor existente la momentul actual am găsit 3 sisteme asemănătoare cu sistemul dat:

- Kraken.io;
- PIXLR;
- Photoshop.

Kraken.io este un instrument de optimizare a imaginilor. Cu Kraken.io, puteți optimiza cantități mari de fișiere JPEG, PNG și GIF animate.

Spre deosebire de alt optimizator de imagine, Kraken.io optimizează fișierele pentru cea mai mică dimensiune. Adică, cu Kraken.io, se obțin întotdeauna cele mai mici versiuni ale fișierelor introduse. Apoi, fotografiile comprimate se pot descărca pe rând sau în format .zip.

Kraken.io vă permite, de asemenea, să exportați fișiere în Dropbox sau să importați fișiere din Box, Dropbox sau Google Drive.

Versiunea gratuită permite să comprimați fișiere foto de până la 32 MB fiecare și până la un total de 100 MB de fotografii. Funcționarea acestui instrument depinde de dimensiunea inițială a imaginii. În figura 1.1 Kraken.io Pagina principală am reprezentat pagina de pornire a acestui instrument.

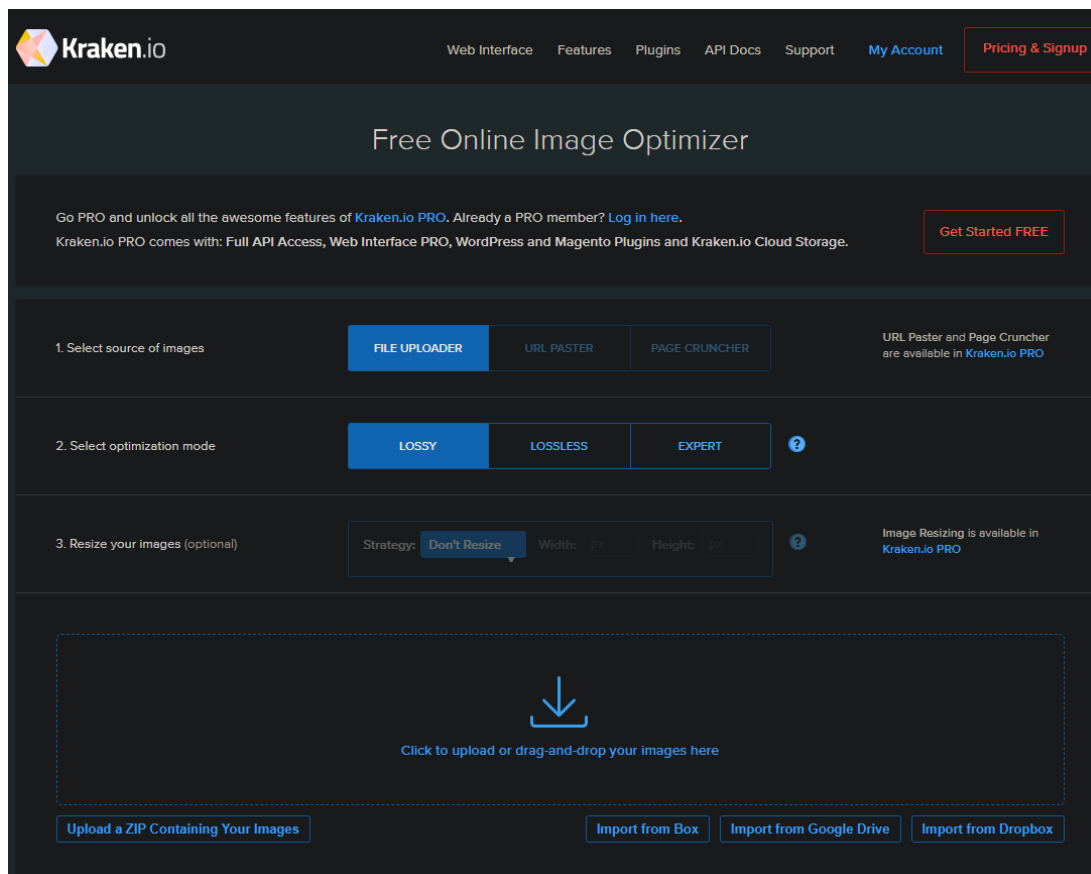


Figura 1.1 - Kraken.io Pagina principală

Pixlr este unul dintre cele mai populare programe de editare a imaginilor disponibile astăzi pe piață. Cele mai recente ediții de Pixlr oferă o experiență de editare de nivel înalt. De la conversia imaginii la multiple alte opțiuni de editare, Pixlr are o soluție pentru orice. Pentru lucrări rapide, acest instrument vine și cu șabloane de colaj prefabricate pentru o editare mai rapidă. Unul dintre motivele pentru care profesioniștii optează pentru Pixlr este datorită funcției Instant de eliminare a fundalului bazată pe inteligență artificială. În figura 1.2 PIXLR Pagina principală am reprezentat pagina de pornire a acestui instrument [7].

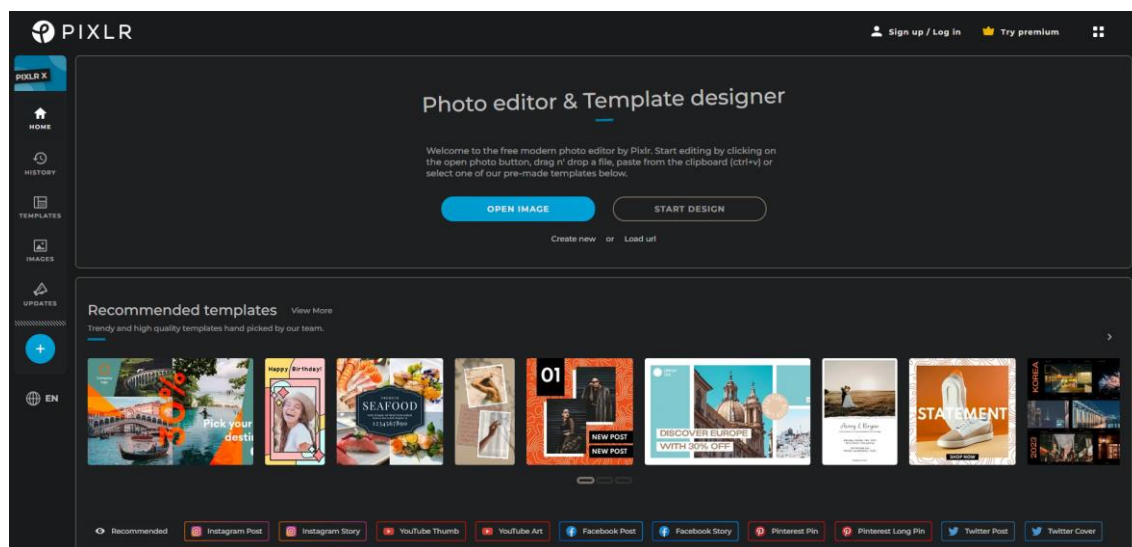


Figura 1.2 - PIXLR Pagina principală

Pixlr este un o aplicatie de editare bazata pe instrumente și utilite de editare a imaginilor cloud, inclusiv o serie de editori foto și un serviciu de partajare a fotografiilor. Suita este destinată din gama de editare foto simplă până la avansată. Dispune de trei planuri de abonament care includ Free, Premium și Team, planurile sunt reprezentate în figura 1.3 PIXLR Abonamente.

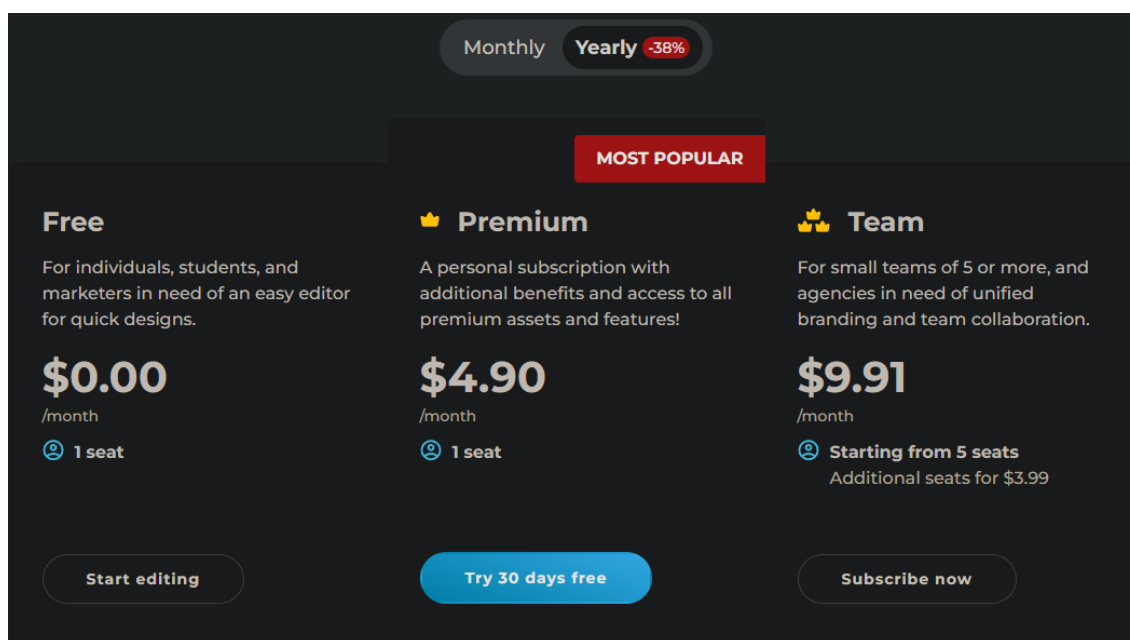


Figura 1.3 - PIXLR Abonamente

Platforma poate fi folosită pe desktop, dar și pe smartphone-uri și tablete. Pixlr este compatibil cu diverse formate de imagine, cum ar fi JPEG, PNG, WEBP, GIF, PSD (Document Photoshop) și PXZ (format nativ de document Pixlr)

Adobe Photoshop este un alt software folosit pentru editarea imaginilor digitale pe calculator, program produs și distribuit de compania americană Adobe Systems și care se adresează în special profesioniștilor domeniului.

Adobe Photoshop, așa cum este cunoscut astăzi, este vârful de lance al gamei de produse software pentru editare de imagini digitale, fotografii, grafică pentru tipar, video și Web de pe piață. Photoshop este un program cu o interfață intuitivă și care permite o multitudine extraordinară de modificări necesare în mod curent profesioniștilor și nu numai: editări de luminozitate și contrast, culoare, focalizare, aplicare de efecte pe imagine sau pe zone (selecții), retușare de imagini degradate, număr arbitrar de canale de culoare, suport de canale de culoare pe 8, 16 sau 32 biți, efecte third-party etc. Există situații specifice pentru un profesionist în domeniu când alte pachete duc la rezultate mai rapide, însă pentru prelucrări generale de imagine, întrucât furnizează instrumente solide, la standard industrial, Photoshop este efectiv indispensabil. Singurul minus este prețul, pentru o simplă editare utilizatorul este obligat să procure aplicația care are un preț destul de ridicat. În figura 1.3 Interfața Photoshop am reprezentat interfața principală a aplicației Photoshop [8].



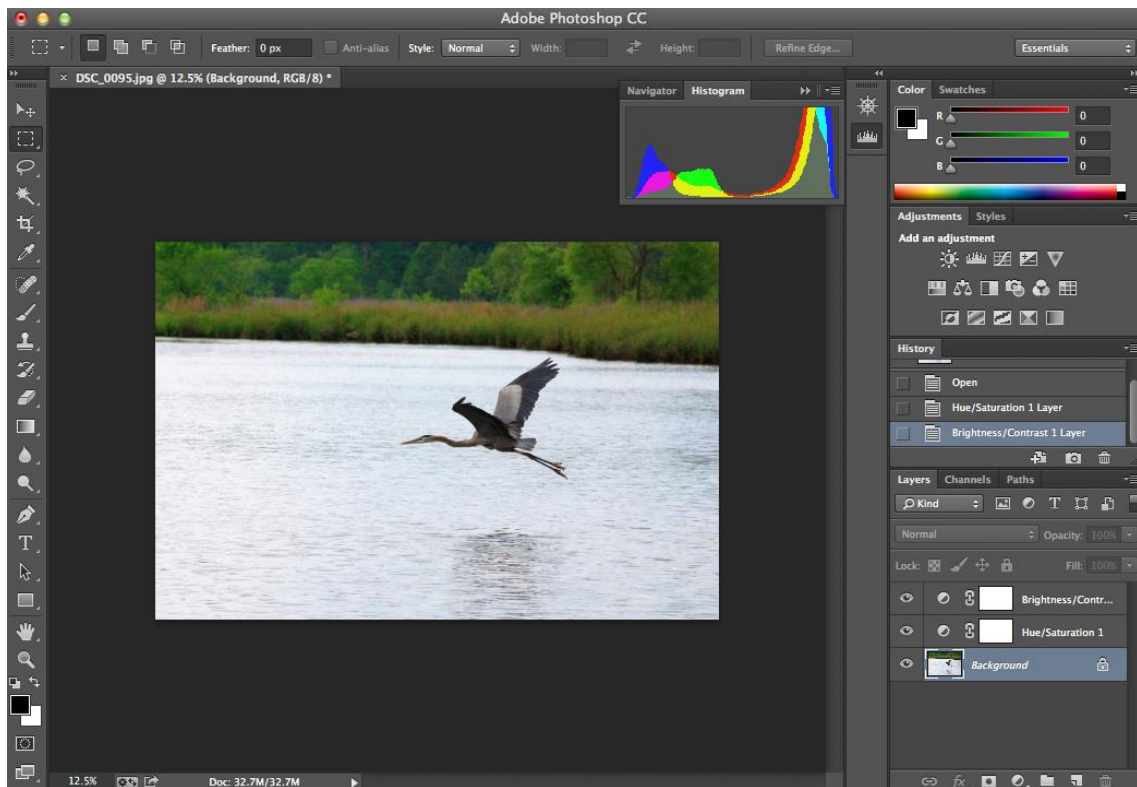


Figura 1.3 - Interfața Photoshop

Cu ajutorul a Photoshop o imagine poate fi editata in orice mod posibil datorita faptului ca Photoshop deține un funcțional larg. Câteva exemple a posibilităților ar fi:

- balansarea culorii;
- filtre artistice;
- efecte picturale;
- retușarea imaginilor.

Balansarea culorii presupune crearea unui efect personalizat prin crearea unui strat de ajustare. Valorile anumitor culori pot fi ridicate sau scăzute pentru modificarea acestora astfel obținându-se alte culori sau modificate contrastul culorilor precedente. Acest funcțional este reprezentat in figura 1.4 Photoshop Balanță de culori.

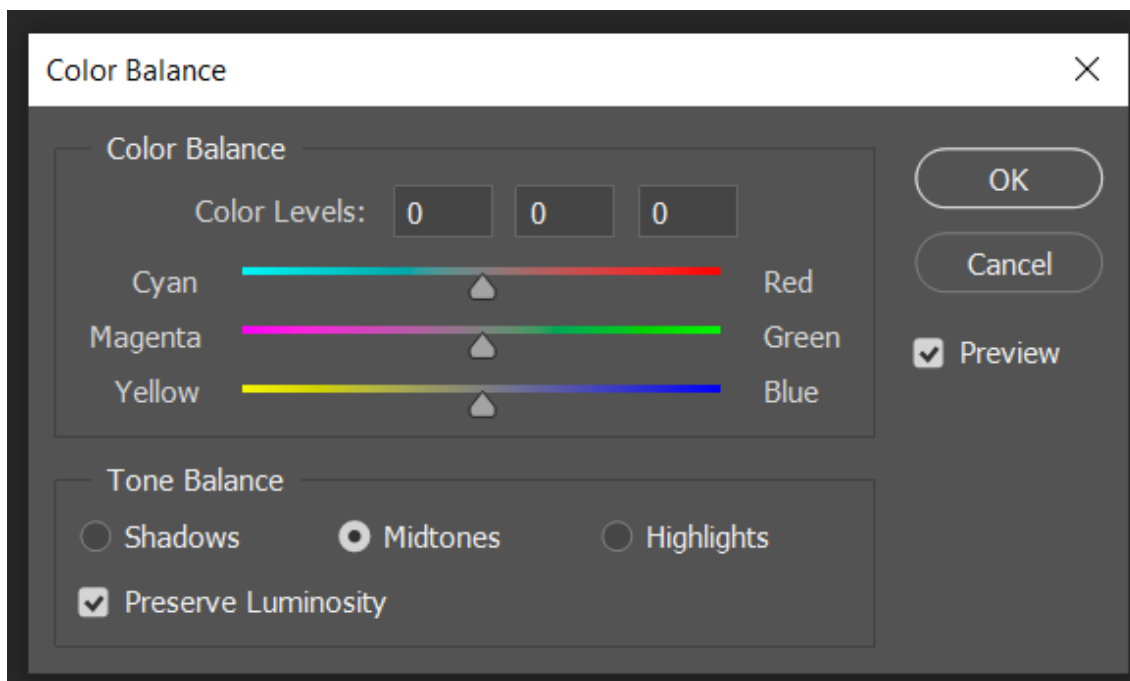


Figura 1.4 - Photoshop Balanță de culori

Oricât de mult nu ne-am dori să obținem imagini perfecte direct de la cameră, deseori este nevoie de post-procesare. Fotografii folosesc filtre și efecte foto atât pentru a îmbunătăți calitatea imaginii, cât și pentru a transforma complet pozele în funcție de imaginația lor. Folosind instrumentele potrivite, puteți realiza fotografii artistice și vă puteți exprima creativitatea.

Filtrele Photoshop implicite sunt dedicate ajustărilor de bază și efectelor comune. Ele pot fi combinate pentru a crea propriile efecte, dar acest lucru necesită timp și energie. Meniul de alegere a filtrelor in Photoshop este aratat in figura 1.5 Filtrele Photoshop.

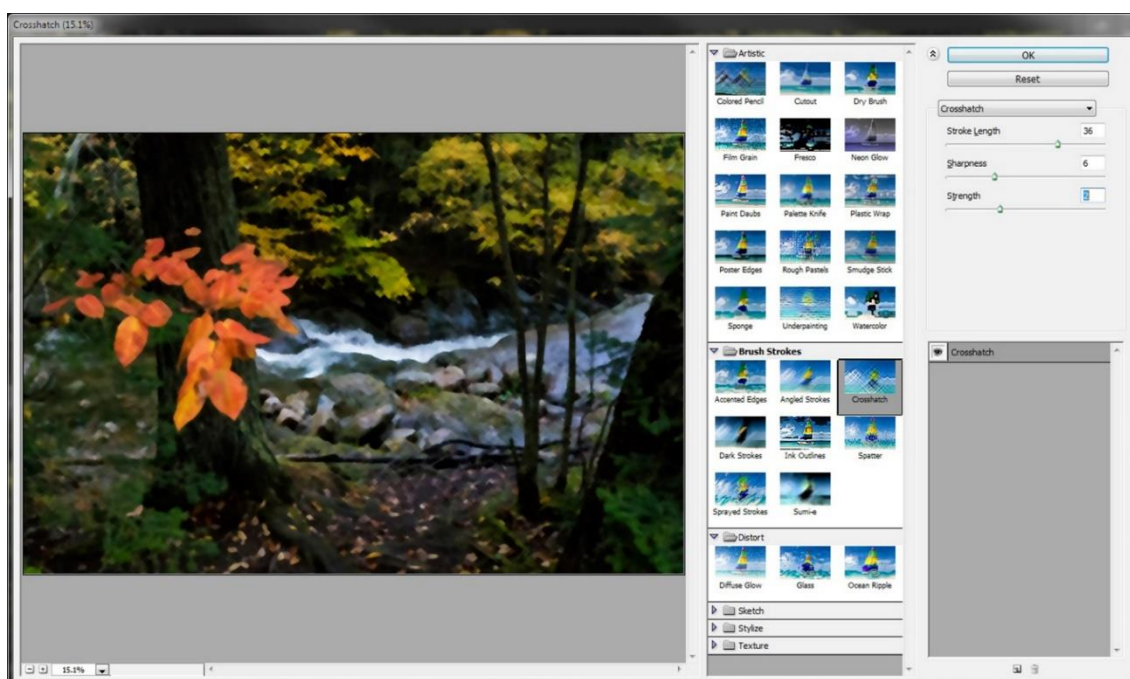


Figura 1.5 - Filtrele Photoshop

O imagine poate fi descrisă ca fiind picturală atunci când iluzia formei este creată prin utilizarea culorilor, liniilor, texturilor și a oricăror alte tehnici unice pentru arta picturii, mai degrabă decât o metodă liniară care implică desenul priceput. În termeni simpli, este folosit pentru a descrie un tablou care arată ca un tablou. Photoshop oferă posibilitatea creării acestor tipuri de imagini utilizând filtre diverse și instrumente prestabilite. Un exemplu de astfel de imagine este reprezentat în figura 1.6 Photoshop efectul Painterly.

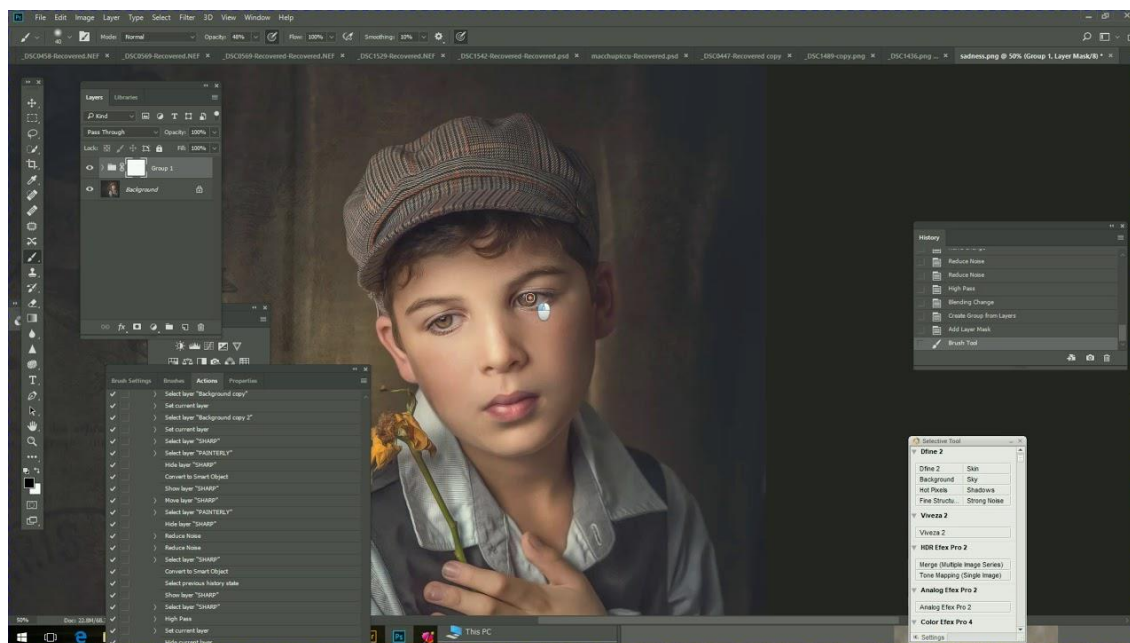


Figura 1.6 - Photoshop efectul Painterly

Ultimul exemplu a unuia din funcționalitățile Photoshop este retușarea imaginilor. Retușarea pozelor înseamnă eliminarea tuturor imperfecțiunilor dintr-o fotografie, care include de obicei corectarea culorii și a tonului, eliminarea petelor și a cercurilor de sub ochi, modificarea luminozității, contrastului și saturației. Acest funcțional este foarte popular deoarece el adesea permite crearea unei poze ideale având la baza o imagine simplă, poate nu chiar reușită. Modul în care arată acest funcțional și un exemplu al utilizării lui este reprezentat în figura 1.7 Photoshop Retușarea imaginii.

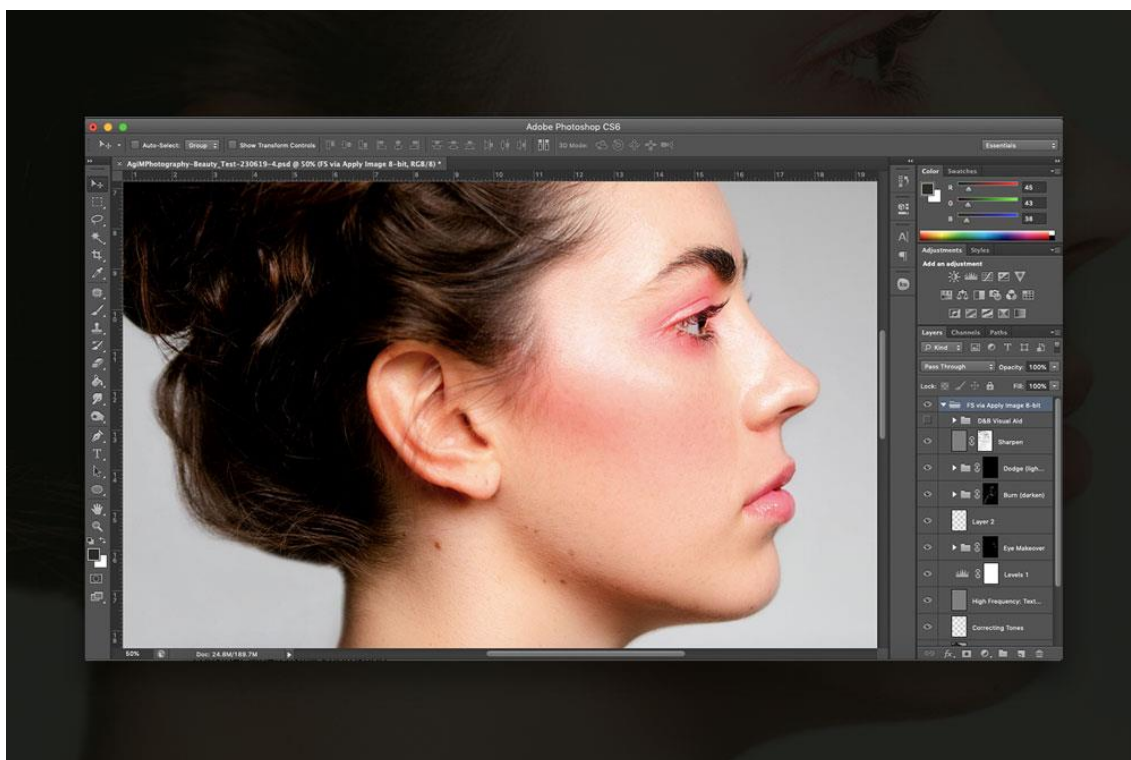


Figura 1.7 - Photoshop Retușarea imaginii.

### 1.3 SCOPUL, OBIECTIVELE ȘI CERINȚELE SISTEMULUI

#### Scopul:

Crearea unei aplicații ce permite procesarea și optimizarea imaginilor.

#### Obiectivele:

- Cercetarea domeniului „Tehnologii Multimedia”;
- Analiza soluțiilor existente de „Sisteme de optimizare a imaginilor”;
- Concluzii și Argumentarea propunerii de proiect „Sistem de optimizare a imaginilor”;
- Elaborarea concepției sistemului;
- Elaborarea Caietului de Sarcini;
- Argumentarea platformei și soft-ului de realizare;
- Cercetarea și selectarea instrumentelor pentru dezvoltarea platformei;
- Utilizarea unui algoritm de optimizare;
- Adăugarea funcționalului de compresare a unei imagini;
- Adăugarea funcționalului de redimensionare a unei imagini;
- Crearea funcționalului de adăugare a unui filtru;
- Crearea funcționalului de adăugare, editare și ștergere a unei imagini;

- Crearea componentei de vizualizarea a elementului ce urmează a fi editat;
- Testarea sistemului;
- Documentarea sistemului informațional;
- Planificarea proiectului si estimarea costului;
- Concluzii și recomandări;

Sistemul de editare si optimizare a imaginilor trebuie sa fie capabil sa îndeplinească funcțiunile de baza fără nici o întrerupere. În acțiunile de bază sunt prevăzute următoarele acțiuni:

- aplicarea unui filtru;
- salvarea imaginii editate;
- rotirea unei imagini;
- decuparea unui imagini;
- selectarea locației de salvare;
- adăugarea detaliilor.

## 2 MODELAREA ȘI PROIECTAREA SISTEMUL INFORMATIC

Această lucrare se concentrează pe proiectarea și dezvoltarea unei aplicații de editare a imaginilor utilizând metoda de proiectare Waterfall. Metoda Waterfall este una dintre cele mai vechi și mai răspândite metode de dezvoltare software, care implică o abordare secvențială a procesului de dezvoltare. Prin urmare, fiecare fază a procesului de dezvoltare trebuie să fie finalizată înainte de a se trece la faza următoare, iar schimbările care trebuie făcute după ce o fază a fost finalizată sunt dificil de implementat. Această metodă are avantajul de a fi structurată și ușor de urmat, iar planificarea și managementul proiectului sunt mai ușor de realizat. Dezavantajul este că poate fi dificil să se adapteze schimbărilor în timpul procesului de dezvoltare, ceea ce poate duce la probleme de implementare și la costuri suplimentare.

Pentru proiectarea arhitecturii aplicației de editare a imaginilor, s-a utilizat limbajul de proiectare Enterprise Architect, o unealtă puternică și versatilă pentru proiectarea și modelarea sistemelor complexe de software. Enterprise Architect a permis proiectarea modelelor UML. Acesta a fost ales datorită funcționalității sale puternice și flexibilității sale, oferind o varietate de caracteristici care au făcut posibilă proiectarea arhitecturii aplicației într-un mod precis și organizat.

În ceea ce privește mediul de dezvoltare, s-a optat pentru utilizarea IntelliJ IDE, un mediu integrat de dezvoltare pentru limbajul de programare Java, cu o gamă largă de funcții și caracteristici care facilitează dezvoltarea de software de înaltă calitate. IntelliJ IDE a fost ales datorită interfeței sale intuitive și a capacității sale de a ajuta dezvoltatorii să scrie cod eficient și să găsească rapid erori. De asemenea, IntelliJ IDEA oferă o gamă largă de funcții avansate, inclusiv completarea automată a codului, navigarea rapidă în cod, refactorizarea automată a codului, gestionarea de proiecte complexe și multe altele. Aceste caracteristici permit dezvoltatorilor să scrie cod mai rapid și să își concentreze eforturile asupra dezvoltării de funcționalități noi și inovatoare. IntelliJ IDEA suportă, de asemenea, o gamă largă de tehnologii și framework-uri Java, inclusiv Java EE, Spring, Hibernate, Struts, Groovy, Scala și multe altele. Acest lucru permite dezvoltatorilor să utilizeze cele mai recente tehnologii și să se concentreze pe dezvoltarea de software de înaltă calitate. În concluzie, IntelliJ IDEA este un mediu de dezvoltare integrat puternic și versatil pentru dezvoltatorii Java, care oferă o gamă largă de funcții și caracteristici avansate pentru dezvoltarea rapidă și eficientă a software-ului.

Aceste instrumente și metode de proiectare și dezvoltare au fost alese cu grijă pentru a asigura o abordare riguroasă și structurată a dezvoltării aplicației de editare a imaginilor, cu accent pe calitatea produsului final și eficiența procesului de dezvoltare. Proiectarea și dezvoltarea unei aplicații de editare a imaginilor este un proces complex, iar utilizarea unor instrumente și metode de proiectare adecvate poate contribui semnificativ la succesul final al proiectului.

## 2.1 DESCRIEREA COMPORTAMENTALĂ A SISTEMULUI

Imaginea generală asupra sistemului reprezintă o diagramă de nivel înalt care prezintă componentele principale ale sistemului și relațiile dintre acestea. Această diagramă poate fi utilizată pentru a ilustra arhitectura generală a sistemului și pentru a oferi o perspectivă de ansamblu asupra funcționalităților sale. Ea este realizată prin diagrama use case unde se descrie interacțiunea între sistem și utilizatorii săi, precum și interacțiunea între utilizatori (figura 2.1 Cazurile de utilizare pentru interacțiunea cu sistemul)

Modelarea vizuală a fluxurilor, reprezentată prin diagrama de activitate (Activity Diagram), arată secvența de acțiuni sau etape pe care utilizatorul le urmează pentru a realiza o anumită sarcină în cadrul sistemului. Această diagramă poate fi utilizată pentru a planifica interacțiunile utilizator-sistem și pentru a asigura că fluxul de lucru este logic și eficient. Diagrama de activități poate fi folosită pentru a descrie fluxurile de lucru sau fluxurile de date, ceea ce face posibilă dezvoltarea software-ului conform cerințelor de afaceri și necesităților utilizatorilor.(figura 2.2 Diagrama de activitate a salvării unui fișier)

Afișarea stărilor de tranzație este realizată prin diagramele de tip diagrama de stare (Statechart Diagram e), diagrama dată prezintă stările posibile ale unui obiect în cadrul sistemului și tranzițiile între aceste stări. Această diagramă poate fi utilizată pentru a modela comportamentul obiectelor în timp și pentru a identifica posibilele probleme sau conflicte. Este utilă pentru a înțelege și identifica secvențele de evenimente care declanșează tranzițiile între diferitele stări ale obiectului.(figura 2.3 Diagrame de stare a selectării operațiunii)

Descrierea scenariilor de utilizarea a aplicației se realizează prin Diagrama de Secvențe (Sequence Diagram), ea ilustrează interacțiunile dintre obiectele sistemului într-un anumit scenariu de utilizare. Această diagramă poate fi utilizată pentru a clarifica comunicarea și fluxul de date dintre obiectele sistemului.(figura 2.4 Diagrama de secvență a procesului de adăugare a unui fișier)

Diagrama de Colaborare (Collaboration Diagram) prezintă fluxurile de mesaje și legăturile dintre obiectele sistemului într-un anumit scenariu de utilizare. Această diagramă poate fi utilizată pentru a ilustra interacțiunile complexe dintre obiecte și pentru a identifica eventualele probleme de comunicare sau dependențe.(figura 2.5 Diagrama de colaborare a procesului de decupare)

## **2.1.1 IMAGINEA GENERALĂ ASUPRA SISTEMULUI**

Diagrama use case este o unealtă importantă în proiectarea de software, care permite identificarea și definirea necesităților și cerințelor utilizatorilor. Această diagramă descrie interacțiunea între sistem și utilizatorii săi, precum și interacțiunea între utilizatori. Prin utilizarea diagramelor de use case, se pot identifica diverse scenarii de utilizare a sistemului și se pot defini cerințele de bază ale sistemului. Diagrama use case ajută la identificarea scenariilor de utilizare ale sistemului și la definirea cerințelor funcționale ale acestuia. Aceste scenarii reprezintă modul în care utilizatorii vor folosi sistemul și acoperă toate funcționalitățile acestuia. Diagrama de use case poate fi utilizată și pentru a identifica și defini cerințele de performanță, de securitate sau de interfață grafică ale sistemului.

O diagramă use case este importantă deoarece permite dezvoltatorilor să înțeleagă mai bine nevoile și cerințele utilizatorilor și să creeze soluții care satisfac aceste nevoi. De asemenea, diagrama de use case poate ajuta la comunicarea mai eficientă între dezvoltatorii de software și clienții sau utilizatorii finali ai sistemului. Diagrama de use case poate fi, de asemenea, utilizată pe parcursul întregului ciclu de dezvoltare a software-ului, de la planificare și proiectare până la testare și livrare. Prin urmare, diagrama de use case este un instrument important și util pentru proiectarea și dezvoltarea de software de calitate. În figura 2.1 este arătată diagrama cazurilor de utilizare pentru interacțiunea utilizatorului cu sistemul.

Acțiunile care pot fi efectuate de către un utilizator în cazul curent sunt:

- adăugarea unei imagini;
- salvarea unei imagini;
- rotirea stânga;
- rotire dreapta;
- aplicare filtru;
- decupare;
- apropiere.



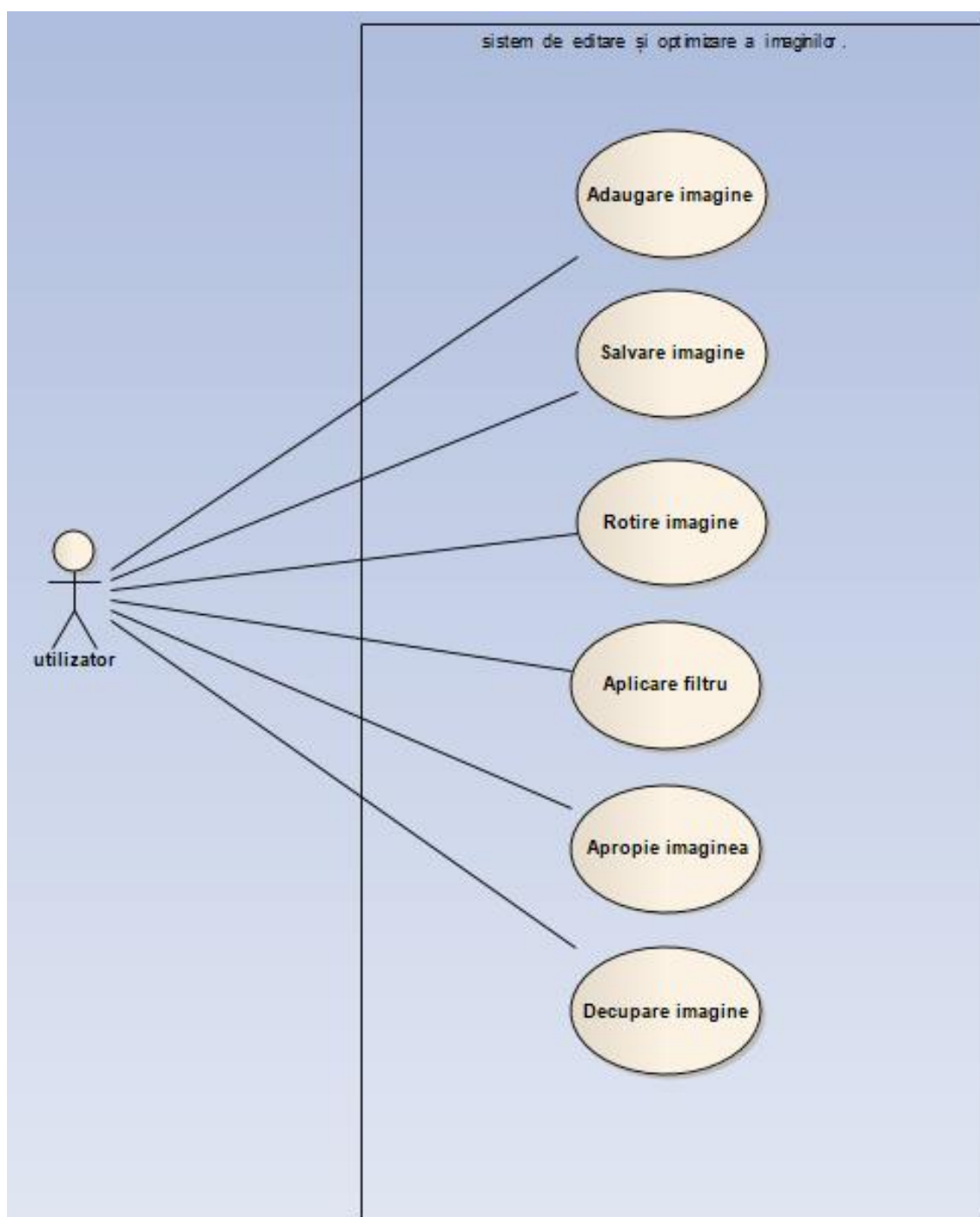


Figura 2.1 – Cazurile de utilizare pentru interacțiunea cu sistemul

## 2.1.2 MODELAREA VIZUALĂ A FLUXURILOR

Diagrama de activități ajută la definirea, descrierea și analizarea proceselor și a fluxurilor de lucru dintr-o aplicație. Aceasta este esențială în dezvoltarea software-ului, deoarece permite programatorilor să înțeleagă și să modeleze procesele complexe dintr-o aplicație și să le împartă în acțiuni mai mici și mai ușor de gestionat. Diagrama de activități poate fi folosită pentru a descrie fluxurile de lucru sau fluxurile de date, ceea ce face posibilă dezvoltarea software-ului conform

cerințelor de afaceri și necesităților utilizatorilor. Aceasta poate fi utilizată pentru a analiza fluxul de date în aplicație și pentru a identifica posibile probleme în designul aplicației. În figura 2.2 este reprezentată diagrama de activitate pentru procesul de salvare a unui fișier editat pe calculatorul personal. Utilizatorul indică locația și denumirea fișierului ce urmează a fi salvat.

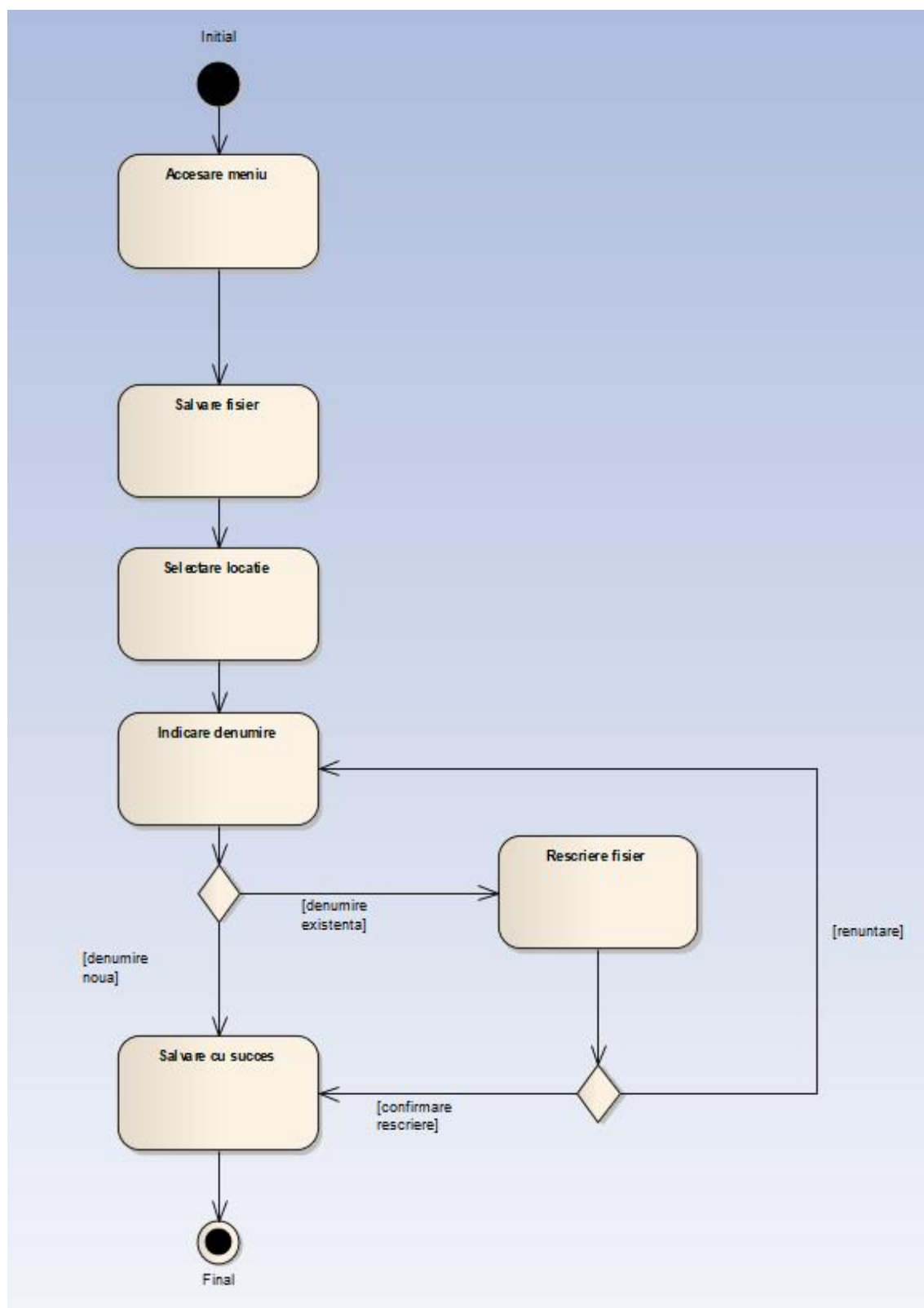


Figura 2.2 – Diagrama de activitate a salvării unui fișier

### 2.1.3 STĂRILE DE TRANZACȚIE A SISTEMULUI

Diagrama de stare poate fi utilizată în toate etapele de dezvoltare a software-ului, începând de la analiza cerințelor și planificarea sistemului, până la implementarea, testarea și mentenanța acestuia. Diagrama de stare este importantă în modelarea sistemelor software deoarece:

- Permite definirea și reprezentarea modului în care obiectele din sistem se comportă în diferite stări;
- Ajută la identificarea modului în care obiectele pot trece de la o stare la alta;
- Poate fi folosită pentru a modela comportamentul obiectelor și pentru a reprezenta interacțiunile între obiecte și sistemul înconjurător;
- Este utilă pentru a înțelege și identifica secvențele de evenimente care declanșează tranzițiile între diferitele stări ale obiectului;
- Poate ajuta la identificarea situațiilor neașteptate sau incorecte care pot apărea în sistem;
- Este utilă pentru dezvoltarea de software deoarece poate ajuta la identificarea și eliminarea problemelor de design sau a erorilor în timpul procesului de dezvoltare.

În figura 2.3 este reprezentată diagrama de stare a selectării unei operațiuni. După rularea aplicației are loc un state-machine pentru efectuarea operațiunii unde utilizatorul urmează să aleagă operațiunea și tipul, în cazul în care selectarea este eronată procesul de selectare a unei acțiuni se începe de la început.

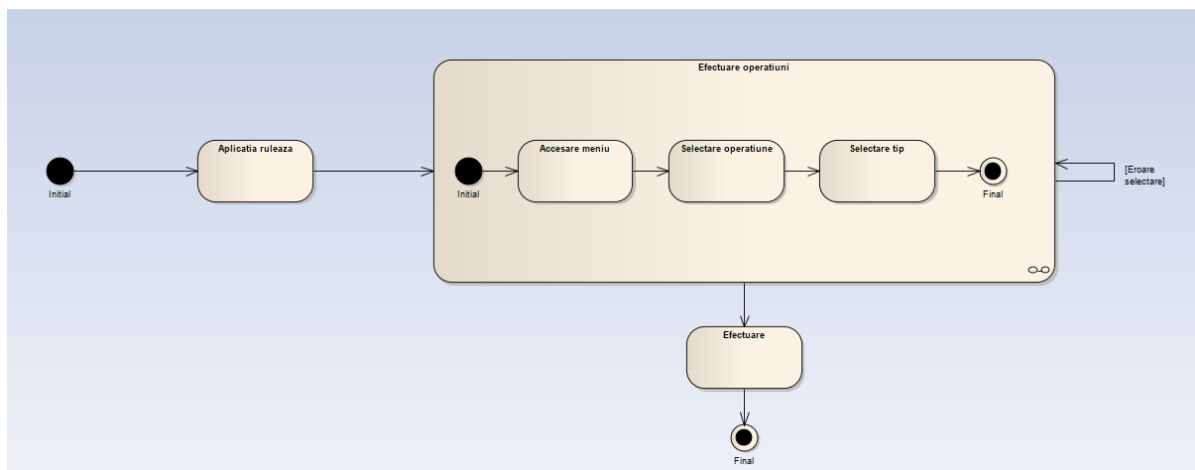


Figura 2.3 – Diagrame de stare a selectării operațiunii

## 2.1.4 DESCRIEREA SCENARIILOR DE UTILIZARE A APLICAȚIEI

Diagrama de secvențe prezintă interacțiunile între obiecte într-o secvență ordonată de mesaje. Diagrama de secvențe poate fi utilizată pentru a modela și analiza interacțiunile între obiecte și procesele sistemului, precum și pentru a identifica probleme de sincronizare sau de performanță în timpul execuției. De asemenea, poate fi folosită pentru a comunica interacțiunile complexe între diferite module și subsisteme ale unui sistem la dezvoltatori și la alte părți interesate. În figura 2.4 este reprezentată diagrama de secvență a procesului de adăugare a unui fișier. În aceasta diagrama este arătat cine deține focusul pe durata procesului de adăugare a unui fișier și cum are loc adăugarea la nivel intern.

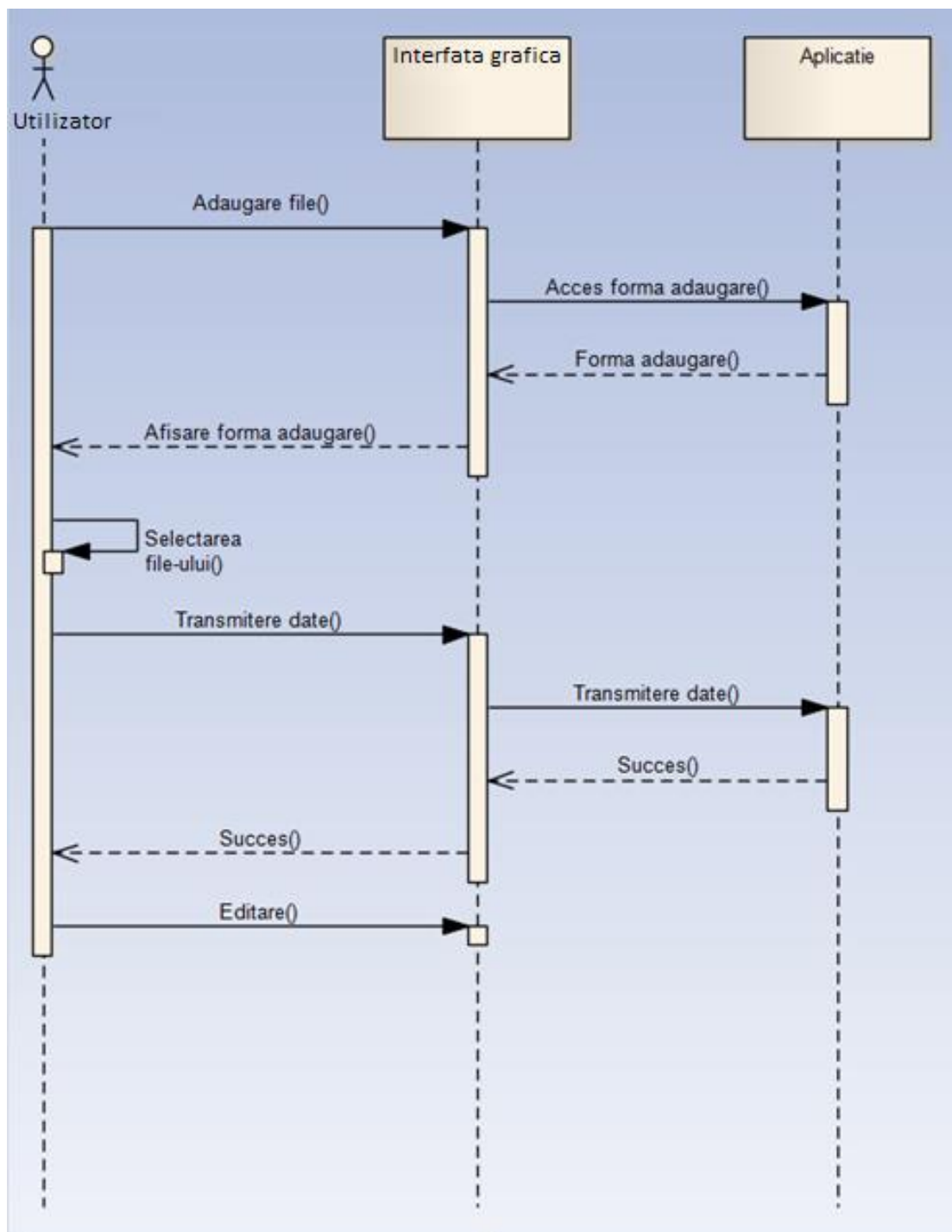


Figura 2.4 – Diagrama de secvență a procesului de adăugare a unui fișier

## 2.1.5 FLUXURILE DE MESAJE ȘI LEGĂTURILE DINTRE COMPONENTELE SISTEMULUI

Diagrama de colaborare este importantă în dezvoltarea de software din mai multe motive:

- ajută la înțelegerea interacțiunilor dintre obiecte: Diagrama de colaborare oferă o imagine vizuală asupra modului în care obiectele interacționează între ele în cadrul sistemului software. Prin urmare, este mai ușor de înțeles modul în care funcționează sistemul și cum obiectele comunică între ele.
- ajută la testarea și depanarea sistemului: Diagrama de colaborare poate fi utilizată pentru a testa și depana sistemul software. Analizând interacțiunile dintre obiecte, se pot identifica eventualele probleme și se pot face ajustări în consecință.
- documentare: Diagrama de colaborare poate fi utilizată pentru a documenta sistemul software. Prin urmare, aceasta poate ajuta la instruirea utilizatorilor și dezvoltatorilor noi, care trebuie să înțeleagă modul în care sistemul este construit și cum funcționează.
- identificarea problemelor: Diagrama de colaborare poate ajuta la identificarea problemelor în timpul dezvoltării software-ului. De exemplu, dacă un obiect nu comunică corect cu celelalte obiecte, diagrama poate fi utilizată pentru a identifica problema și a face ajustările necesare.
- optimizarea performanței: Analizând interacțiunile dintre obiecte, diagrama de colaborare poate fi utilizată pentru a identifica zonele care pot fi optimizate pentru a îmbunătăți performanța sistemului.

În figura 2.5 este arătată diagrama de colaborare a procesului de decupare. În această diagramă sunt indicate toate cererile care sunt efectuate de către utilizator către aplicație.

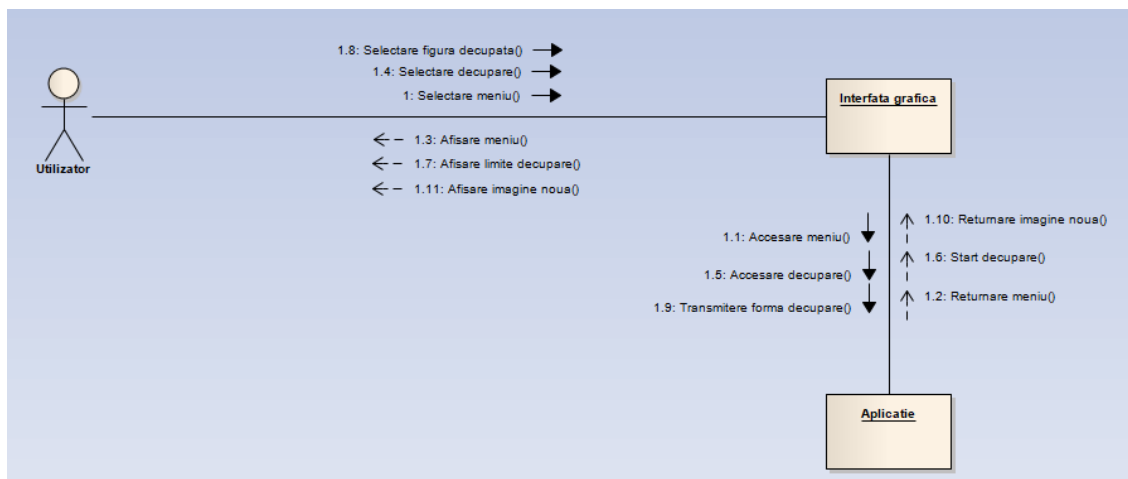


Figura 2.5 – Diagrama de colaborare a procesului de decupare

## 2.2 DESCRIEREA STRUCTURALĂ A SISTEMULUI

O aplicație de editare a imaginilor este o aplicație software care permite utilizatorilor să editeze și să manipuleze imagini digitale. Această aplicație este structurată în mai multe module, fiecare având un scop specific și funcționalitate distinctă. Un modul important în aplicația de editare a imaginilor este modulul de interfață utilizator. Acesta permite utilizatorilor să interacționeze cu aplicația prin intermediul unei interfețe grafice de utilizator (GUI). Acest modul poate fi divizat în sub-module, cum ar fi meniuri, bara de instrumente și panouri laterale, care permit utilizatorului să acceseze funcționalitățile specifice ale aplicației. Un alt modul important este modulul de manipulare a imaginilor. Acest modul conține funcții și algoritmi care permit utilizatorilor să modifice și să manipuleze imaginile, cum ar fi adăugarea de efecte, schimbarea culorilor sau ajustarea contrastului și luminozității. Acest modul este responsabil de aplicarea modificărilor asupra imaginii și de afișarea rezultatelor în interfața utilizator. Modulul de gestiune a fișierelor este un alt modul component în acest sistem. Acest modul permite utilizatorilor să importe și să exporte fișiere de imagine și să le salveze în diferite formate.

Structura statică a unui sistem reprezintă organizarea și relațiile dintre componente într-un moment dat, fără a lua în considerare comportamentul acestora. Aceasta poate fi descrisă prin intermediul diagramelor UML, cum ar fi diagrama de clasă. Diagrama de clasă reprezintă structura statică a sistemului prin intermediul claselor și a relațiilor dintre ele. Fiecare clasă descrie un set de attribute și metode, împreună cu relațiile pe care le are cu alte clase. Aceste relații pot fi de tipul moștenire, asociere, agregare sau compoziție. (figura 2.6 Diagrama de clasă a aplicației)

Relațiile de dependență între componentele unui sistem pot fi reprezentate într-o diagramă de dependență (Dependency Diagram). Această diagramă poate fi utilizată pentru a identifica și analiza dependențele între componentele sistemului, precum și pentru a evalua impactul schimbărilor într-o componentă asupra altor componente. Relația de dependență poate fi de tipul "utilizare" (usage), care indică faptul că o componentă utilizează o altă componentă pentru a îndeplini o anumită funcție sau sarcină. Aceasta poate fi o dependență directă, în cazul în care o componentă apelează funcții dintr-o altă componentă, sau o dependență indirectă, în cazul în care o componentă utilizează o altă componentă prin intermediul unei interfețe comune sau a unui mediator. (figura 2.7 Diagrama de componente a aplicației)

Modelarea echipamentelor mediului de implementare se realizează în cadrul diagramelor de implementare (Deploy Diagrams) și are ca scop reprezentarea fizică a componentelor software în diferitele noduri hardware ale sistemului. Aceste diagrame sunt utilizate pentru a modela și planifica arhitectura fizică a sistemului, precum și pentru a identifica și analiza dependențele și interacțiunile între componentele software și hardware. Echipamentele mediului de implementare includ calculatoare, servere alte dispozitive hardware utilizate pentru a susține funcționarea sistemului. Diagrama de implementare reprezintă aceste echipamente printr-un set de noduri, care pot fi conectate între ele prin intermediul unor legături de rețea. (figura 2.8 Diagrama de implementare a sistemului)

## 2.2.1 DESCRIEREA STRUCTURII STATICE A SISTEMULUI

Diagrama de clase este una dintre cele mai importante și utilizate diagrame UML în modelarea sistemelor software. Această diagramă reprezintă clasele din sistem, relațiile dintre acestea și atributele și metodele fiecărei clase.

Diagrama de clase este importantă în modelarea sistemelor software deoarece:

- permite vizualizarea structurii sistemului, prin reprezentarea claselor și a relațiilor dintre acestea;
- oferă o perspectivă asupra funcționalităților sistemului și asupra interacțiunii dintre acestea;
- permite identificarea claselor și a relațiilor dintre acestea, astfel încât dezvoltatorii să poată înțelege mai bine structura sistemului și să poată planifica mai bine dezvoltarea acestuia;
- permite identificarea moștenirii și a relațiilor de tipul "este un" și "are un" dintre clase, ceea ce poate fi util pentru dezvoltarea de software modular și pentru reutilizarea codului;
- permite identificarea potențialelor probleme de design sau de performanță în sistem, precum și identificarea nevoilor de optimizare a structurii sau a codului;
- poate fi utilizată ca bază pentru generarea automată a codului, reducând astfel timpul și efortul necesar pentru dezvoltarea de software.

În figura 2.6 este arătată diagrama de clasă a aplicației. În ea sunt indicate clasele principale, relațiile între ele și părțile componente a fiecărei clase.

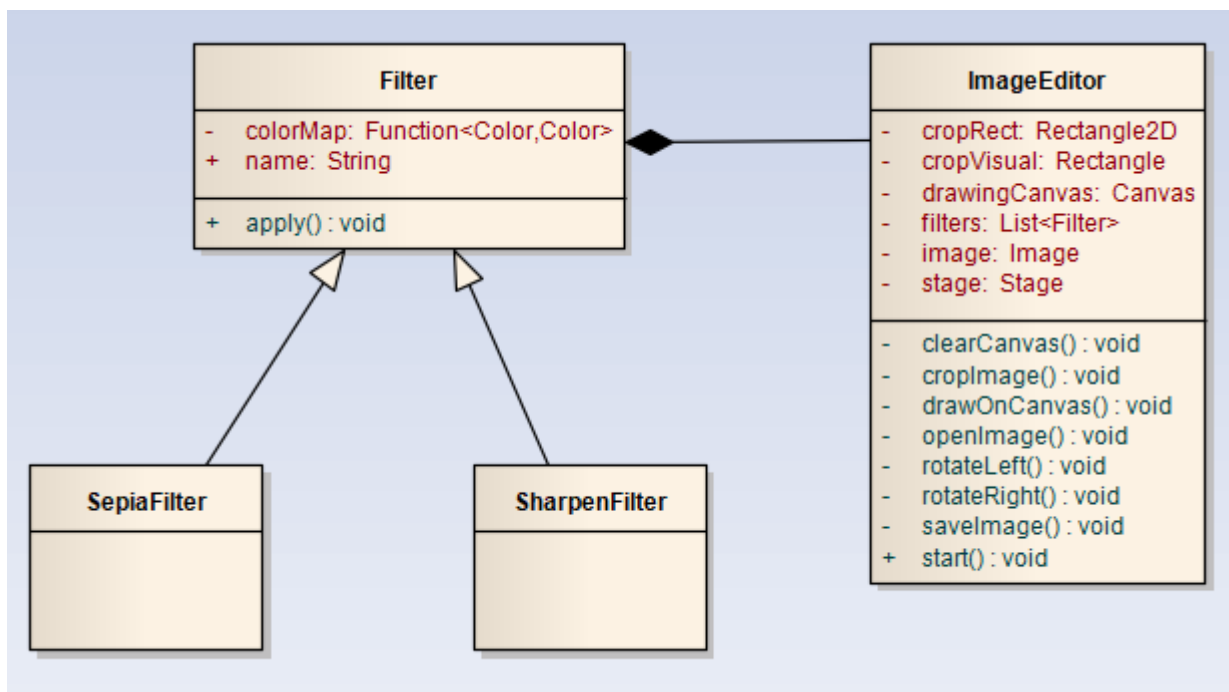


Figura 2.6 – Diagrama de clasă a aplicației

## 2.2.2 RELAȚIILE DE DEPENDENȚĂ ÎNTRE COMPONENTELE SISTEMULUI

Diagrama de componentă permite o viziune de ansamblu asupra componentelor sistemului și modul în care acestea interacționează între ele. Această diagramă poate fi folosită pentru a identifica relațiile dintre diferitele componente ale sistemului și poate ajuta la îmbunătățirea modularității și a structurii sistemului. Diagrama de componentă poate fi utilă pentru proiectarea arhitecturii sistemului software și poate ajuta la identificarea dependențelor între diferitele componente. Această diagramă poate fi folosită și pentru a reprezenta interacțiunea dintre componentele sistemului și cu alte sisteme, precum și pentru a identifica resursele necesare pentru implementarea și rularea sistemului. În plus, diagrama de componentă poate fi utilizată pentru a comunica cu cei interesați de proiectarea și dezvoltarea sistemului software, cum ar fi dezvoltatorii, designerii, managerii de proiect și clienții. Această diagramă poate ajuta la stabilirea unor obiective clare și la asigurarea unei înțelegeri comune a modului în care sistemul trebuie să funcționeze și să fie structurat. În figura 2.7 este reprezentată diagrama de componente a aplicației. În aceasta diagramă sunt indicate părțile componente a sistemului și modul în care ele interacționează între ele.

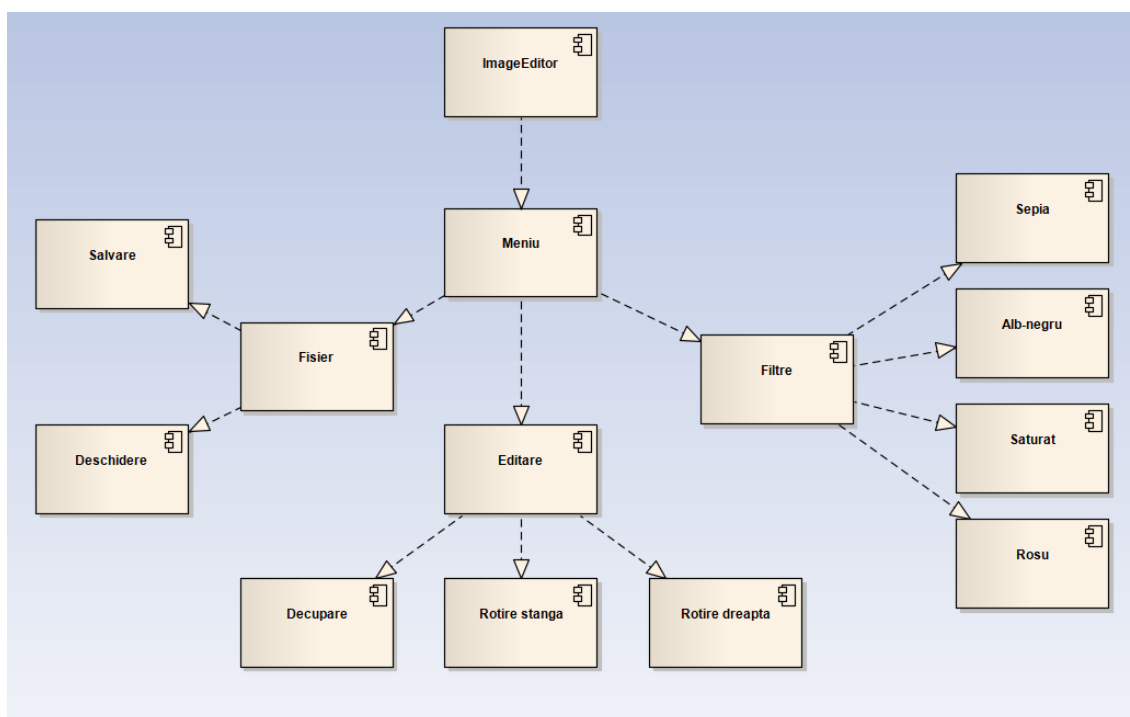


Figura 2.7 – Diagrama de componente a aplicației



## 2.2.3 MODELAREA ECHIPAMENTELOR MEDIULUI DE IMPLEMENTARE

Diagrama de implementare (Deploy Diagram) este importantă în modelarea sistemelor software deoarece:

- permite reprezentarea fizică a componentelor software în diferitele noduri hardware ale sistemului;
- permite planificarea arhitecturii fizice a sistemului;
- identifică și analizează dependențele și interacțiunile între componentele software și hardware;
- planifică și gestionează resursele hardware și software necesare sistemului pentru a asigura un nivel optim de performanță și disponibilitate;
- ajută la identificarea și eliminarea eventualelor probleme de compatibilitate între componentele hardware și software;
- facilitează procesul de mentenanță a sistemului prin reprezentarea fizică a componentelor software și hardware și a dependențelor dintre acestea;

Diagrama de implementare este utilă pentru dezvoltarea de software, deoarece permite o planificare și o gestiune eficientă a resurselor, o identificare și o rezolvare rapidă a problemelor și o îmbunătățire a performanței și disponibilității sistemului. În figura 2.8 este reprezentată diagrama de implementare a sistemului. În aceasta diagrama sunt reprezentate nodurile principale și relațiile dintre ele.

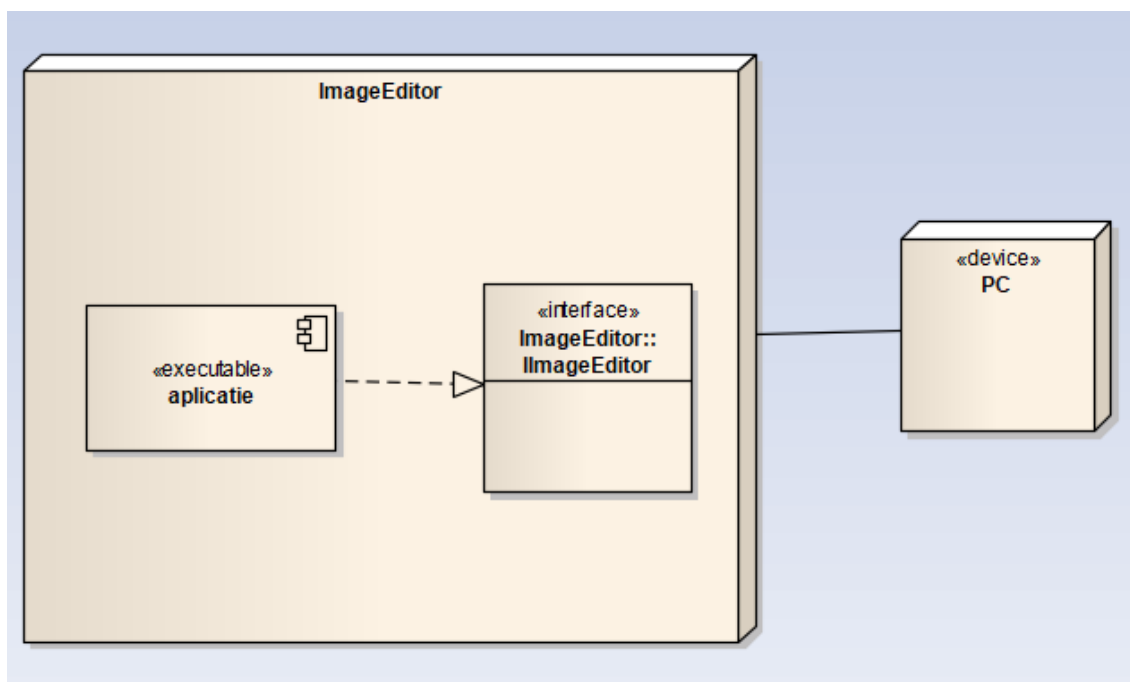


Figura 2.8 – Diagrama de implementare a sistemului

### 3 REALIZAREA SISTEMULUI

Pentru realizarea acestui proiect s-a utilizat limbajul Java, framework-ul JavaFX, Scene Builder, IntelliJ IDEA și șablonul de proiectare Memento.

Java este un limbaj de programare orientat pe obiecte, creat de Sun Microsystems (achiziționată de Oracle Corporation în 2010), care permite dezvoltatorilor să creeze aplicații pentru o varietate de dispozitive, de la computere personale până la dispozitive mobile și încorporate. Java a devenit popular datorită portabilității sale, adică codul sursă scris în Java poate fi compilat într-un format intermediar numit bytecode, care poate fi apoi interpretat și executat de o mașină virtuală Java (JVM) pe orice platformă care suportă JVM. Este un limbaj de programare flexibil și puternic, cu o sintaxă simplă și clară, care permite dezvoltatorilor să creeze aplicații sofisticate și scalabile. În plus, Java este un limbaj de programare de nivel înalt, cu o gamă largă de librării și framework-uri disponibile, care permite dezvoltatorilor să creeze aplicații complexe cu ușurință

JavaFX este un framework pentru crearea de aplicații desktop și mobile în Java. Acesta oferă o interfață grafică de utilizator (GUI) modernă și puternică, cu posibilități avansate de grafică 2D și 3D, animație și multimedia. JavaFX este un framework popular pentru crearea de aplicații desktop, jocuri și aplicații mobile.. Acesta oferă o serie de funcții pentru a simplifica dezvoltarea de interfețe grafice moderne, inclusiv suport pentru animații, efecte vizuale, stiluri și teme personalizabile. JavaFX a fost lansat pentru prima dată în 2008 și a fost inclus în distribuția standard JDK începând cu versiunea 8.

Scene Builder este o unealtă puternică și ușor de utilizat, care vine cu o varietate de opțiuni de personalizare a interfeței utilizator. Dezvoltatorii pot alege dintre o serie de stiluri predefinite, culori și teme, sau pot crea design-uri personalizate pentru a se potrivi cu brandul sau cerințele proiectului. Scene Builder poate fi integrat cu diverse medii de dezvoltare integrate (IDE), precum IntelliJ IDEA sau Eclipse, pentru a facilita fluxul de lucru și a spori eficiența programării. În plus, Scene Builder suportă limbajul FXML, care permite separarea codului interfeței utilizator de codul logic al aplicației, ceea ce face dezvoltarea și întreținerea mai ușoară și mai eficientă.

IntelliJ IDEA este un mediu integrat de dezvoltare (IDE) pentru programare în limbajul de programare Java. A fost creat de către compania JetBrains și este cunoscut pentru caracteristicile sale avansate de editare, analiză statică a codului, refactorizare și depanare. IntelliJ IDEA oferă suport pentru o varietate de tehnologii și framework-uri, inclusiv Spring, Hibernate, Maven, Gradle și multe altele. Este disponibil pentru platformele Windows, macOS și Linux.

Memento este un design pattern care permite salvarea stării unui obiect și restaurarea acestei stări la o dată ulterioară. Acesta este util în situațiile în care este necesară păstrarea unui istoric al stărilor obiectului, precum și pentru a face posibilă revenirea la o anumită stare anterioară a obiectului. Pattern-ul Memento este utilizat în JavaFX pentru a implementa operațiunile undo/redo, prin salvarea stărilor anterioare ale obiectelor și restaurarea acestora la cerere.

### 3.1 DESCRIEREA LA NIVEL DE COD PE MODULE

Structura generală a proiectului este arătată în figura 3.

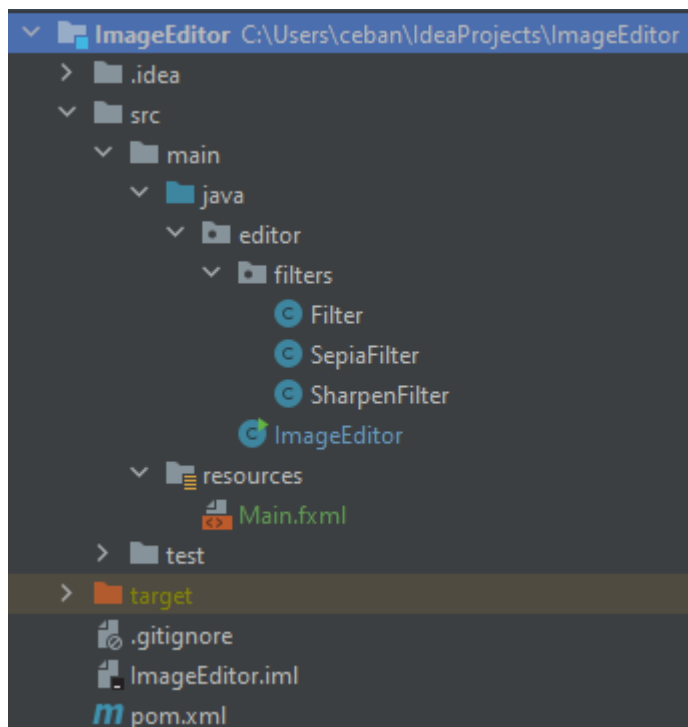


Figura 3 – Structura generală

În secvența de cod de mai jos este indicată funcția pentru deschiderea unui fișier de pe calculatorul personal. La efectuarea unui clic pe butonul din meniu se deschide un meniu aparte.

```
private void openImage() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Open Image");
    fileChooser.getExtensionFilters().addAll(
        new FileChooser.ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif"));
    File selectedFile = fileChooser.showOpenDialog(stage);
    if (selectedFile != null) {
        clearCanvas();
        Image image = new Image(selectedFile.toURI().toString());
        imageView.setImage(image);
    }
}
```

În următoarea secvență de cod este reprezentată funcția ce permite salvarea unui fișier deja editat într-un loc ales. În primul rând, este creat un obiect FileChooser și este configurat cu un titlu și filtre de extensie de fișiere. Apoi, metoda showSaveDialog() a obiectului FileChooser este apelată pentru a afișa o fereastră de dialog pentru selectarea fișierului în care se va salva imaginea. Dacă utilizatorul selectează un fișier, metoda creează un obiect WritableImage cu aceeași dimensiune ca și obiectul ImageView din

interfața utilizator. Obiectul `ImageView` este apoi utilizat pentru a lua o captură de ecran a stării curente a imaginii și a o stoca în obiectul `WritableImage`. În continuare, este creat un nou obiect `Canvas` cu aceeași dimensiune ca și imaginea modificată, iar obiectul `drawingCanvas` din interfața utilizator este utilizat pentru a lua o captură de ecran a stării sale curente. Captura de ecran este apoi desenată pe noul obiect `Canvas` utilizându-se `GraphicsContext2D`. În cele din urmă, imaginea modificată este actualizată pentru a include modificările făcute la `drawingCanvas`, iar metoda `ImageIO.write()` este utilizată pentru a scrie datele de imagine în fișierul selectat în format PNG.

În general, această metodă permite utilizatorilor să salveze starea curentă a imaginii și orice desene făcute pe ea într-un format de fișier ales de utilizator.

```
private void saveImage() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Save Image");
    fileChooser.getExtensionFilters().addAll(
        new FileChooser.ExtensionFilter("PNG Files", "*.png"),
        new FileChooser.ExtensionFilter("JPG Files", "*.jpg"),
        new FileChooser.ExtensionFilter("All Files", "*.*")
    );
    File selectedFile = fileChooser.showSaveDialog(stage);
    if (selectedFile != null) {
        WritableImage modifiedImage = new WritableImage((int) imageView.getFitWidth(), (int)
imageView.getFitHeight());
        imageView.snapshot(null, modifiedImage);

        SnapshotParameters parameters = new SnapshotParameters();
        parameters.setFill(Color.TRANSPARENT);
        Canvas tempCanvas = new Canvas(modifiedImage.getWidth(), modifiedImage.getHeight());
        tempCanvas.getGraphicsContext2D().drawImage(modifiedImage, 0, 0);
        tempCanvas.getGraphicsContext2D().drawImage(drawingCanvas.snapshot(parameters,
null), 0, 0);
        modifiedImage = tempCanvas.snapshot(parameters, null);
        try {
            ImageIO.write(SwingFXUtils.fromFXImage(modifiedImage, null), "png",
selectedFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

### 3.2 TESTAREA SISTEMULUI

Pentru testarea aplicației s-a utilizat testarea manuala si unit teste. Testarea manuala s-a efectuat in următorii pași:

- Testarea funcționalității: s-a testat funcționalitatea de bază a aplicației, cum ar fi încărcarea, editarea și salvarea imaginilor. A fost verificat că toate funcțiile și instrumentele disponibile în aplicație funcționează corect;
- Testarea interfeței cu utilizatorul: a fost testată interfața cu utilizatorul a aplicației pentru a se asigura că este intuitivă, ușor de utilizat și atrăgătoare din punct de vedere vizual. De asemenea s-a verificat dacă butoanele și alte elemente ale interfeței de utilizare funcționează conform așteptărilor;
- Testare de compatibilitate: a fost efectuată testarea aplicației pe diferite dispozitive și sisteme de operare pentru a se asigura că funcționează conform așteptărilor în diferite medii;
- Testarea performanței: s-au efectuat teste asupra aplicației pentru a identifica probleme de performanță încărcând imagini mari și verificând cât timp durează aplicația să le proceseze. Testarea aplicației cu diferite formate de fișiere pentru a se asigura că gestionează toate formatele în mod eficient;
- Testare de securitate: testarea aplicației pentru probleme de securitate încărcând imagini cu cod rău intenționat;
- Testare de utilizare: aceasta etapa prevede testarea aplicației din perspectiva utilizatorului pentru a se asigura că este intuitivă și ușor de utilizat. Aceasta implică testarea elementelor UI, a fluxului aplicației și a experienței generale a utilizatorului;
- Testarea gestionării erorilor: testarea gestionării erorilor aplicației, furnizând intenționat intrări incorecte și verificând modul în care aplicația răspunde.

Pentru testarea bazată pe unit teste s-au creat teste care verifica funcționalul. De exemplu următorul test creează o instanță a clasei `ImageView`, se setează o rotație inițială de 45 de grade, se creează o instanță a clasei `ImageEditor` cu instanța `ImageView` ca parametru, se apelează metoda `rotateLeft()`, și apoi se verifică dacă rotația `ImageView` a fost actualizată corect. Metoda `assertEquals` verifică dacă valorile așteptate și cele obținute pentru rotație sunt egale, cu o toleranță de 0,001 (pentru a lua în considerare posibile erori de rotunjire în virgulă mobilă).

```
@Test
void testRotateLeft() {
    ImageView imageView = new ImageView();
    double initialRotation = 45;
    imageView.setRotate(initialRotation);
```

```

ImageEditor editor = new ImageEditor(imageView);
editor.rotateLeft();

double expectedRotation = initialRotation - 90;
double actualRotation = imageView.getRotate();
assertEquals(expectedRotation, actualRotation, 0.001);
}

```

In urma rulării testului s-a obținut rezultatul arătat in figura 3.1

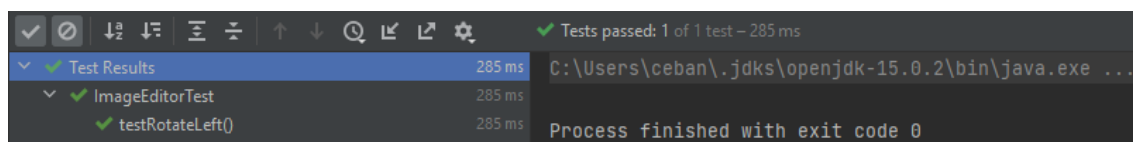


Figura 3.1 – Rezultatul executării testului

## CONCLUZII

În aceasta lucrare am studiat și realizat o aplicație ce are ca funcție principală editarea și optimizarea unei imagini. Toată munca efectuată a fost repartizată pe capitole, fiecare capitol având ca scop descrierea detaliată a unei părți distincte din aceasta lucrare. În total au fost realizate cinci capitole, ele la rândul său având subcapitole ce permit explicarea mai detaliată a ceea ce se efectuează în capitol.

Primul capitol realizat a fost „Analiza domeniul de studiu”, în acest capitol s-a studiat importanta temei, s-au cercetat sistemele similare cu proiectul realizat din mai multe aspecte și s-au identificat obiectivele principale, cerințele și scopul sistemului. În urma analizei s-a ajuns la concluzia că tema data este importantă din motiv că editarea unei imagini reprezintă o acțiune cotidiană pentru o persoană obișnuită. Sistemele similare oferă un funcțional larg de posibilități însă majoritatea din aceste funcționale sunt contra plată sau bazate pe abonamente. Ca scop principal a fost identificat faptul că o astfel de aplicație trebuie să dețină un funcțional de bază care ar include adăugare unei imagini, adăugarea unui filtru, decuparea și salvarea imaginii finale.

Al doilea capitol realizat a fost „Modelarea și proiectarea sistemului informatic”. Acest capitol s-a bazat pe descrierea comportamentală și structurală a sistemului realizat. Descrierea comportamentală presupune realizarea imaginii generale, modelarea vizuală a fluxurilor, stările de tranzație și scenariile de utilizare a aplicației. Imaginea generală la rândul ei reprezintă diagrame de tip use case unde sunt descrise interacțiunile între utilizator și sistem. Pentru modelarea vizuală a fluxurilor s-a realizat o diagramă de activitate unde este descris un proces din cadrul aplicației. Stările de tranzație reprezintă modul în care obiectele din sistem se comportă în diferite stări, pentru aceasta s-a realizat diagrama de stare unde au fost indicate stările în care trece sistemul. Ultima parte a descrierii comportamentale a fost identificarea scenariilor de utilizare a aplicației. Scenariile de utilizare ajută la modelarea și analiza interacțiunilor între obiecte și procesele sistemului. Descrierea structurală a fost efectuată prin realizarea descrierii structurii statice, reprezentarea relațiilor între componentele sistemului și modelarea echipamentelor mediului de implementare. Descrierea structurii statice prevede realizarea diagramelor de clasă, pentru aceasta a fost realizată diagramă de clasă a aplicației unde au fost indicate clasele principale, relațiile între ele și părțile componente. Relațiile de dependență au fost arătate printr-o diagramă de componente unde au fost indicate părțile componente a sistemului. Ultima etapă al acestui capitol a fost modelarea echipamentelor mediului de implementare, prin aceasta se are în vedere realizarea diagramei de implementare unde au fost indicate nodurile principale și relațiile dintre ele.

Capitolul trei a fost unul mai mult practic, acesta fiind dedicat realizării sistemului și testării lui. În acest capitol am descris instrumentele și tehnologiile utilizate pentru realizarea funcționalului, după care a fost descris cum funcționează codul, pentru aceasta au fost explicate secvențe de cod care execută o anumită acțiune. Descrierea codului a fost realizată pe module, din fiecare secțiune fiind selectată câte o funcție principală. De asemenea în acest capitol a fost explicată și testarea sistemului, mai exact a fost descris

modul în care a avut loc testarea. Testarea a fost efectuată în două etape, manuală și unit teste. Testarea manuală a prevăzut selectarea a mai multor aspecte și testarea lor din perspectiva unui utilizator.



## BIBLIOGRAFIE

- [1] Adriana Bogdan, ISTORIA PRIN APLICAȚII MULTIMEDIA; [Resursa electronica] – Regim de acces: <http://www.historica-cluj.ro/anuare/AnuarHistorica2014/24.pdf>
- [2] Grafică digitală; [Resursa electronica] – Regim de acces: [https://ro.wikipedia.org/wiki/Grafic%C4%83\\_digital%C4%83](https://ro.wikipedia.org/wiki/Grafic%C4%83_digital%C4%83)
- [3] An introduction to raster images; [Resursa electronica] – Regim de acces: <https://shorthand.com/the-craft/raster-images/index.html>
- [4] What is image compression and how does it work; [Resursa electronica] – Regim de acces: <https://www.techtarget.com/whatis/definition/image-compression>
- [5] Desktop Applications Vs. Web Applications; [Resursa electronica] – Regim de acces: [https://www.streetdirectory.com/travel\\_guide/114448/programming/desktop\\_applications\\_vs\\_web\\_applications.html](https://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html)
- [6] Pros and Cons of Desktop Apps; [Resursa electronica] – Regim de acces: <https://outsourcenz.com/pros-and-cons-of-web-apps-and-desktop-apps/>
- [7] Best Photo Editing Software; [Resursa electronica] – Regim de acces: <https://www.softwaretestinghelp.com/best-free-photo-editing-software/>
- [8] History of Photoshop; [Resursa electronica] – Regim de acces: [www.designbyfire.com/pdfs/history\\_of\\_photoshop.pdf](http://www.designbyfire.com/pdfs/history_of_photoshop.pdf)

**ANEXA A**  
**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**  
**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**CAIETUL**  
**STAGIULUI DE PRACTICĂ**  
**PENTRU STUDENȚII STUDIILOR SUPERIOARE DE LICENȚĂ – CICLULUI I**

**Stagiul de practică** \_\_\_\_\_ *de licență*  
tipul stagiului de practică

**A studentului (ei)** \_\_\_\_\_ *Ceban Vitalie*  
numele, prenumele

**Facultatea** \_\_\_\_\_ *Calculatoare, Informatică și Microelectronică*

**Programul de studii** \_\_\_\_\_ *Tehnologia informației*

**Ciclul** \_\_\_\_\_ *I* **Anul de studii** \_\_\_\_\_ *IV* **Grupa** \_\_\_\_\_ *TI-194*

**Locul stagiului de practică** \_\_\_\_\_ *Universitatea Tehnică a Moldovei*  
denumirea unității economice

**Conducătorul stagiului de practică**  
**de la UTM** \_\_\_\_\_ *asist.univ. Cojocaru Svetlana*  
funcția, numele, prenumele

**Conducătorul stagiului de practică**  
**de la unitatea economică** \_\_\_\_\_  
funcția, numele, prenumele

**Chișinău, 2023**

## I. CAIETUL DE SARCINI

Nr. crt.	Tematica lucrărilor preconizate	Termene planificate	
		început/ sfârșit	numărul de zile
1	<i>Analiza domeniului de studii și Modelarea și proiectarea sistemul informatic</i>	30.01.2023- 03.02.2023	5 zile
2	<i>Realizarea sistemului</i>	06.02.2023- 10.02.2023	5 zile
3	<i>Realizarea sistemului</i>	20.02.2023- 24.02.2023	5 zile
4	<i>Realizarea raportului și prezentării</i>	27.02.2023- 03.03.2023	5 zile

## II. SARCINA INDIVIDUALĂ (SE INDICĂ TEMA PROIECTULUI DE LICENȚĂ)

Tema: Proiectarea si dezvoltarea unui sistem de editare si optimizare a imaginilor

Nr. crt.	Conținutul concis al sarcinilor individuale	Mențiuni cu privire la realizare
1	<i>Analiza domeniului de studii</i>	<ul style="list-style-type: none"> <li>• Se descrie domeniul din care face parte proiectul creat ( tehnologii informaționale). Ce tip de aplicație ( web, mobile...etc). În ce domeniu se va aplica proiectul (învățământ, sfera serviciilor.....etc).</li> <li>• Se va argumenta importanța creării unui astfel de proiect.</li> <li>• Se va face o descriere a cel puțin 3 sisteme deja existente ( sau sisteme care au ceva comun din punct de vedere funcțional cu ce se va face în proiect). Se va face o comparare a sistemelor descrise.</li> <li>• În baza comparării se va scri scopul și obiectivele, cerințele sistemului. Determinarea cerințelor funcționale, nefuncționale. Determinarea funcționalului sistemului și subsistemelor. Se va descrie cât mai detaliat caietul de sarcini a proiectului.</li> </ul>
2	<i>Modelarea și proiectarea sistemul informatic</i>	<ul style="list-style-type: none"> <li>• Descrierea comportamentală a sistemului (imaginea generală asupra sistemului; modelarea vizuală a fluxurilor; stările de tranzație a sistemului (Statechart Diagram); descrierea scenariilor de utilizare a aplicației (Sequence Diagram); fluxurile de mesaje și legăturile dintre componentele sistemului (Collaboration Diagram).</li> <li>• Descrierea structurală a sistemului (descrierea structurii statice a sistemului; relațiile de dependență între componentele; modelarea echipamentelor mediului de implementare.</li> </ul>
3	<i>Realizarea sistemului</i>	<ul style="list-style-type: none"> <li>• Se face o introducere în ce s-a folosit pentru realizarea sarcinilor. Se descrie pe scurt tehnologiile folosite, limbaje de programare utilizate, instrumente, algoritmi, șabloane...etc , totul ce s-a folosit la realizarea proiectului.</li> <li>• Se descrie la nivel de cod realizarea funcționalităților</li> <li>• Se descrie testarea sistemului</li> </ul>
4	<i>Realizarea raportului</i>	<ul style="list-style-type: none"> <li>• De structurat după sarcini individuale</li> <li>• De redactat confor standardelor de redactare</li> </ul>
5	<i>Realizarea prezentării</i>	<ul style="list-style-type: none"> <li>• Slide cu denumirea temei</li> <li>• Slide cu Analiza domeniului</li> <li>• Slide cu Scopul și Obiectivele proiectului</li> <li>• Slide cu Imaginea generală a sistemului</li> <li>• Slide cu Descrierea structurii statice a sistemului</li> <li>• Slide cu Instrumente, tehnologii utilizate pentru realizarea sistemului</li> <li>• Slide cu Video- prezentarea aplicației</li> </ul>

Conducătorul \_\_\_\_\_ / Cojocaru Svetlana /  
semnătura numele și prenumele

## FIȘA DE ACTIVITATE

### Săptămâna I

---

Sarcini planificate:

- Argumentarea importanței unui astfel de sistem
- Descrierea domeniului aplicației, tipul de aplicație și domeniul în care este utilizat
- Studiarea sistemelor existente asemănătoare cu sistemul planificat.
- Identificarea scopului și obiectivelor, funcționalului unui asemenea sistem

#### Activități desfășurate, observații personale

##### **Importanța unui astfel de sistem**

- Sistemele ce utilizează imagini în majoritatea cazurilor stochează aceste imagini în baze de date iar din cauza că numărul acestor fișiere este gigantic spațiul necesar este de asemenea unul foarte mare. Spațiul pe servere costă bani iar reducerea spațiului necesar ar scădea suma totală pe care o necesită un sistem ce vizează stocarea datelor. Aplicația dată are ca scop reducerea dimensiunilor imaginilor pentru economisirea spațiului.

##### **Domeniul aplicației**

- Aplicația face parte din domeniul tehnologiilor multimedia. Aplicația este de tip desktop având un GUI ce permite activitățile de bază pentru o aplicație ce ar permite editarea unei imagini sau optimizarea ei. Astfel de aplicații sunt utilizate pentru lucrul cu imaginile.

##### **- Sisteme asemănătoare cu sistemul dat sunt:**

- Kraken.io;
- PIXLR;
- Photoshop;

Kraken.io este un instrument de optimizare a imaginilor. Cu Kraken.io, puteți optimiza cantități mari de fișiere JPEG, PNG și GIF animate. Spre deosebire de alt optimizator de imagine, Kraken.io optimizează fișierele pentru cea mai mică dimensiune. Adică, cu Kraken.io, se obțin întotdeauna cele mai mici versiuni ale fișierelor introduse.

Pixlr este unul dintre cele mai populare programe de editare a imaginilor disponibile astăzi pe piață. Cele mai recente ediții de Pixlr oferă o experiență de editare de nivel înalt. De la conversia imaginii la multiple alte opțiuni de editare, Pixlr are o soluție pentru orice.

Adobe Photoshop este un alt software folosit pentru editarea imaginilor digitale pe calculator, program produs și distribuit de compania americană Adobe Systems și care se adresează în special profesioniștilor domeniului.

Adobe Photoshop, așa cum este cunoscut astăzi, este vârful de lance al gamei de produse software pentru editare de imagini digitale, fotografii, grafică pentru tipar, video și Web de pe piață. Photoshop este un program cu o interfață intuitivă și care permite o multitudine extraordinară de modificări necesare în mod curent profesioniștilor și nu numai: editări de luminozitate și contrast, culoare, focalizare, aplicare de efecte pe imagine sau pe zone (selecții), retușare de imagini degradate, număr arbitrar de canale de culoare, suport de canale de culoare pe 8, 16 sau 32 biți, efecte third-party etc.

**-Scopul si obiectivele:**

Principalul scop este crearea unei aplicații ce permite procesarea si optimizarea imaginilor.

**Obiectivele:**

- Cercetarea domeniului „Tehnologii Multimedia”;
- Analiza soluțiilor existente de „Sisteme de editare si optimizare a imaginilor”;
- Concluzii si Argumentarea propunerii de proiect „Sistem de editare optimizare a imaginilor”;
- Elaborarea concepției sistemului;
- Elaborarea Caietului de Sarcini;
- Argumentarea platformei și soft-ului de realizare;
- Cercetarea si selectarea instrumentelor pentru dezvoltarea platformei;
- Utilizarea unui algoritm de optimizare;
- Adăugarea funcționalului de compresare a unei imagini;
- Adăugarea funcționalului de redimensionare a unei imagini;
- Crearea funcționalului de adăugare a unui filtru;
- Crearea funcționalului de adăugare, editare si ștergere a unei imagini;
- Crearea componentei de vizualizarea a elementului ce urmează a fi editat;
- Testarea sistemului;
- Documentarea sistemului informațional;
- Planificarea proiectului si estimarea costului.

## FIȘA DE ACTIVITATE

### Săptămâna II

Sarcini planificate:

- Descrierea stărilor de tranzacție(Statechart Diagram)
- Descrierea scenariilor de utilizare a aplicației (Sequence Diagram)
- Descrierea fluxurilor de mesaje și legăturile dintre componentele sistemului (Collaboration Diagram)
- Descrierea structurală a sistemului (descrierea structurii statice a sistemului)
- Descrierea limbajului utilizat
- Descrierea bibliotecilor utilizate
- Descrierea la nivel de cod a funcționalităților

### Activități desfășurate, observații personale

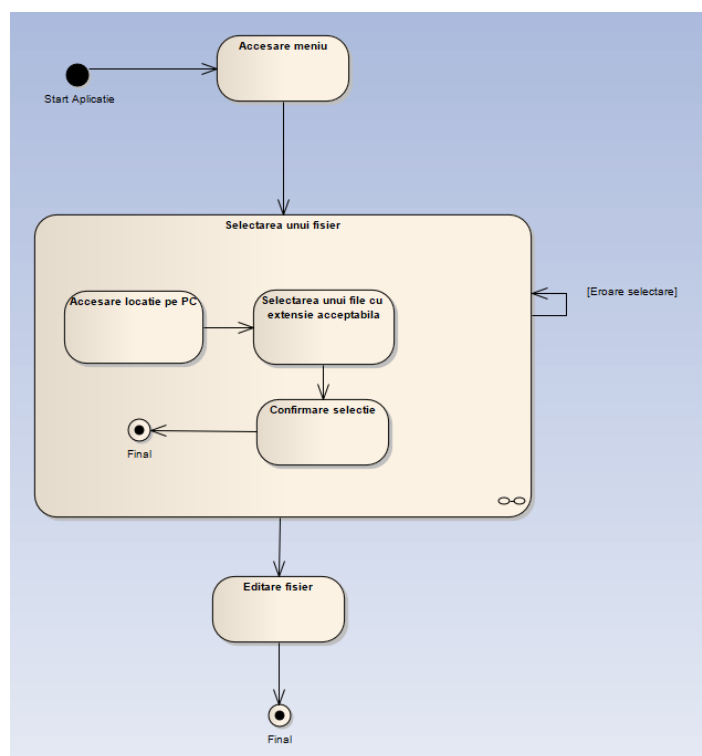


Figura 1 Diagrama de stare

În figura 1 este reprezentată diagrama de stare a sistemului pentru procesul de selectarea a unui fișier cu extensia permisă ce urmează a fi editat. După accesarea meniului are loc un state machine pentru selectarea fișierului cu extensia permisă pentru editare, în cazul în care selectarea este eronată procesul se începe de la început.

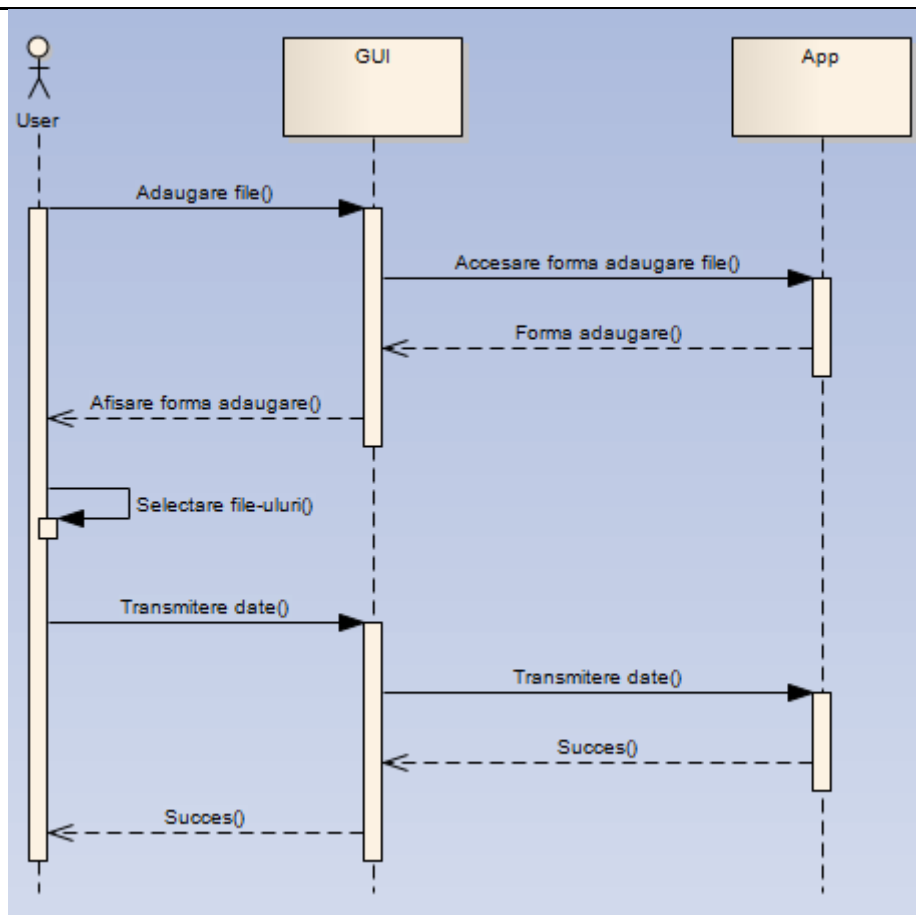


Figura 2 Diagrama de secventa

Un scenariu de utilizare a aplicației ar fi adăugarea unui file in lista de file-uri ce urmează a fi editate.

În figura 2 observăm diagrama de secvență pentru procesul de adăugare a unui file in lista de redare. In sistema data putem observa ca mesajele sunt de tip *call* si *return*. Mesajul de tip *call* este utilizat pentru a invoca chemarea unei operațiuni(adăugare file, accesare forma adăugare file).

Mesajul de tip *return* returnează valoarea operației executate, in cazul dat daca adăugarea a fost cu succes.

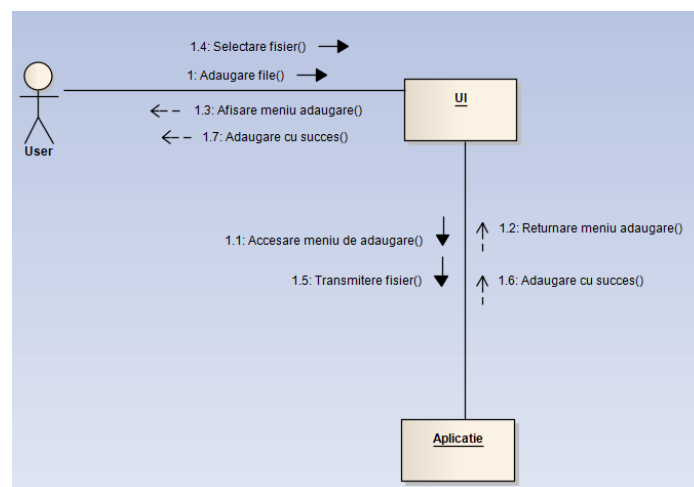


Figura 3 Diagrama de colaborare



De asemenea a fost realizata și diagrama de colaborare de nivel de exemple pentru acest sistem.

În Figura 3. Diagrama procesului de adăugare file(nivel de exemplu) este reprezentata diagrama procesului de adăugare a unui file nou la nivel de exemplu. În aceasta diagrama User-ul face request la forma adăugare file, primește forma de adăugare după care selectează file-ul dorit. După care file-ul dorit este transmis la aplicație unde urmează a fi prelucrat. În cazul în care datele sunt corecte el primește răspuns pozitiv.

### **Utilizarea limbajului Java**

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către James Gosling la Sun Microsystems (acum filială Oracle) la începutul anilor '90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile, spre exemplu telefon, agenda electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intranet-urilor.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java (engleză Java Virtual Machine, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (engleză byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

### **Biblioteca JavaFX**

JavaFX este o platformă bazată pe Java pentru construirea de aplicații GUI bogate . Poate fi folosit atât pentru a crea aplicații desktop care rulează direct din sistemele de operare, cât și pentru aplicații de Internet ( RIA ) care rulează în browsere , cât și pentru aplicații pe dispozitive mobile. JavaFX este destinat să înlocuiască biblioteca Swing utilizată anterior . Platforma JavaFX concurează cu Microsoft Silverlight , Adobe Flash și sisteme similare. Din Java 11, acesta nu mai este inclus în Java SE și nu este dezvoltat de Oracle (suportat ca modul separat de Gluon). Oracle va suporta JavaFX ca parte a Java JDK 8 până în martie 2025.

### **Descrierea la nivel de cod**

```

private void openImage() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Open Image");
    fileChooser.getExtensionFilters().addAll(
        new FileChooser.ExtensionFilter( s: "Image Files", ...strings: "*.png", "*.jpg", "*.gif"),
        new FileChooser.ExtensionFilter( s: "All Files", ...strings: "*.*")
    );
    File selectedFile = fileChooser.showOpenDialog(stage);
    if (selectedFile != null) {
        Image image = new Image(selectedFile.toURI().toString());
        imageView.setImage(image);
    }
}

```

Figura 4 Funcția de adăugare a unei imagini

În imaginea de mai sus este arătată funcția de adăugare a unui fișier ce urmează a fi editat. Fișierul trebuie să destina o extensie compatibilă în caz contrar nimic nu va fi adăugat.

```

private void rotateLeft() {

    imageView.setRotate(imageView.getRotate() - 90);
}

private void rotateRight() {

    imageView.setRotate(imageView.getRotate() + 90);
}

```

Figura 5 Funcțiile de rotire a unei imagini

În figura 5 sunt reprezentate funcțiile ce permit rotirea unei imagini la dreapta sau stânga.

## FIȘA DE ACTIVITATE

### Săptămâna III

Sarcini planificate:

- Adăugarea funcționalităților noi
- Descrierea la nivel de cod a funcționalităților
- Testarea manuala a funcționalului

### Activități desfășurate, observații personale

#### Descrierea la nivel de cod

```
private void drawOnCanvas(MouseEvent mouseEvent) {
    drawingCanvas.setOnMousePressed(event -> {
        graphicsContext.beginPath();
        graphicsContext.moveTo(event.getX(), event.getY());
        graphicsContext.stroke();
    });

    drawingCanvas.setOnMouseDragged(event -> {
        graphicsContext.lineTo(event.getX(), event.getY());
        graphicsContext.stroke();
    });

    drawingCanvas.setOnMouseReleased(event -> {
        graphicsContext.closePath();
    });

    // Clear
    graphicsContext.clearRect(0, 0, drawingCanvas.getWidth(), drawingCanvas.getHeight());
}
```

Figura 1 Funcția de adăugare a unei ilustrări

În Figura 1 este arătată funcția ce permite adăugarea unor detalii noi (linii) pe o imagine existentă. Se alege un punct de start la apăsarea pe un punct de pe ecran și până când nu este eliberat butonul stâng de pe mouse este desenată o linie.

```
private void saveImage() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Save Image");
    fileChooser.getExtensionFilters().addAll(
        new FileChooser.ExtensionFilter("PNG Files", "*.png"),
        new FileChooser.ExtensionFilter("All Files", "*.*")
    );
    File selectedFile = fileChooser.showSaveDialog(stage);
    if (selectedFile != null) {
        WritableImage modifiedImage = new WritableImage((int) imageView.getFitWidth(), (int) imageView.getFitHeight());
        imageView.snapshot(snapshotParameters: null, modifiedImage);

        // Add canvas changes
        SnapshotParameters parameters = new SnapshotParameters();
        parameters.setFill(Color.TRANSPARENT);
        Canvas tempCanvas = new Canvas(modifiedImage.getWidth(), modifiedImage.getHeight());
        tempCanvas.getGraphicsContext2D().drawImage(modifiedImage, 0, 0);
        tempCanvas.getGraphicsContext2D().drawImage(drawingCanvas.snapshot(parameters, writableImage: null), 0, 0);
        modifiedImage = tempCanvas.snapshot(parameters, writableImage: null);
        try {
            ImageIO.write(SwingFXUtils.fromFXImage(modifiedImage, bufferedImage: null), formatName: "png", selectedFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figura 2 Funcția de salvarea a unei imagini

În Figura 2 este reprezentată funcția ce permite salvarea unei imagini editate într-o locație dorită pe calculatorul personal.

```
public class SepiaFilter extends Filter {  
  
    public SepiaFilter() {  
        super("Sepia", c -> {  
            double r = Math.min(1.0, 0.393 * c.getRed() + 0.769 * c.getGreen() + 0.189 * c.getBlue());  
            double g = Math.min(1.0, 0.349 * c.getRed() + 0.686 * c.getGreen() + 0.168 * c.getBlue());  
            double b = Math.min(1.0, 0.272 * c.getRed() + 0.534 * c.getGreen() + 0.131 * c.getBlue());  
            return Color.color(r, g, b);  
        });  
    }  
}
```

Figura 3 Filtrul Sepia

În Figura 3 este arătat filtrul Sepia. Acest filtru modifică culorile inițiale a unei imagini astfel se obține o imagine nouă.

## FIȘA DE ACTIVITATE

### Săptămâna IV

---

Sarcini planificate:

- Realizarea prezentării
- Realizarea raportului

**Activități desfășurate, observații personale**

# Sistem de editare și optimizare a imaginilor

A ELABORAT: ST.GR. TI-194 CEBAN VITALIE

## Scopul și obiectivele propuse

- ▶ Principalul scop al acestui Crearea unei aplicații ce permite procesarea și optimizarea imaginilor.
- ▶ Obiectivele de baza sunt:
  - Cercetarea domeniului „Tehnologii Multimedia”;
  - Analiza soluțiilor existente de „Sisteme de optimizare a imaginilor”;
  - Elaborarea concepției sistemului;
  - Elaborarea Caietului de Sarcini; Cercetarea și selectarea instrumentelor pentru dezvoltarea platformei;
  - Crearea funcționalului de adăugare a unui filtru;
  - Crearea funcționalului de adăugare, editare și ștergere a unei imagini;



## Analiza domeniului

- ▶ Aplicația face parte din domeniul tehnologiilor multimedia. Aplicația este de tip desktop având un GUI ce permite activitățile de baza pentru o aplicație ce ar permite optimizarea unei imagini. Astfel de aplicații sunt utilizate pentru editarea imaginilor și reducerea dimensiunilor.
- ▶ Pentru realizarea acestei aplicații s-a utilizat platforma JavaFX.



## Instrumente si tehnologii utilizate

- ▶ Pentru realizarea acestei aplicații a fost ales limbajul Java. Toată aplicația fiind scrisă în Java.
- ▶ Principala librărie utilizată în elaborarea acestei aplicații a fost JavaFX. JavaFX este o platformă bazată pe Java pentru construirea de aplicații GUI.
- ▶ Pentru crearea fișierului de tip .fxml (partea vizuală) a fost utilizat Scene Builder.

