Universitatea Tehnică a Moldovei

SISTEM DE EDITARE ȘI OPTIMIZARE A IMAGINILOR

IMAGE EDITING AND OPTIMIZATION SYSTEM

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

Coordonator: **Cebotari Daria**
asistent universitar

Chişinău, 2023

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

Admis la susţinere
Şef departament:
FIODOROV Ion dr., conf.univ.

„___" _____ 2023

Sistem de editare și optimizare a imaginilor

Proiect de licenţă

Student: **_____** **Ceban Vitalie, TI-194**

Coordonator: **_____** **Cebotari Daria, asist. univ.**

Consultant: **_____** **Cojocaru Svetlana, asist.univ.**

Chişinău, 2023

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

Programul de studii Tehnologia informaţiei

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

_____

„ 23” decembrie 2022

CAIET DE SARCINI
pentru proiectul de licenţă al studentului

Ceban Vitalie
(numele şi prenumele studentului)

1. Tema proiectului de licenţă *Sistem de editare și optimizare a imaginilor*

confirmată prin hotărârea Consiliului facultăţii nr. *3* **din** *„ 23” decembrie 2022*

2. Termenul limită de prezentare a proiectului de licență *20.05.2023*

3. Date iniţiale 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 – Procesul de salvare a unui fișier

6. Lista consultanţilor:

Consultant	Capitol	Confirmarea realizării activităţii	
		Semnătura consultantului (data)	Semnătura studentului
Cojocaru Svetlana	*Standarde tehnologice, Controlul calităţii, Estimarea costului proiectului*		

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
1	*Elaborarea sarcinii, primirea datelor pentru sarcină*	*02.09.22– 30.09.22*	*10%*
2	*Analiza domeniului de studiu*	*06.10.22– 30.11.22*	*15%*
3	*Proiectarea sistemului*	*01.12.22 – 25.12.22*	*15%*
4	*Realizarea sistemului*	*10.01.23 – 05.03.23*	*30%*
5	*Descrierea sistemului*	*06.03.23– 01.04.23*	*10%*
6	*Estimarea costurilor sistemului*	*02.04.23– 15.04.23*	*15%*
7	*Finisarea proiectului*	*16.04.23– 20.05.23*	*5%*

Student *Ceban Vitalie ( )*

Coordonator de proiect de licență *Cebotari Daria ( )*

DECLARAŢIA STUDENTULUI

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ
DEPARTAMENTUL INGINERIA SOFTWARE ŞI AUTOMATICĂ
PROGRAMUL DE STUDII TEHNOLOGIA INFORMAŢIEI

AVIZ

la proiectul de licență

Titlul: 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 şi salvate cu modificări.

4. Estimarea rezultatelor obținute: Acest program este creat pentru editarea imaginilor. Este un program ușor în utilizare şi 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 utilizata de orice utilizator familiar cu un calculator.

8. Observații și recomandări: Cerințele fată 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 _____

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 şi cum au loc tranzițiile între ele. Diagramele sunt de mai multe tipuri, în dependenta 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 si 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

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 simplu 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 pot fi de mai multe tipuri, cum ar fi rastru sau vectoriale. 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 algoritmii sofisticați din editorii de imagini. O imagine raster este o imagine digitală compusă dintr-o serie de pixeli. Pixelii sunt cea mai mică unitate individuală a unei imagini raster și fiecare pixel conține informații despre culoare care sunt folosite pentru a genera imaginea finală. Imaginile raster sunt adesea folosite în fotografia digitală, precum și în design web și alte aplicații în care este nevoie de o imagine de înaltă rezoluție. Din cauza modului în care sunt stocate, imaginile raster pot apărea uneori „blocate" sau „pixelate" atunci când sunt mărite. Cele mai comune cinci formate raster sunt JPEG, PNG, GIF, BMP și TIFF. O imagine vectorială este o imagine digitală care este compusă din forme geometrice, cum ar fi puncte, linii și curbe. Fișierele vectoriale sunt adesea folosite în designul grafic pentru ilustrații și logo-uri, precum și în alte aplicații în care nu este necesară o imagine de înaltă rezoluție. Deoarece imaginile vectoriale sunt compuse din forme geometrice, ele pot fi editate și manipulate pentru a-și schimba aspectul fără a deveni „blocate" sau „pixelate". În plus, deoarece imaginile vectoriale sunt compuse mai degrabă din formule matematice decât din pixeli, acestea pot fi mărite sau reduse în dimensiune fără a pierde nicio calitate.Cele mai comune cinci formate vectoriale sunt AI, EPS, PDF și SVG. 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 sa 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 ca 2/3 din ele necesită deținerea unei conexiuni la internet pentru utilizarea lor. Acest neajuns adesea înseamnă ca 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 baza 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 incorporate, 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 intuitiva și ușor de folosit, ceea ce face ca editarea imaginilor sa fie o activitate plăcută si relaxantă. Pe lângă acestea, aplicația propusa are și un design modern și atractiv, care oferă o experiență de utilizare plăcută şi 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 ca 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

In 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 și compresie a imaginilor bazat pe cloud, care permite utilizatorilor să reducă dimensiunea imaginilor lor fără a compromite calitatea. Serviciul acceptă o gamă largă de formate de imagine, inclusiv JPEG, PNG, GIF și SVG.. Cu Kraken.io, puteți optimiza cantități mari de fișiere JPEG, PNG și GIF animate. Kraken.io oferă atât tipuri de abonament gratuite, cât și plătite, cu prețuri bazate pe numărul de imagini optimizate și nivelul de optimizare necesar. Serviciul oferă, de asemenea, o varietate de funcții la nivel de întreprindere, inclusiv branding personalizat, asistență dedicată și instrumente de management al echipei. 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 descarcă 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 inițiala a imaginii. În figura 1.1 Kraken.io Pagina principală am reprezentat pagina de pornire a acestui instrument.

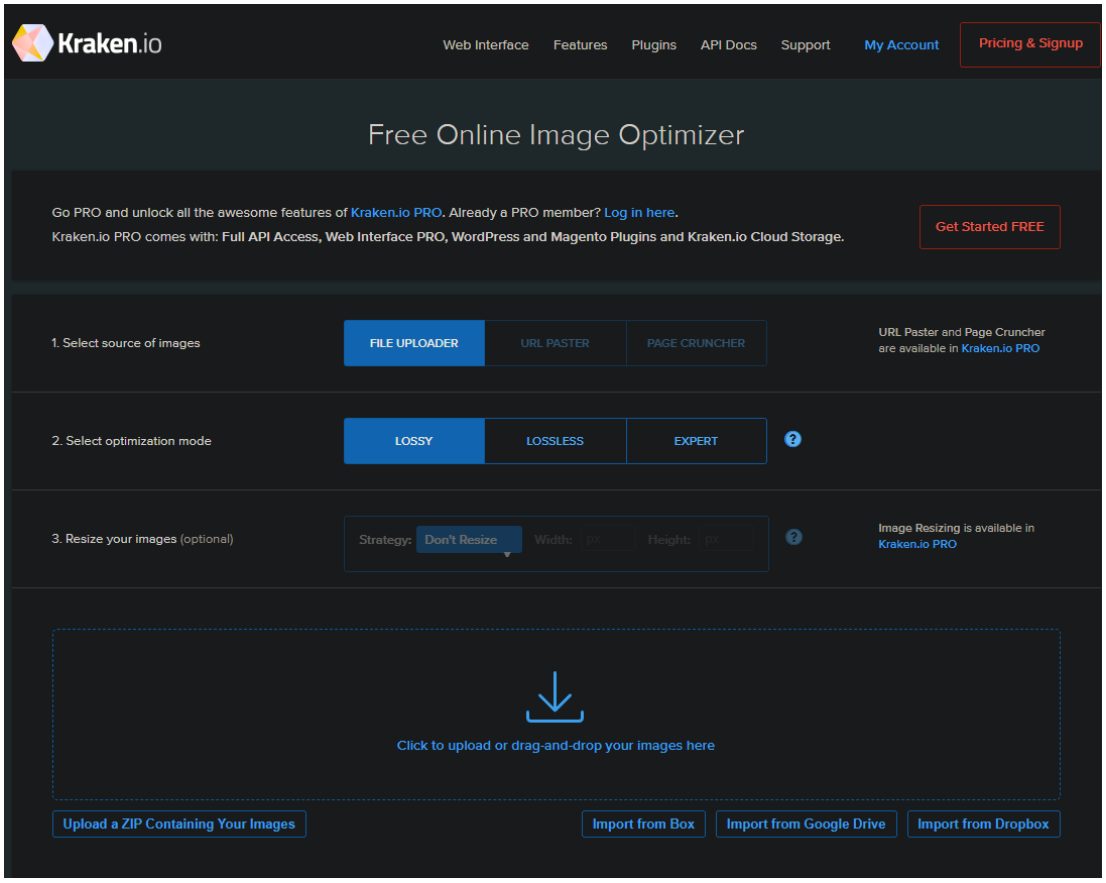

Figura 1.1 - Kraken.io Pagina principală

Pixlr este un instrument de editare a imaginilor foarte versatil, oferind o gamă largă de funcții și opțiuni pentru a satisface nevoile diferitelor tipuri de utilizatori, de la amatori la profesioniști. Acesta este disponibil în versiuni gratuite și plătite, iar cele plătite oferă mai multe opțiuni și funcții.

Una dintre caracteristicile remarcabile ale Pixlr este funcția sa de eliminare a fundalului bazată pe inteligență artificială. Această funcție utilizează algoritmi avansați pentru a identifica subiectul principal al imaginii și a elimina fundalul în mod automat. Această funcție este foarte utilă pentru designerii grafici, fotografi și alte persoane care lucrează cu imagini digitale. Pe lângă funcția de eliminare a fundalului, Pixlr oferă și alte opțiuni de editare precum ajustarea culorilor, contrastului și luminozității imaginii, adăugarea de text și grafică, eliminarea ochilor roșii, îndepărtarea petelor și alte defecte ale imaginii și multe altele. De asemenea, Pixlr este cunoscut pentru șabloanele sale de colaj prefabricate, care permit utilizatorilor să creeze rapid și ușor colaje impresionante din mai multe imagini. Această funcție este ideală pentru cei care doresc să creeze colaje pentru social media sau alte scopuri personale. În plus, Pixlr oferă o gamă largă de filtre și efecte pentru a transforma imaginile în lucrări de artă impresionante. Aceste filtre pot fi aplicate cu ușurință și pot transforma imagini obișnuite în lucrări de artă uimitoare. În figura 1.2 PIXLR Pagina principală am reprezentat pagina de pornire a acestui instrument [7].

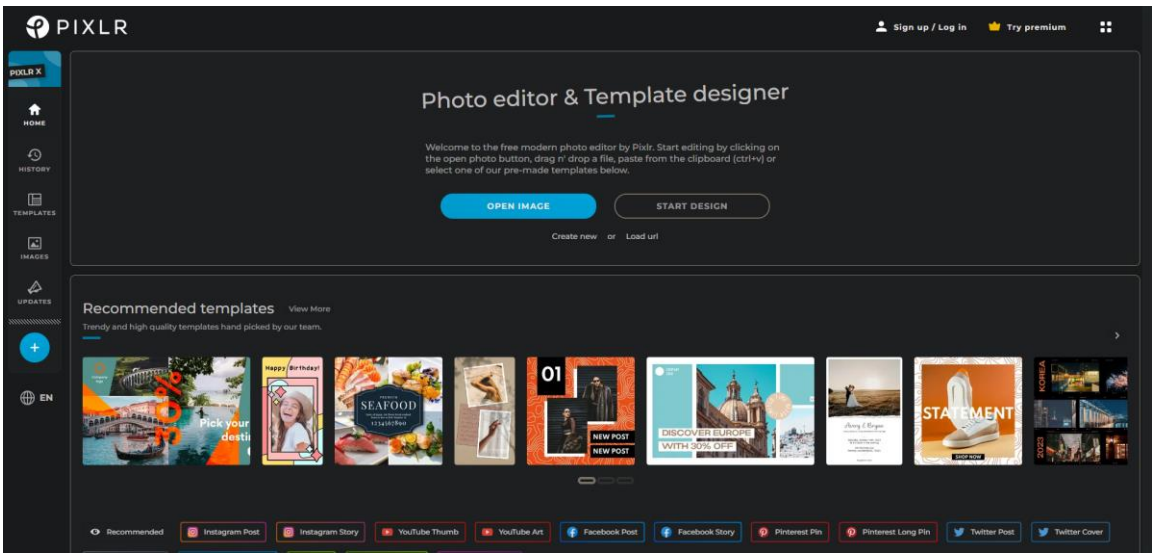

Figura 1.2 - PIXLR Pagina principală

Pixlr este un o aplicație 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.

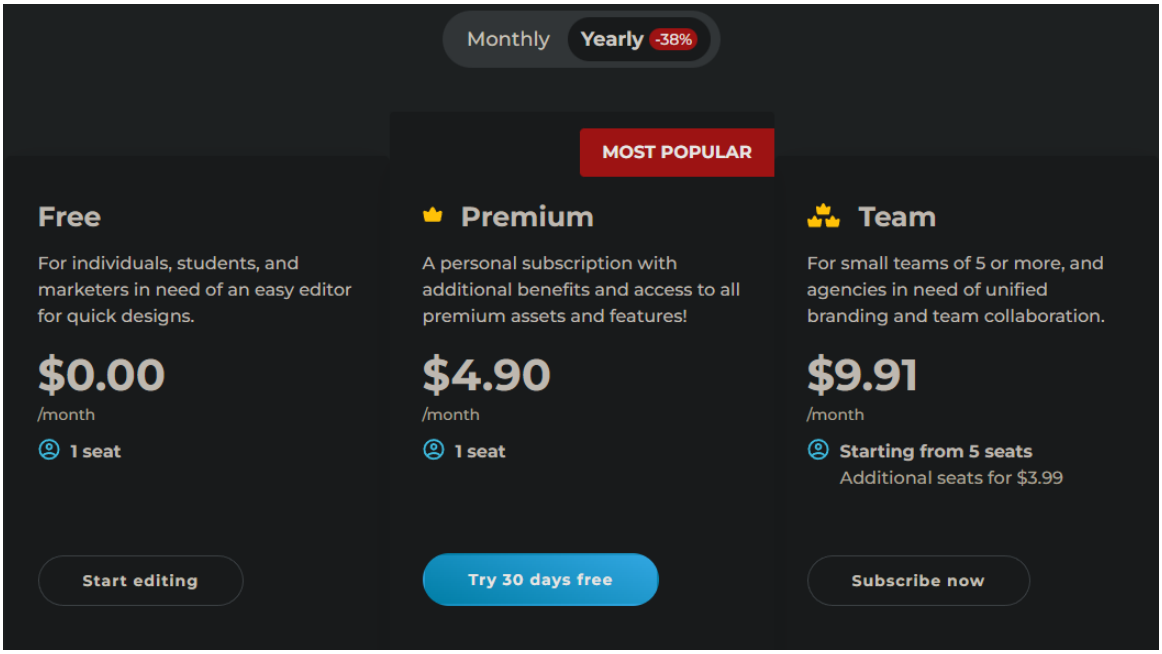

Figura 1.3 - PIXLR Abonamente

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 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. În plus față de funcțiile sale standard, Photoshop oferă, de asemenea, suport pentru o varietate de plugin-uri și extensii, care pot îmbunătăți funcționalitatea acestuia și își pot extinde capacitățile. Programul include, de asemenea, suport pentru o gamă largă de formate de fișiere, inclusiv formate de fișiere de imagine populare, cum ar fi JPEG, PNG și GIF, precum și formate utilizate în producția video, tipar și web design. Singurul minus este prețul, pentru o simpla editare utilizatorul este obligat sa procure aplicația care are un preț destul de ridicat. În figura 1.3 Interfața Photoshop am reprezentat interfața principală a aplicației Photoshop [8].

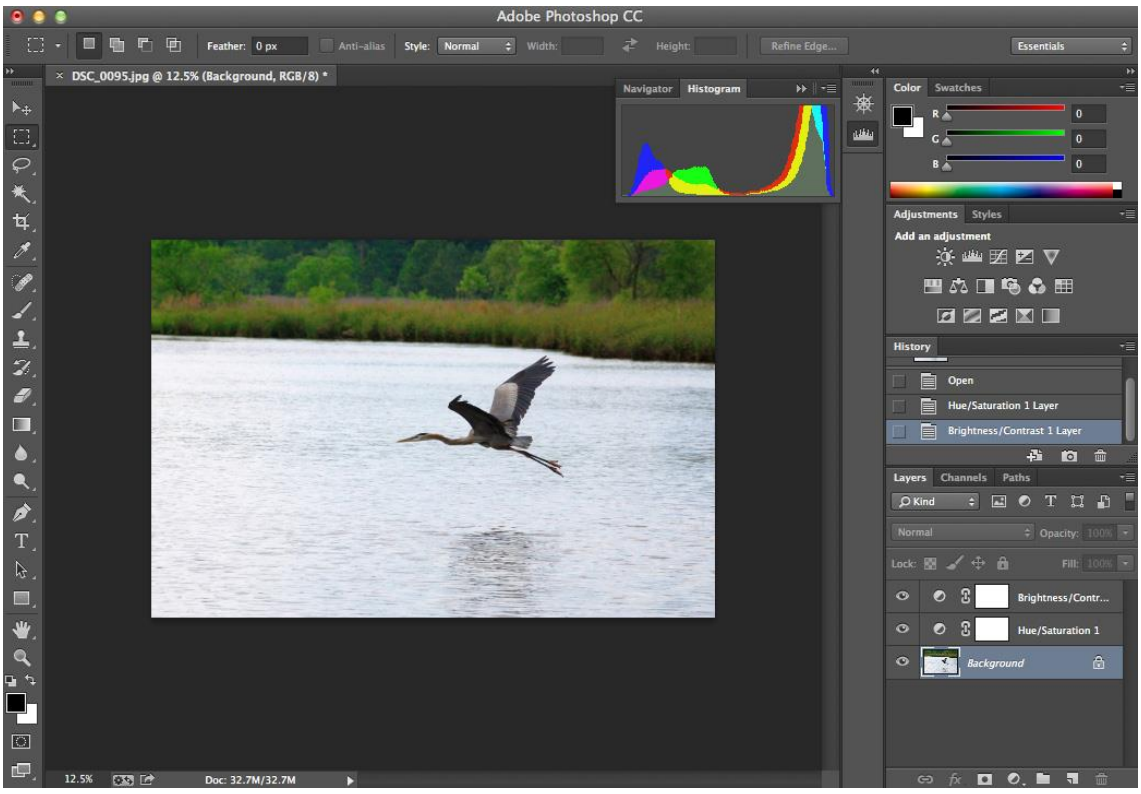

Figura 1.3 - Interfața Photoshop

Photoshop este unul dintre cele mai avansate programe de editare a imaginilor disponibile pe piață. Funcționalitatea sa largă îi permite utilizatorului să facă mult mai mult decât doar să ajusteze balanța culorilor, aplicând filtre artistice sau efecte picturale. Cu Photoshop, puteți să vă îmbunătățiți imaginile prin retușarea lor în moduri foarte precise. Aceasta include eliminarea petelor și defectelor minore, eliminarea ochilor roșii sau reducerea zgomotului din imagine. Programul vă permite să creați chiar și colaje de imagini, să adăugați text și să aplicați efecte 3D pentru o experiență de editare de nivel înalt. În plus, Photoshop oferă și posibilitatea de a edita fișierele în diferite moduri de culoare, inclusiv CMYK pentru imprimare, ceea ce face din acesta un instrument preferat pentru profesioniști în domeniul fotografiei, designului grafic și editare video.

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 în figura 1.4 Photoshop Balanță de culori.

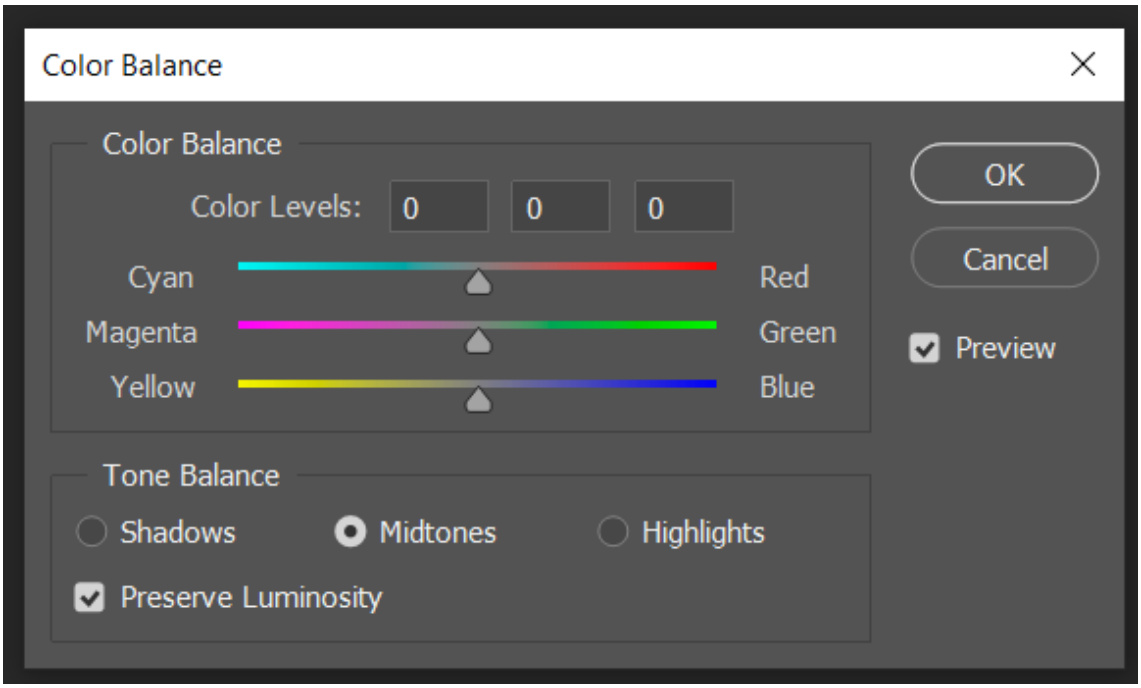

Figura 1.4 - Photoshop Balanță de culori

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 în Photoshop este aratat în figura 1.5 Filtrele Photoshop.

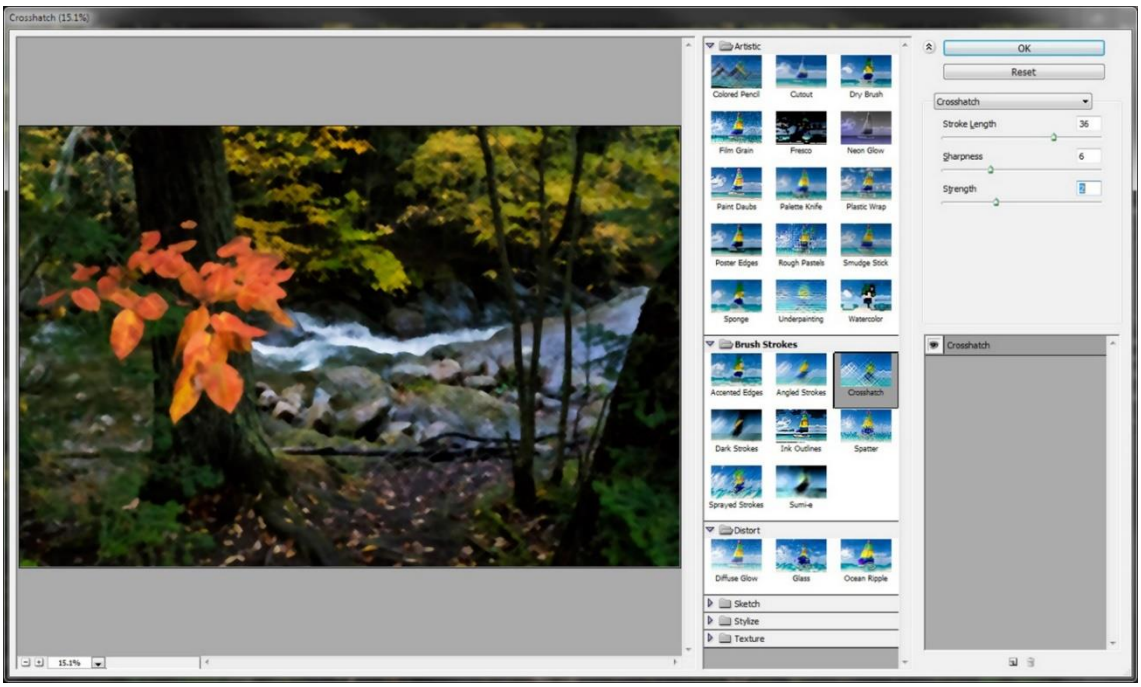

Figura 1.5 - Filtrele Photoshop

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 şi instrumente prestabilite. Un exemplu de astfel de imagine este reprezentat în figura 1.6 Photoshop efectul Painterly.

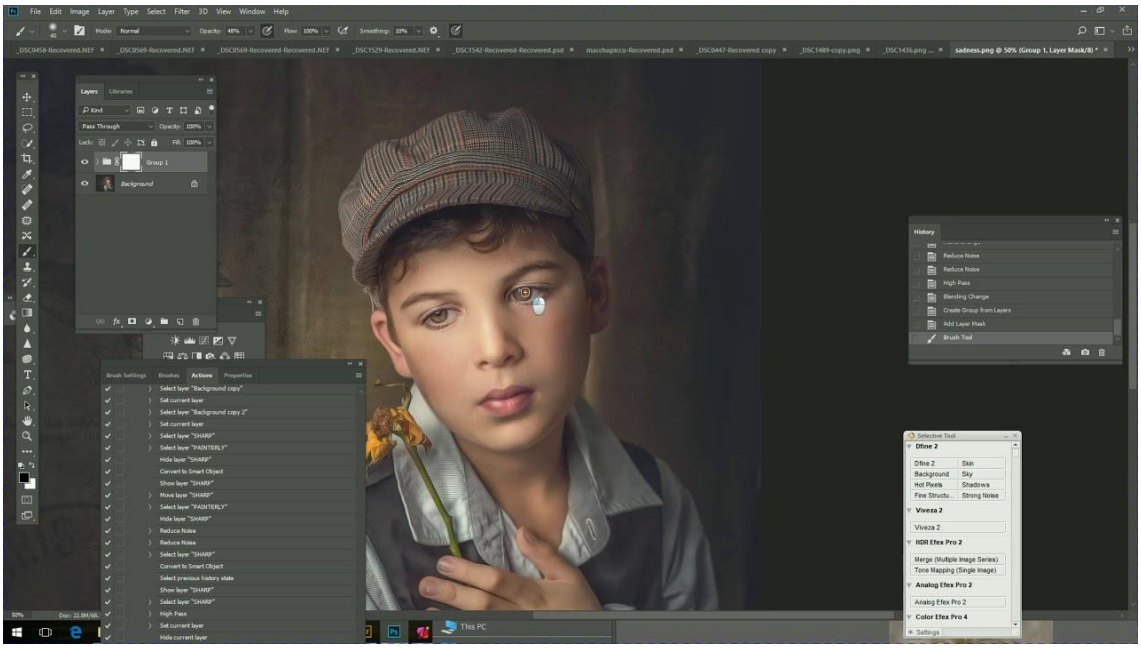

Figura 1.6 - Photoshop efectul Painterly

Ultimul exemplu a unuia din funcționalitățile Photoshop este retușarea imaginilor. Retușarea imaginilor este o metodă esențială în editarea imaginilor și este utilizată nu doar în industria de fotografie,

ci și în publicitate, design grafic și alte domenii de creație. Photoshop oferă o gamă largă de instrumente și opțiuni pentru retușarea imaginilor, cum ar fi:

- corectarea erorilor de expunere, eliminarea petelor și zgârieturilor din imagine;
- corectarea tonurilor de piele, îndepărtarea ridurilor și a semnelor de îmbătrânire;
- ajustarea culorilor și a luminozității, îmbunătățirea contrastului și a saturației;
- adăugarea de elemente grafice sau texte;
- îndepărtarea sau înlocuirea obiectelor din imagine.

Toate aceste opțiuni și multe altele permit utilizatorilor să transforme o imagine banală într-un produs final impresionant și de calitate. Retușarea imaginilor este un proces esențial în procesul de producție de fotografii și design grafic, iar Photoshop este una dintre cele mai populare și avansate soluții pentru această activitate. Modul în care arata acest funcțional și un exemplu al utilizării lui este reprezentat în figura 1.7 Photoshop Retușarea imaginii.

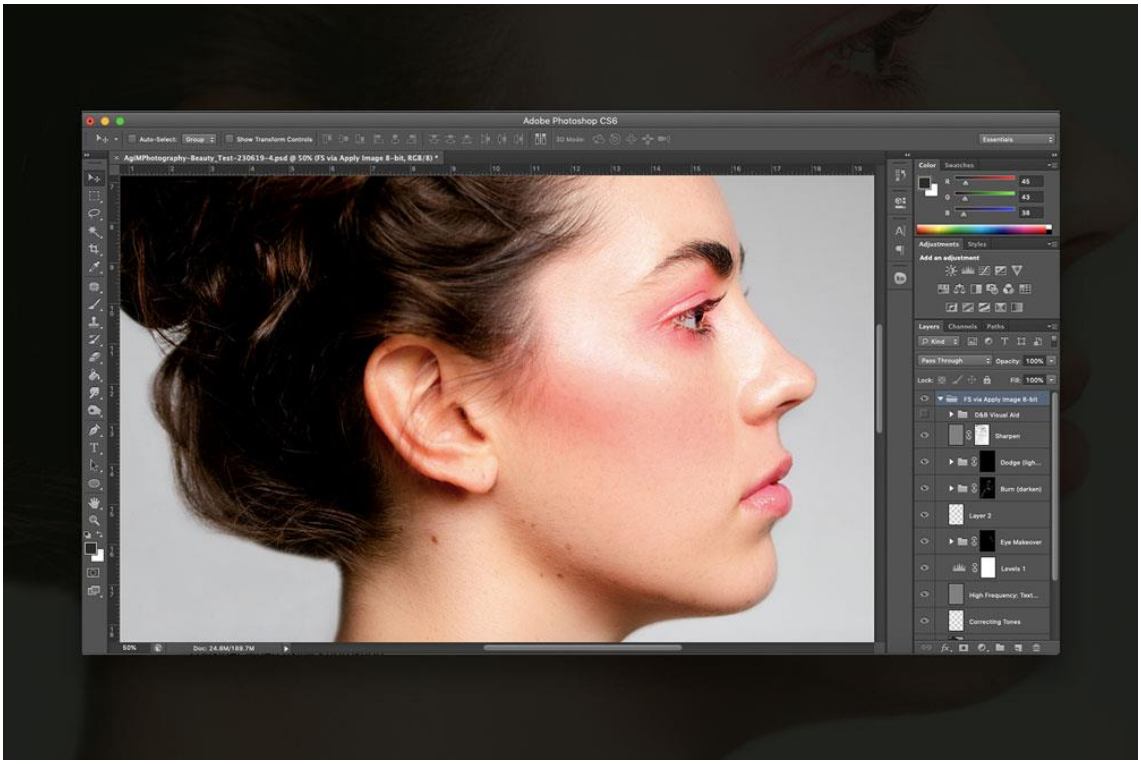

Figura 1.7 - Photoshop Retușarea imaginii.

1.3 SCOPUL, OBIECTIVELE ȘI CERINȚELE SISTEMULUI

Scopul: Crearea unei aplicații ce permite procesarea şi optimizarea imaginilor.

Obiectivele:

- cercetarea domeniului „Tehnologii multimedia”;
- analiza soluțiilor existente de „Sisteme de optimizare a imaginilor”;

- concluzii şi argumentarea propunerii de proiect „Sistem de optimizare a imaginilor”;
- elaborarea concepției sistemului;
- elaborarea caietului de sarcini;
- argumentarea platformei și soft-ului de realizare;
- cercetarea şi 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 şi ștergere a unei imagini;
- crearea componentei de vizualizarea a elementului ce urmează a fi editat;
- testarea sistemului;
- documentarea sistemului informațional;
- planificarea proiectului şi estimarea costului;
- concluzii și recomandări;

Cerintele:

Sistemul de editare şi 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.

Aplicarea unui filtru trebuie să permită utilizatorului să selecteze un filtru și să aplice acest filtru asupra imaginii. Acesta poate fi un filtru de culoare, de contrast, de luminozitate sau orice alt tip de filtru. De asemenea aplicația ar trebui să poată lucra cu o varietate de formate de fișiere, cum ar fi JPEG, PNG, astfel încât utilizatorii să poată aplica filtre pe diferite tipuri de fișiere.

Opțiunea de a salva o imagine editată trebuie sa ofere posibilitatea de a selecta locul în care fișierul urmează a fi salvat și formatul in care acesta are sa fie salvat. De asemenea aceasta funcție trebuie sa ia în considerație toate modificările efectuate asupra fișierului și să îl salveze cu toate aceste schimbări prezente în fișierul final. Aplicația ar trebui să aibă o performanță rapidă și să nu ia mult timp pentru a procesa și a salva imaginea editată. Nu în ultimul rând aplicația ar trebui să aibă o interfață de utilizator intuitivă și ușor de utilizat, astfel încât chiar și utilizatorii neexperimentați să poată utiliza aplicația cu ușurință.

Funcționalitatea de rotire - aplicația trebuie să permită utilizatorului să rotească imaginea în orice direcție, cu diferite grade de rotație. Interfața cu utilizatorul urmează să fie intuitivă și ușor de utilizat, astfel încât chiar și utilizatorii neexperimentați să poată utiliza aceasta funcționalitate a aplicației fără careva probleme sau dificultăți. Aplicația ar trebui să aibă o performanță rapidă și să nu ia mult timp pentru a procesa și a roti imaginea.

Funcționalitatea de decupare - aplicația trebuie să permită utilizatorului să selecteze o zonă din imagine și să o decupeze. Funcționalul ce permite decuparea trebuie să permită utilizatorului să selecteze zona care el dorește să fie decupată, aceasta zonă trebuie să salveze modificările ce au avut loc asupra ei în acțiunile precedente.

O altă funcționalitate ce urmează a fi implementată este posibilitatea de a adaugă detalii, acest lucru are sa fie implementat prin intermediul posibilității de lucra cu un așa zis creion ce v-a permite toate aceste opțiuni.

Aplicația dată nu necesită cerințe de siguranță din motivul că nu stochează date sensibile sau date personale. O astfel de aplicație nu necesita nici date de autentificare deoarece nu rulează web, ea poate fi utilizata imediat după instalare. Pentru a menține o performanță maximă a sistemului, singura cerință pentru o performanță maximală este deținerea unui PC cu componente din lista de cerințe minime enumerate in subcapitolul 2.5. Utilizarea aplicației în mai multe instanțe este posibila însă acest fapt v-a reduce performanța totală a PC-ului în cazul în care acesta nu este cel mai performant. De asemenea utilizarea a tuturor operațiunilor posibile asupra unui document v-a creste sarcina totală asupra procesorului deoarece el este responsabil de îndeplinirea instrucțiunilor dorite de către utilizator.

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 însă in cazul curent apariția unei schimbări nu a influențat major procesul de elaborare datorita faptului ca dimensiunile întregii aplicații nu sunt mari.

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 complexe, 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 realizata prin diagrama use case unde se descrie interacțiunea între sistem și utilizatorii săi, precum și interacțiunea între utilizatori. O implementare a acestei diagrame este arătată în 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 activitate poate fi utilă și pentru a identifica posibilele probleme sau zone de îmbunătățire în fluxul de lucru existent. De exemplu, prin utilizarea acestei diagrame se poate observa dacă există etape redundante sau etape care pot fi eliminate pentru a simplifica procesul. De asemenea, poate ajuta la identificarea unor eventuale conflicte sau probleme de securitate care pot apărea în fluxul de lucru. Diagrama de activitate poate fi, de asemenea, folosită pentru a planifica și documenta fluxul de lucru într-un mod clar și precis. Acest lucru poate fi util atât pentru dezvoltatorii software, cât și pentru utilizatorii finali ai sistemului, întrucât poate facilita înțelegerea fluxului de lucru și poate contribui la o utilizare mai eficientă a sistemului. În figura 2.2 este reprezentată Diagrama de activitate a salvării unui fișier.

Afișarea stărilor de tranzacție este realizata 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. În figura 2.3 Diagrame de stare a selectării operațiunii am reprezentat diagrama dată.

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, ea oferă o modalitate eficientă de a ilustra interacțiunile dintre diferitele componente ale sistemului într-un mod grafic și ușor de înțeles. Un exemplu a unei astfel de diagrame si a modului in care ea funcționează este reprezentată în figura 2.4 Diagrama de secvență a procesului de adăugare a unui fișier.

Diagrama de Colaborare (Collaboration Diagram) este un tip de diagramă de modelare a comportamentului care poate fi utilizată pentru a modela interacțiunile dintre obiectele dintr-un sistem software într-un anumit scenariu de utilizare. Scopul său este de a prezenta modul în care obiectele

interacționează între ele pentru a îndeplini anumite sarcini și de a identifica eventuale probleme de comunicare sau dependențe. Într-o diagramă de colaborare, obiectele sunt reprezentate ca noduri și mesajele dintre ele sunt reprezentate ca legături. Fiecare mesaj este asociat cu o anumită acțiune și poate fi sincron sau asincron. Sincronizarea implică așteptarea ca destinatarul să execute acțiunea înainte ca expeditorul să continue, în timp ce asincronul permite expeditorului să continue execuția fără așteptarea răspunsului destinatarului. Interacțiunile dintre obiecte și modul in care ele sunt apelate este reprezentat în 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 arătata 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.

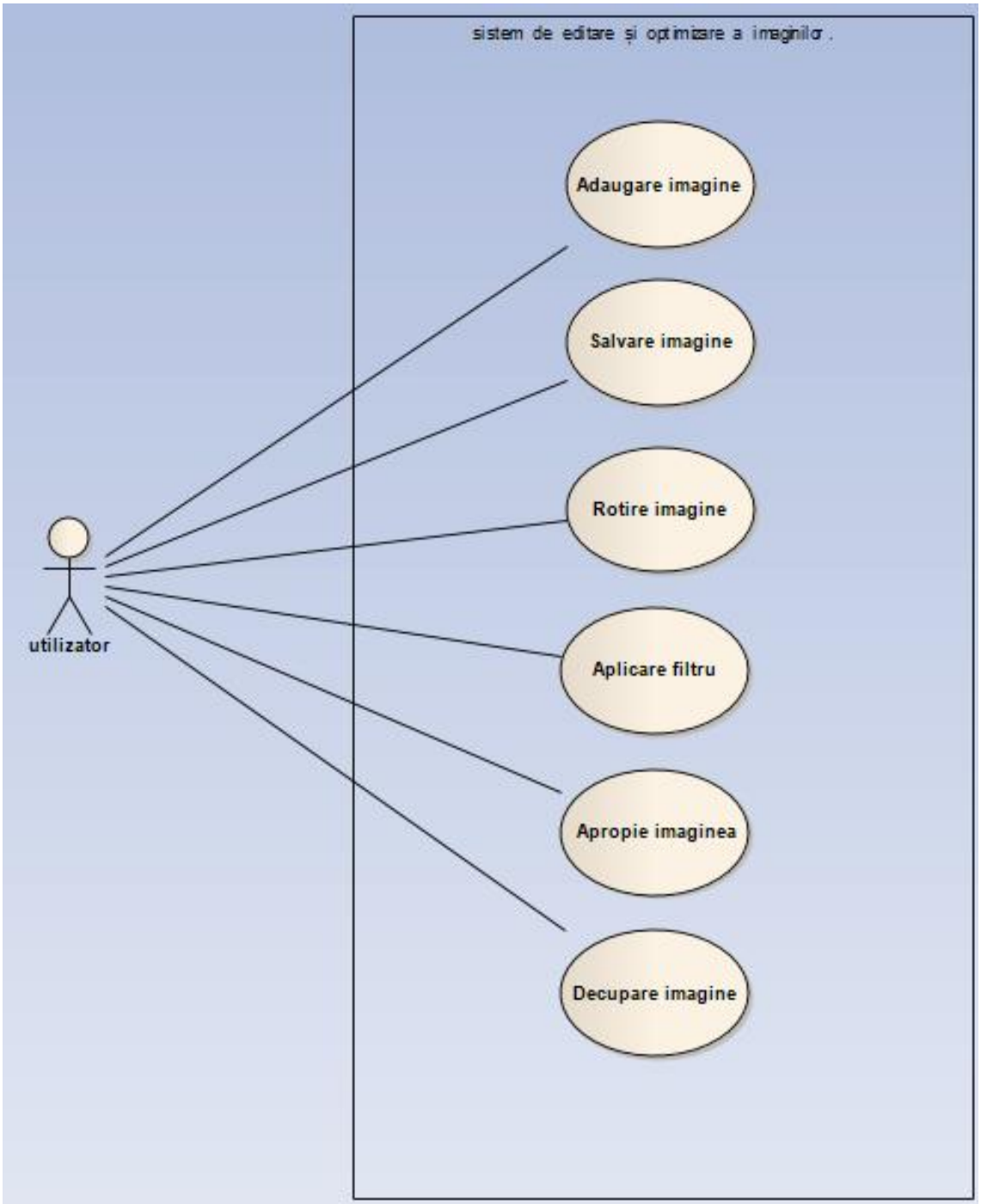

Figura 2.1 – Cazurile de utilizare pentru interacțiunea cu sistemul

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.

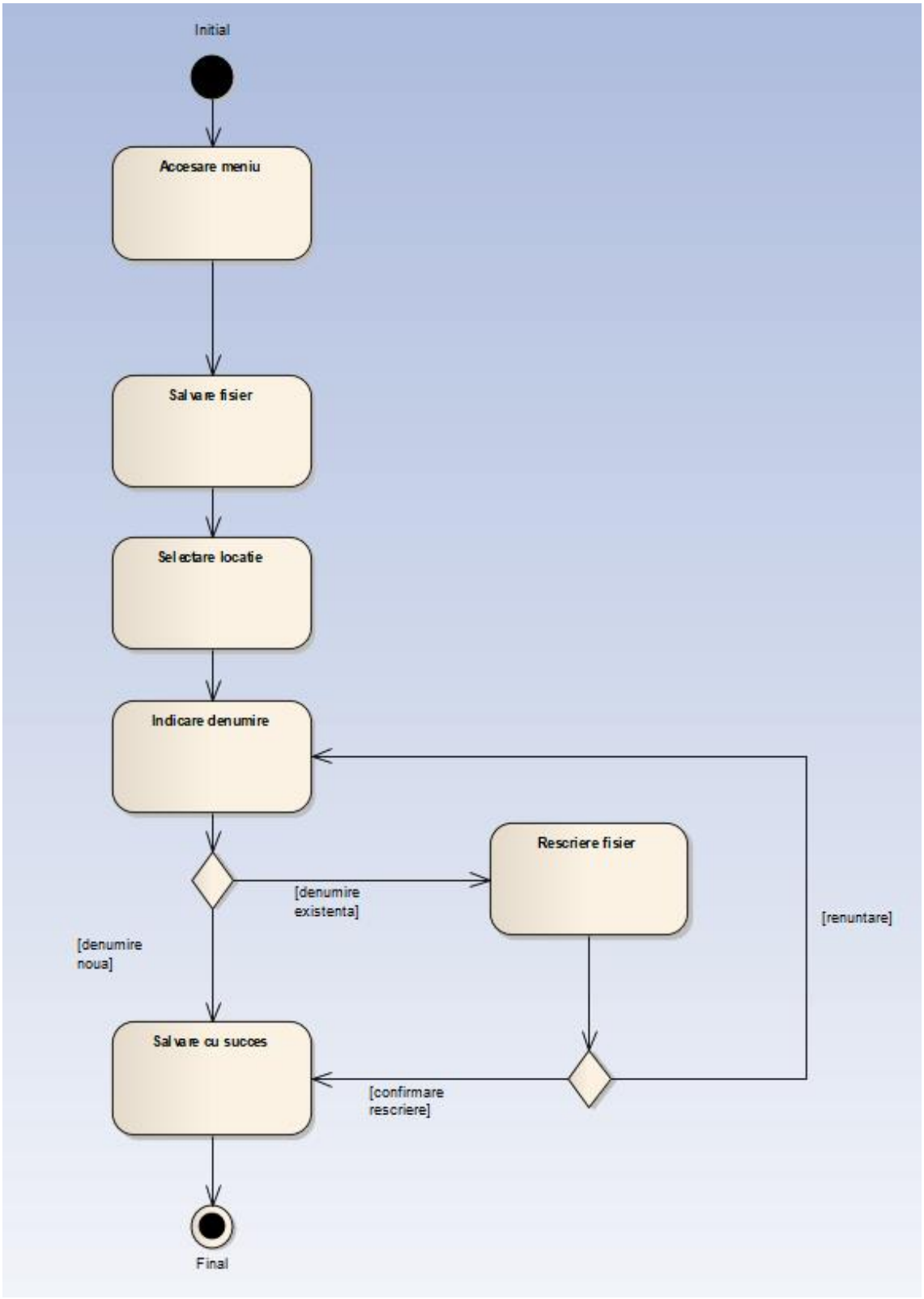

Figura 2.2 – Procesul de salvare a unui fișier

Un alt proces important pentru aceasta aplicație este funcționalul de a anula o modificare efectuată prin accesarea unei combinații de la tastatură. Acest proces presupune ca în urma unei acțiuni utilizatorul se poate întoarce la starea precedentă, în cazul în care nu exista stări precedente nimic nu se v-a întâmpla. Diagrama de activitate pentru acest proces este indicată în figura 2.3.

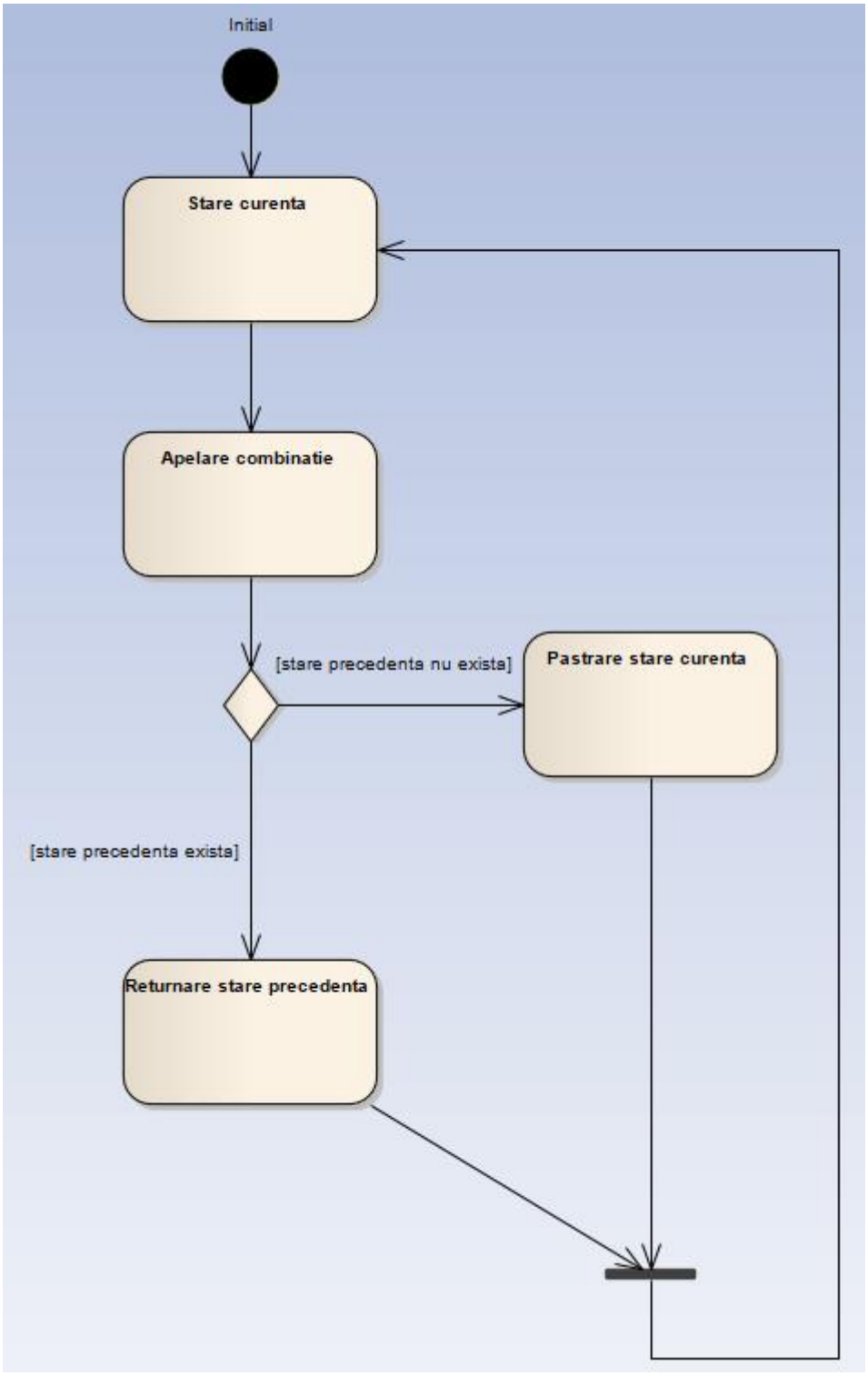

Figura 2.3 – Returnare la starea precedentă

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 2.4 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 şi tipul, în cazul în care selectarea este eronată procesul de selectare a unei acțiuni se începe de la început, în cazul selectării cu succes are loc ieșirea din acest state-machine și se efectuează operațiunea selectata de către utilizator. Operațiunile disponibile sunt create prealabil la rularea aplicației, ele fiind disponibile către selectare prin accesarea unor meniuri.

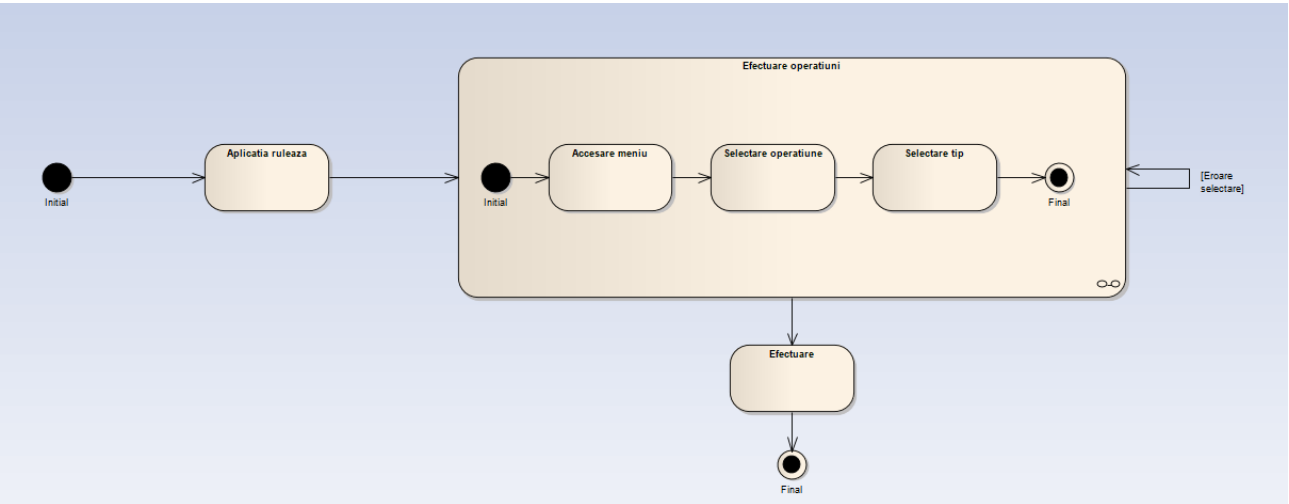

Figura 2.4 – Selectarea unei operațiuni

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 2.5 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 şi cum are loc adăugare la nivel intern.

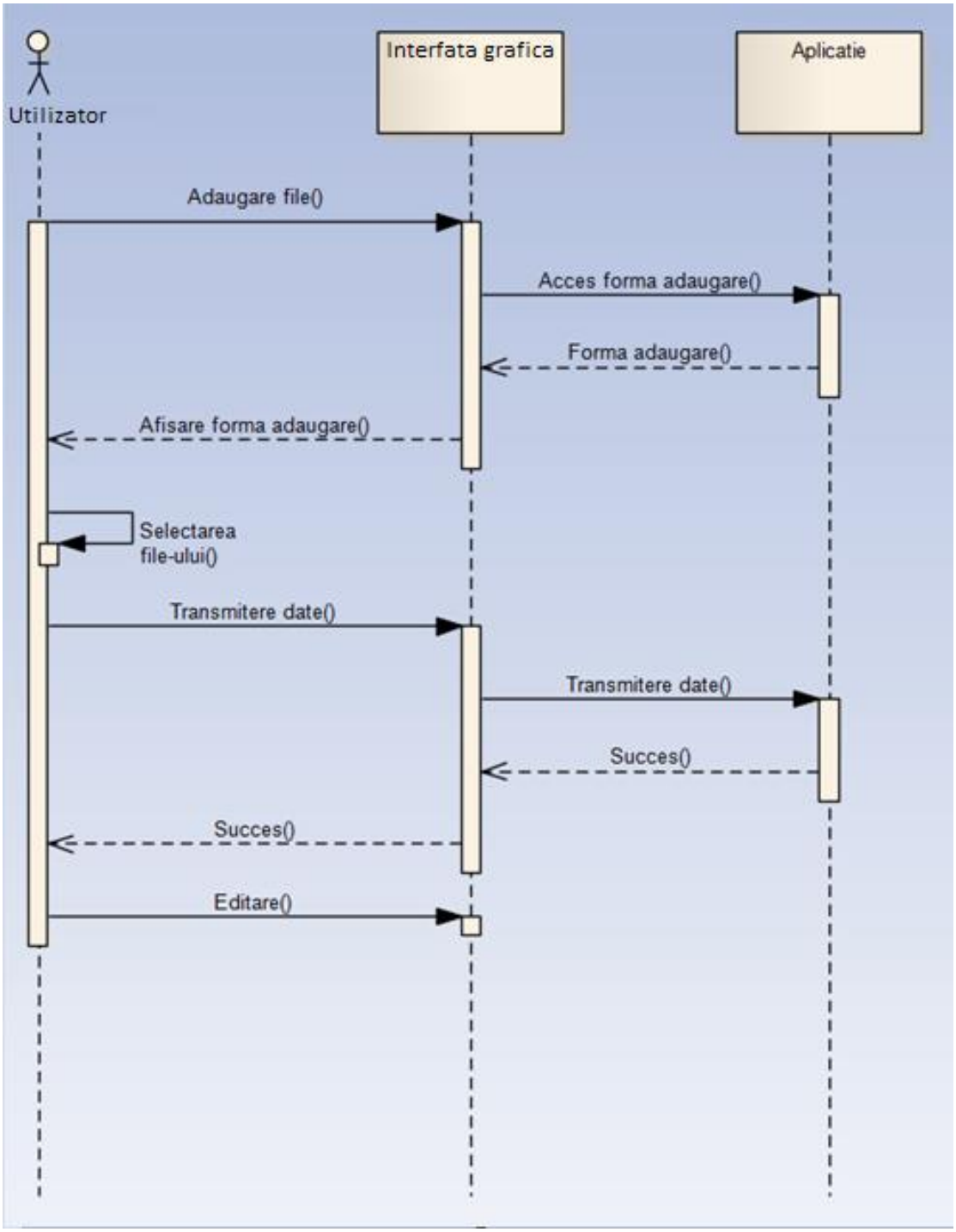

Figura 2.5 – Procesul de adăugare a unui fișier nou

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 2.6 este arătată diagrama de colaborare a procesului de decupare. În aceasta diagrama sunt indicate toate cererile care sunt efectuate de către utilizator către aplicație și răspunsurile care el le primește în cazul in care operațiunea este efectuata cu succes. Operațiunile sunt de tip call și return, unde fiecare cerere are un răspuns propriu

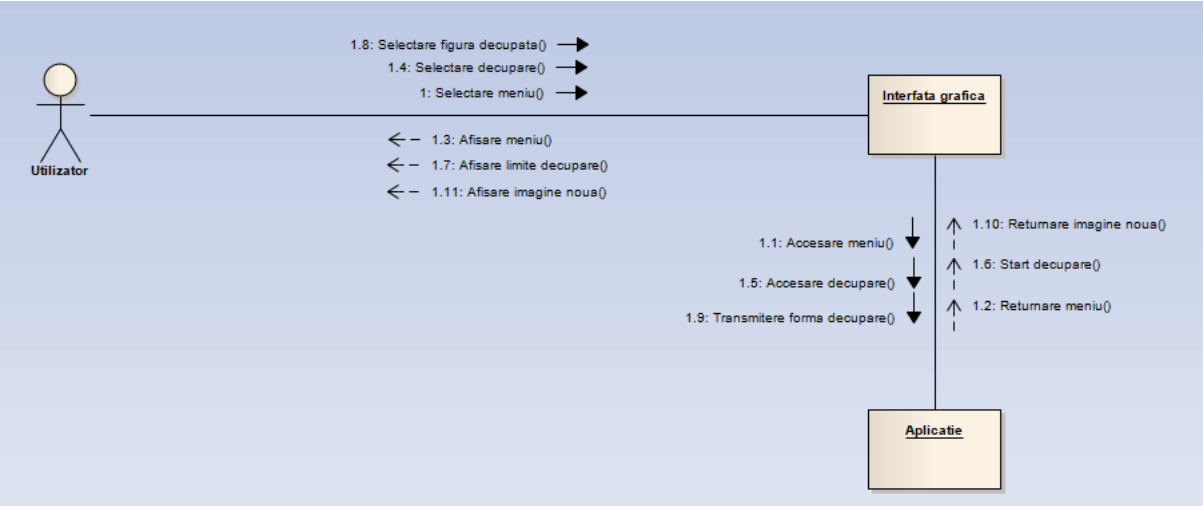

Figura 2.6 – Reprezentarea procesului de decupare

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 atribute și metode, împreună cu relațiile pe care le are cu alte clase. Aceste relații pot fi de tipul moștenire, asociere, agregare sau compoziție.(figura 2.7 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 2.8 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 2.9 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 2.7 este arătată diagrama de clasă a aplicației. În ea sunt indicate clasele principale, relațiile între ele şi părțile componente a fiecărei clase.

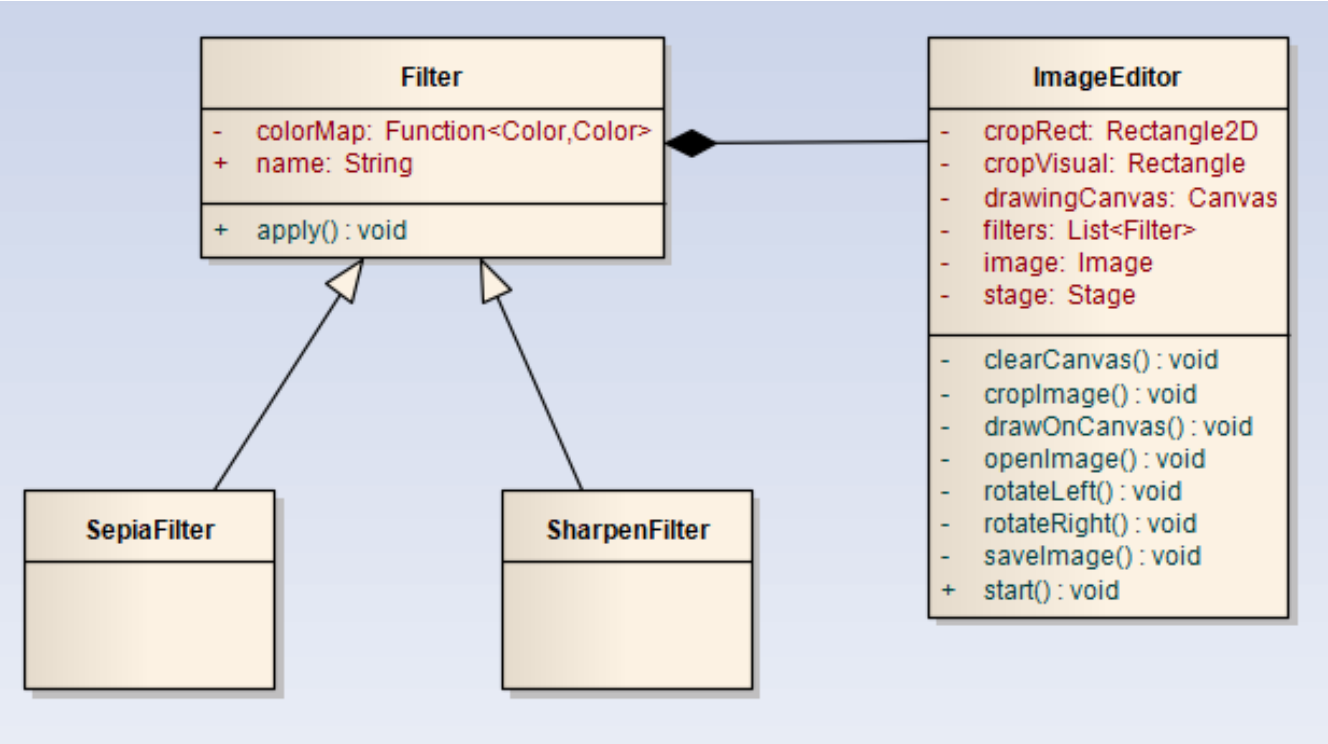

Figura 2.7 – Clasele aplicației

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 2.8 este reprezentată diagrama de componente a aplicației. În aceasta diagramă sunt indicate părțile componente a sistemului și modul în care ele interacționează între ele.

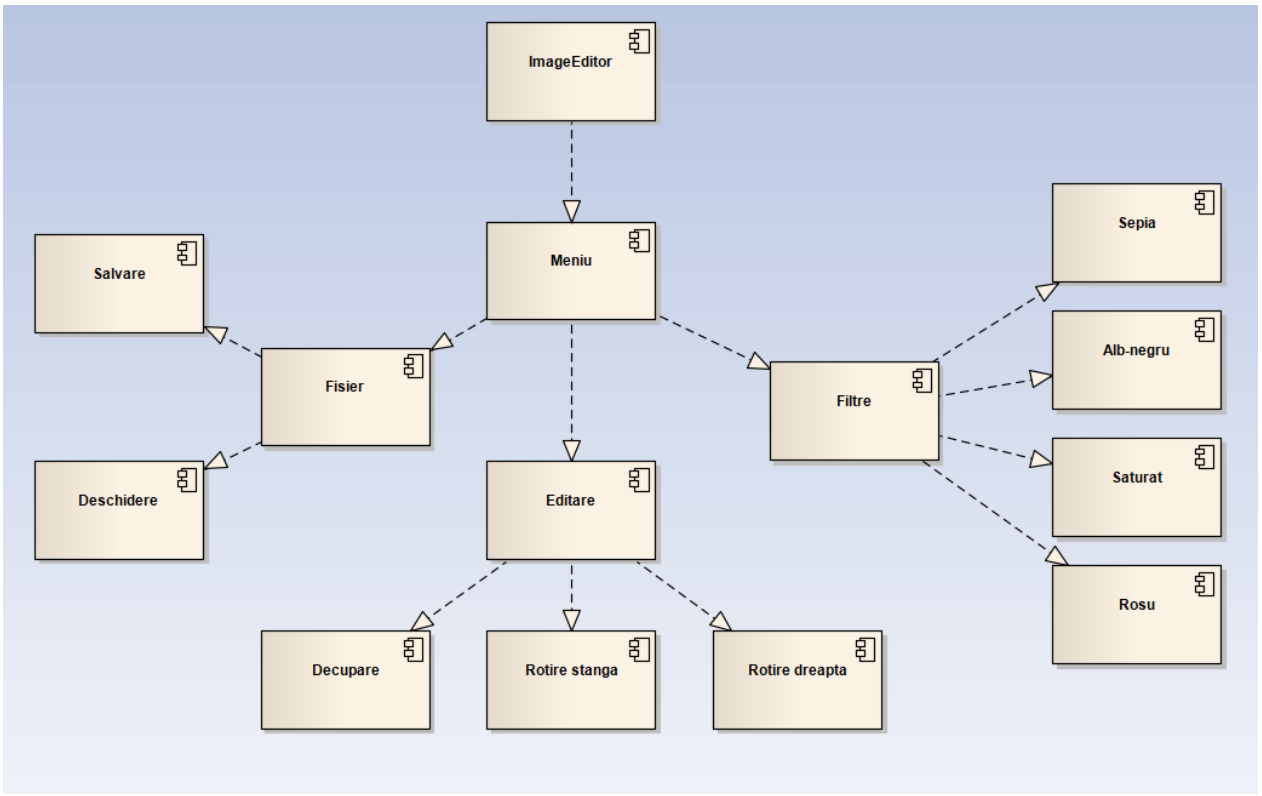

Figura 2.8 – Părțile componente a aplicației

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 2.9 este reprezentată diagrama de implementare a sistemului. În aceasta diagrama sunt reprezentate nodurile principale si relațiile dintre ele.

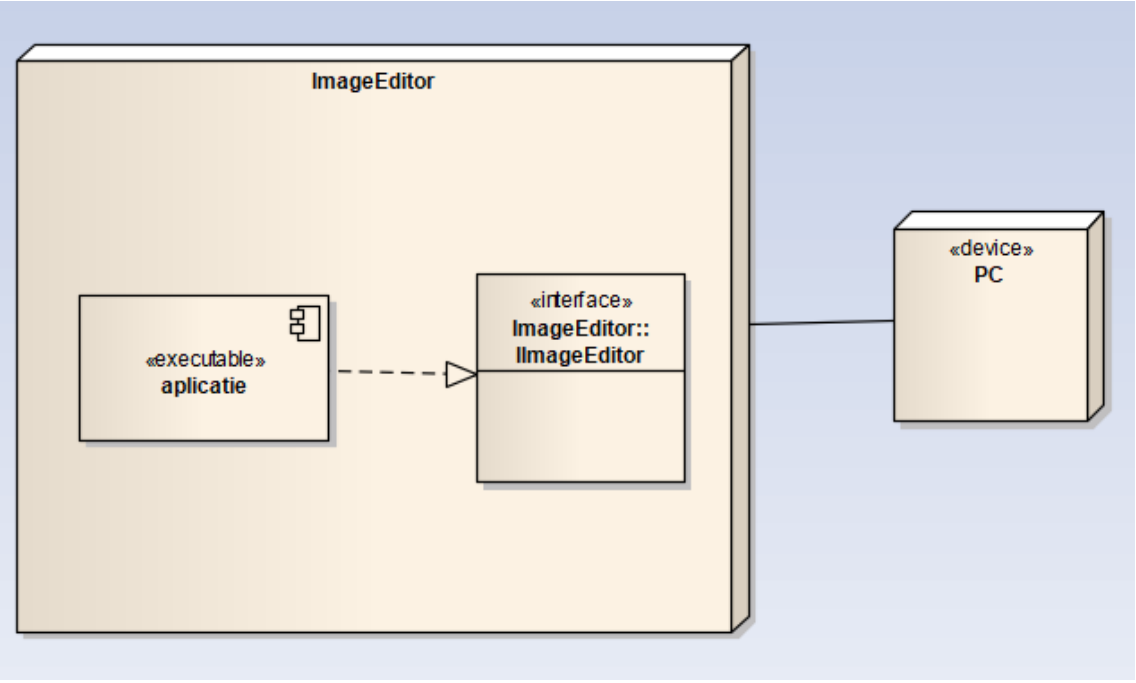

Figura 2.9 – Nodurile principale a sistemului

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. Java este un limbaj orientat pe obiecte, ceea ce înseamnă că se bazează pe conceptul de obiecte, care sunt instanțe ale claselor. Aceasta suportă multe caracteristici ale programării orientate pe obiecte, inclusiv încapsularea, moștenirea și polimorfismul. Java este de asemenea cunoscut pentru robustețea sa și pentru caracteristicile de securitate. Limbajul include gestionarea automată a memoriei prin colectarea de gunoi, care ajută la prevenirea scurgerilor de memorie și altor erori legate de memorie. Java include, de asemenea, un model de securitate care ajută la protejarea împotriva codului rău intenționat, inclusiv sandboxing-ul și încărcătoarele de clase. Java are o comunitate mare și activă de dezvoltatori, care au creat multe biblioteci, cadre și instrumente care fac mai ușoară dezvoltarea de aplicații Java. Unele cadre populare Java includ Spring, Hibernate și Struts, în timp ce instrumentele populare includ Eclipse și IntelliJ IDEA.

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. JavaFX include, de asemenea, un API de scenă grafică, care permite dezvoltatorilor să creeze interfețe utilizator complexe, structurate ierarhic. Acesta suportă, de asemenea, stilizarea și personalizarea elementelor interfeței utilizator, permițând dezvoltatorilor să creeze o aparență și o simțire coerente în cadrul unei aplicații. JavaFX este conceput pentru a funcționa perfect cu alte tehnologii Java, inclusiv mașina virtuală Java (JVM), kitul de dezvoltare Java (JDK) și alte biblioteci și framework-uri. Poate fi utilizat pentru crearea de aplicații desktop independente, precum și aplicații web și mobile, utilizând instrumente precum JavaFXports. JavaFX oferă numeroase beneficii dezvoltatorilor, inclusiv ușurința de utilizare, independența de platformă și un set bogat de componente GUI. De asemenea, are un suport puternic pentru animații și

tranziții, ceea ce-l face potrivit pentru crearea de interfețe utilizator captivante și dinamice. În general, JavaFX este o soluție puternică și flexibilă pentru crearea de interfețe utilizator moderne, vizual bogate, pe o varietate de platforme și dispozitive.

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. După ce dezvoltatorii finalizează design-ul interfeței de utilizator în Scene Builder, pot exporta fișiere FXML care pot fi utilizate în aplicația JavaFX. Printre caracteristicile cheie ale Scene Builder se numără posibilitatea de a pre-visualiza interfața de utilizator în timp real, de a utiliza elemente de interfață de utilizator personalizate, de a oferi suport pentru internaționalizare și de a permite dezvoltatorilor să conecteze elemente de interfață de utilizator la cod Java. În general, Scene Builder este o unealtă utilă și eficientă pentru dezvoltatorii JavaFX care doresc să creeze interfețe de utilizator grafice atractive și funcționale pentru aplicațiile lor fără a fi nevoie să scrie cod manual.

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. Printre caracteristicile cheie ale IntelliJ IDEA se numără inteligența artificială integrată, care ajută la dezvoltarea rapidă și precisă a codului, precum și integrarea cu sistemul de control al versiunii și posibilitatea de a lucra cu multiple proiecte simultan. 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. În general, IntelliJ IDEA este un IDE puternic și ușor de utilizat, care poate ajuta dezvoltatorii Java să dezvolte aplicații complexe și robuste într-un mod eficient și productiv.

Memento este un design pattern de tip comportamental în ingineria software-ului, care permite unui obiect să captureze și să salveze starea sa internă la un moment specific în timp, fără a încălca încapsularea. Acest pattern este folosit pentru a oferi funcționalitatea de undo/redo și este adesea implementat în editoare de text, instrumente de design grafic și alte aplicații în care utilizatorii pot modifica starea unui obiect. 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 operațiile 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 3.1

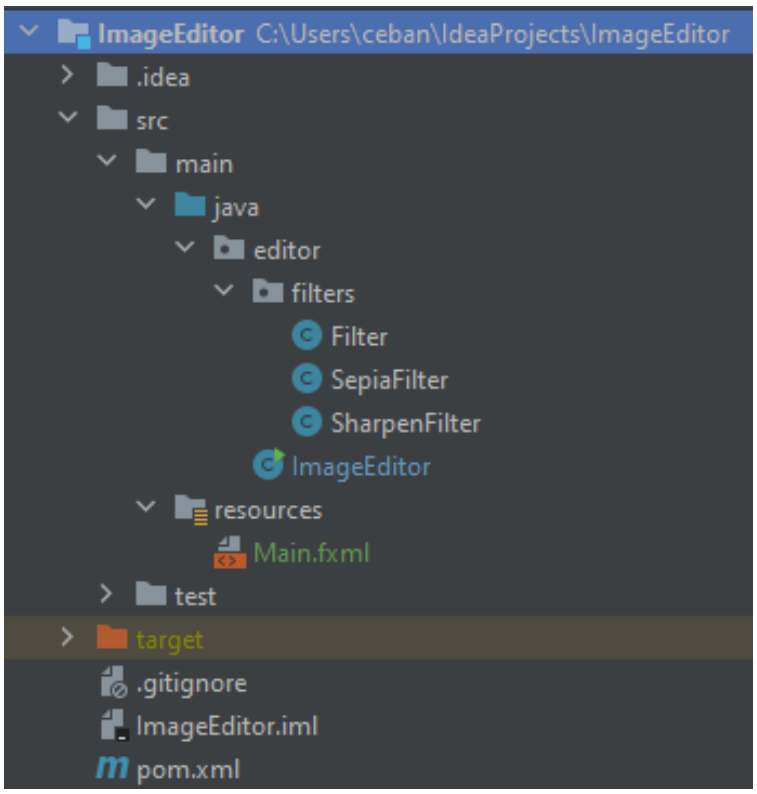

Figura 3.1 – Structura generală

În secvența de cod de mai jos este indicata 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();
        }
    }
}
```

O metodă mai complexă este modul de decuparea a unei secvențe dintr-o imagine. Pentru aceasta a fost creată metoda `cropImage()`, această metodă este folosită pentru a decupa o imagine afișată într-un widget JavaFX ImageView. Metoda creează un dreptunghi vizual peste imagine pentru a indica zona de decupare și permite utilizatorului să ajusteze dreptunghiul folosind evenimente de mouse. Obiectul cropVisual este un obiect Rectangle folosit pentru a vizualiza zona de decupare. Este inițializat cu o margine albă și umplere albă semi-transparentă. Variabila parentPane se referă la containerul părinte Pane al widget-

ului ImageView, care este folosit pentru a adăuga cropVisual la graful de scenă. Variabila cropRect este un obiect Rectangle2D care reprezintă dreptunghiul actual de decupare. Inițial, se întinde pe întreaga imagine. Metoda configurează trei manipulatoare de evenimente pentru evenimentele de mouse pe un obiect drawingCanvas, care nu este definit în această mostră de cod. Când utilizatorul apasă butonul mouse-ului, se creează un nou cropRect la poziția mouse-ului. Când utilizatorul trage mouse-ul, cropRect este actualizat să se întindă de la poziția inițială a mouse-ului la poziția curentă a mouse-ului. Când utilizatorul eliberează butonul mouse-ului, manipulatoarele de evenimente sunt deregistrate și cropRect este verificat pentru a se asigura că are lățime și înălțime nenule înainte de a finaliza decuparea prin apelarea metodei `finishCrop()`. Dacă cropRect are lățime sau înălțime zero, decuparea este anulată prin apelarea metodei `cancelCrop()`.

```
  private void cropImage() {

    cropVisual = new Rectangle();
    cropVisual.setStroke(Color.WHITE);
    cropVisual.setStrokeWidth(2);
    cropVisual.setFill(Color.rgb(255, 255, 255, 0.3));
    Pane parentPane = (Pane) imageView.getParent();
    parentPane.getChildren().add(cropVisual);

    cropRect = new Rectangle2D(0, 0, image.getWidth(), image.getHeight());

    drawingCanvas.setOnMousePressed(event -> {
        cropRect = new Rectangle2D(event.getX(), event.getY(), 0, 0);
        cropVisual.setX(cropRect.getMinX());
        cropVisual.setY(cropRect.getMinY());
        updateCropVisual();
    });

    drawingCanvas.setOnMouseDragged(event -> {
        cropRect = new Rectangle2D(cropRect.getMinX(), cropRect.getMinY(),
event.getX() - cropRect.getMinX(), event.getY() - cropRect.getMinY());
        updateCropVisual();
    });
    drawingCanvas.setOnMouseReleased(event -> {
        drawingCanvas.setOnMousePressed(null);
        drawingCanvas.setOnMouseDragged(null);
        drawingCanvas.setOnMouseReleased(null);
        drawingCanvas.setPickOnBounds(false);

        if (cropRect.getWidth() > 0 && cropRect.getHeight() > 0) {
            finishCrop();
        } else {
            cancelCrop();
        }
    });
}
```

O altă metodă foarte importantă în acest sistem este posibilitatea de a adăuga detalii la dorință în orice locație, această posibilitate este realizată prin metoda `drawOnCanvas()`. Această metodă `drawOnCanvas()` care este folosită pentru a permite utilizatorului să deseneze pe un obiect Canvas dintr-o aplicație JavaFX. Metoda setează trei manipulatoare de evenimente pentru evenimentele de mouse pe

obiectul drawingCanvas pentru a gestiona acțiunile de desenare. Când utilizatorul apasă butonul mouse-ului, un nou desen începe prin apelarea metodei `beginPath()` pe obiectul graphicsContext. De asemenea, linia curentă este mutată la poziția curentă a mouse-ului folosind `moveTo()` și apoi trasată prin apelarea `stroke()`. Când utilizatorul trage mouse-ul, o nouă linie este trasată prin apelarea `lineTo()` pe obiectul graphicsContext pentru a conecta poziția curentă a mouse-ului cu ultima poziție de mouse. Linia este apoi trasată prin apelarea `stroke()`. Când utilizatorul eliberează butonul mouse-ului, desenul curent se închide prin apelarea `closePath()` pe obiectul graphicsContext. În final, codul curăță canvas-ul prin apelarea `clearRect()` pe obiectul graphicsContext pentru a șterge toate desenele existente de pe canvas înainte de a începe un nou desen.

```
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();
    });

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

Pentru lucrul cu filtrele se utilizează o implementare a interfeței Function. Aceasta funcție primește o imagine și aplică o transformare de culoare la fiecare pixel din imagine. Implementarea primește un obiect Image și returnează un nou obiect Image cu transformarea de culoare aplicată. La început se extrag dimensiunile imaginii originale folosind metodele getWidth() și getHeight() și se stochează în variabilele w și h. Se creează un nou obiect WritableImage cu aceleași dimensiuni ca și imaginea originală, care va fi folosit pentru a stoca imaginea transformată. În următoarele două bucle for, se parcurge fiecare pixel din imaginea originală. Pentru fiecare pixel, se obține culoarea curentă folosind metoda getPixelReader().getColor(x, y) a obiectului Image. Se aplică transformarea de culoare la această culoare utilizând obiectul colorMap care este o altă implementare a interfeței Function și se primește o nouă culoare. Noua culoare este apoi scrisă în obiectul WritableImage folosind metoda getPixelWriter().setColor(x, y, color). La final, metoda returnează noul obiect Image cu transformarea de culoare aplicată.

```
public Image apply(Image image) {
    int w = (int) image.getWidth();
    int h = (int) image.getHeight();

    WritableImage newImage = new WritableImage(w, h);

    for (int y = 0; y < h; y++) {
        for (int x = 0; x < w; x++) {
            Color c1 = image.getPixelReader().getColor(x, y);
            Color c2 = colorMap.apply(c1);

            newImage.getPixelWriter().setColor(x, y, c2);
        }
    }
    return newImage;
}
```

Un exemplu de filtru este filtrul Sepia, pentru acesta s-a utilizat o clasă nouă ce defineste un filtru Sepia pentru imagini. Constructorul clasei inițializează numele filtrului și definiește o funcție lambda care primește un obiect de tipul Color și returnează un alt obiect Color modificat în funcție de formula specifică pentru transformarea unei culori într-o nuanță Sepia. Mai exact, formula utilizată în funcția lambda transformă fiecare componentă a culorii (roșu, verde, albastru) prin combinarea lor ponderată și apoi limitând valoarea la 1.0 (pentru a se asigura că valoarea este întotdeauna în intervalul valid pentru o culoare). Apoi, noile valori ale fiecărei componente sunt utilizate pentru a crea și returna o nouă instanță de Color care reprezintă noua culoare Sepia.

```
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);
    });
}
```

Analogic cu filtrul Sepia este și filtrul Sharpen, acesta presupune o tehnică de procesare a unei imagini fotografice, care face posibilă obținerea unei creșteri a clarității subiective prin îmbunătățirea contrastului detaliilor mici, menținând în același timp un contrast general neschimbat. Pentru realizarea acestei funcții s-a utilizat matrice 3x3. Cele două bucle for exterioare iterează peste toate pixelii imaginii, în timp ce cele două bucle for interioare iterează peste toate elementele matricei. Pentru fiecare pixel, codul calculează o valoare nouă de culoare pe baza sumei ponderate a culorilor pixelilor vecini, conform valorilor din matricea. Valorile de culoare rezultate sunt apoi fixate între 0 și 1 și utilizate pentru a crea o nouă imagine. Valorile din matrice controlează puterea și direcția efectului de filtrare și pot fi ajustate pentru a obține diferite rezultate. Implementarea actuală utilizează o matrice fixă, ceea ce limitează flexibilitatea acesteia, dar permite o execuție mai rapidă.

```
for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++) {
        double red = 0, green = 0, blue = 0;
        for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
                int nx = x + i;
                int ny = y + j;
                if (nx >= 0 && nx < w && ny >= 0 && ny < h) {
                    Color color = image.getPixelReader().getColor(nx, ny);
                    red += color.getRed() * kernel[i+1][j+1];
                    green += color.getGreen() * kernel[i+1][j+1];
                    blue += color.getBlue() * kernel[i+1][j+1];
                }
            }
        }
        red = Math.min(Math.max(red, 0), 1);
        green = Math.min(Math.max(green, 0), 1);
        blue = Math.min(Math.max(blue, 0), 1);
        Color newColor = Color.color(red, green, blue);
        newImage.getPixelWriter().setColor(x, y, newColor);
    }
}
```

3.2 TESTAREA SISTEMULUI

Pentru testarea aplicației s-a utilizat testarea manuală și unit teste. Testarea manuală este un proces prin care se explorează manual o aplicație sau un sistem software pentru a găsi defecte, erori și alte probleme de calitate. Aceasta implică executarea de scenarii de testare, introducerea de date și verificarea funcționalității aplicației sau sistemului într-un mod interactiv și fără utilizarea instrumentelor automate. Testarea manuală poate fi efectuată atât în faza de dezvoltare, cât și după lansarea aplicației sau a sistemului software. Scopul este de a asigura că produsul este funcțional și de înaltă calitate, conform specificațiilor și cerințelor de utilizare. Testarea manuală este importantă deoarece detectează defectele care nu pot fi găsite cu instrumentele automate sau care necesită evaluarea umană. Testarea bazată pe unitate este o metodă de testare software care implică testarea individuală a fiecărei unități componente ale unei aplicații sau a unui sistem software. Unitatea poate fi o funcție, o metodă, o clasă sau un modul. Testarea unitară se concentrează pe verificarea corectitudinii funcționale și a comportamentului unității testate, indiferent de contextul în care este utilizată.

Testarea manuală s-a efectuat în 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 - această etapă 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 gestionarii erorilor aplicației, furnizând intenționat intrări incorecte și verificând modul în care aplicația răspunde.

Pentru testarea bazată pe unit-teste s-au creat teste care verifică 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);
}
```

În urma rulării testului s-a obținut rezultatul arătat în figura 3.2 ceea ce înseamnă ca funcționalul aplicației care răspunde de rotirea unei imagini funcționează corect, rezultatul așteptat fiind egal cu cel obținut.

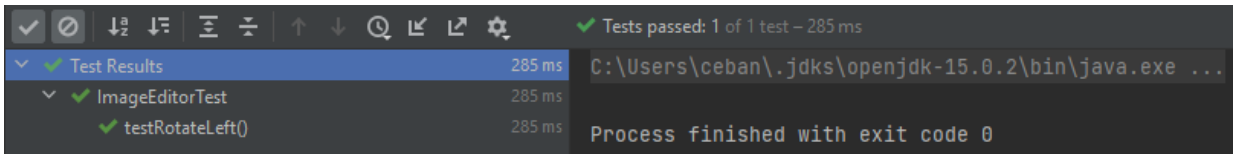

Figura 3.2 – Rezultatul executării testului

4 DOCUMENTAREA PRODUSULUI REALIZAT

Pentru rularea aplicației pe PC-ul personal el trebuie sa îndeplinească cerințele minime pentru ca aplicația sa funcționeze cu succes. Cerințele hardware și soft pentru utilizarea aplicației sunt:

- procesor – Intel Core i3 sau AMD Ryzen 3;
- sistem de operare – Windows 10 (64bit, version 1909);
- GPU:
  Nvidia Quadro M2000 AMD;
  Radeon RX 580;
- RAM – 4 GB;
- VRAM – 2GB;
- monitor – color.

Descărcarea inițială a aplicației are loc prin accesarea repozitoriului de pe Github și clonarea acestuia pe repozitoriul local. După clonare proiectul se deschide cu ajutorului unui IDE, de exemplu IntellijIDEA.

Rularea aplicației are loc prin efectuarea unui click pe simbolul indicat in Figura 4.1 sau prin apăsarea concomitenta a combinației Shift+F10.

Figura 4.1 – butonul pentru rulare

După pornirea aplicației se deschide interfața principala, ea consistă din meniul principal și zona de editare. Modul în care arată aplicația este arătat in Figura 4.2.

Figura 4.2 – Interfața principală

Meniul principal consistă din 3 elemente principale, ele oferă accesul la tot funcționalul aplicației. Descrierea meniului este indicată în Figura 4.3. Fiecare secvență din meniu la accesare v-a deschide un sub meniu.

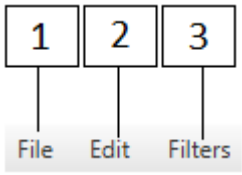

Figura 4.3 – Meniul principal

După cum este indicat în Figura 4.3 meniul consta din 3 părți principale. Fiecare compartiment răspunde de un funcțional aparte, el fiind împărțit în modul următor:

1 – File, se efectuează salvarea si deschiderea unui fișier;

2 – Edit, accesul la funcționalul ce permite editarea unei imagini;

3 – Filters, accesul la lista de filtre.

Pentru deschiderea unei imagini noi se accesează punctul 1 din meniu, după care se alege subpunctul 1 cu denumirea *open.* Cum arată acest meniu este indicat in Figura 4.4.

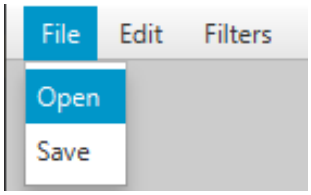

Figura 4.4 – Sub meniu File

După efectuarea unui click pe buton *open* se deschide o fereastră nouă unde se poate selecta imaginea care poate fi încărcată. Fereastra ce se deschide este indicată în Figura 4.5.

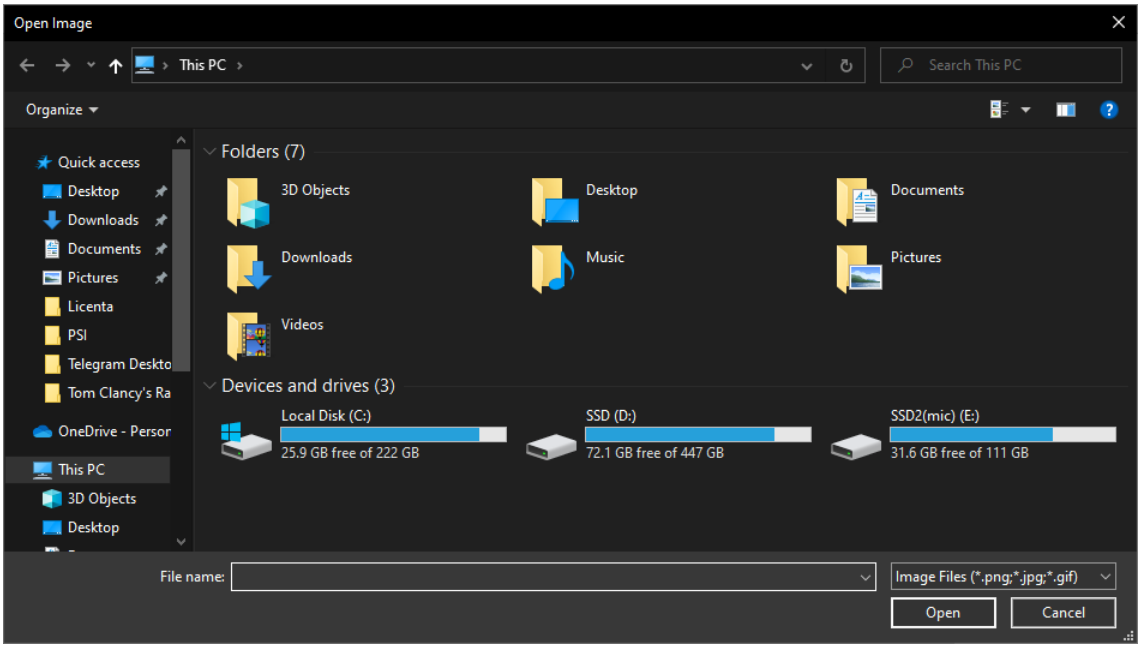

Figura 4.5 – Fereastra selectare fișier

Salvarea unui fișier editat are loc în același mod cum are loc salvarea, este accesat meniul *File*, după care se selectează *Save* și se indică locația.

Pentru rotirea unei imagini noi se accesează punctul 2 din meniu, după care se alege subpunctul 1 sau 2 cu denumirea *rotate left* sau *rotate right.* Rotirea unei imagini presupune schimbarea unghiului de înclinare cu 90 grade în direcția dorită, astfel putem obține o imagine nouă. Cum arată acest meniu este indicat in Figura 4.6.

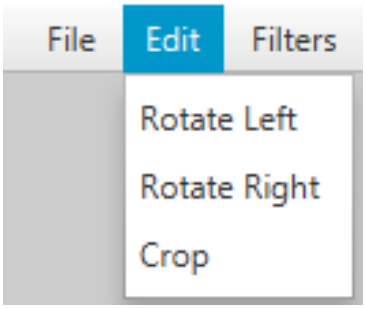

Figura 4.6 – Sub meniu Edit

Decuparea unei imagini are loc analogic cu rotirea, în sub meniu se alege opțiunea *crop* după care se selectează partea necesara, cum arata partea selectată este indicat in Figura 4.7

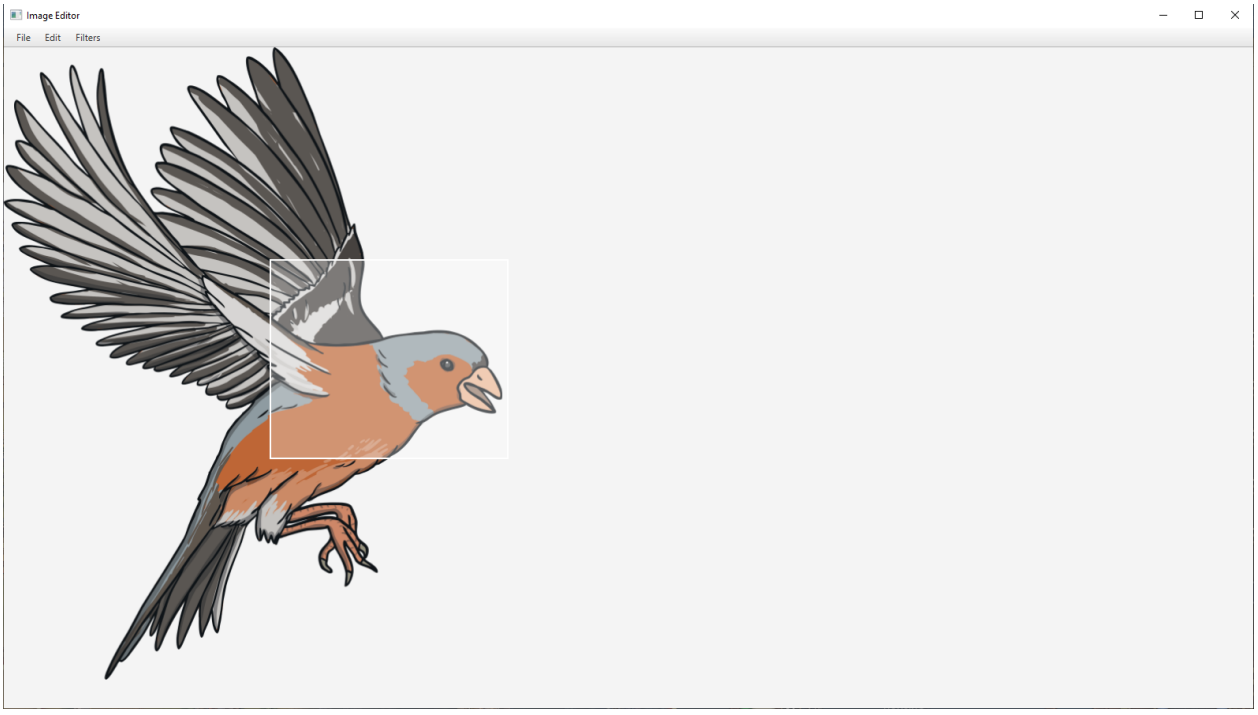

Figura 4.7 – Zona selectată pentru decupare

Pentru aplicarea unui filtru trebuie accesat subpunctul 3 din meniu după care din lista de filtre se selectează filtrul dorit ce urmează a fi aplicat. Cum arată acest sub meniu și filtrele disponibile sunt

indicate în Figura 4.8. Fiecare filtru ce urmează a fi aplicat v-a înlocui filtrul precedent sau v-a fi adăugat adițional asupra lui. Aceasta funcție depinde de filtru în parte.

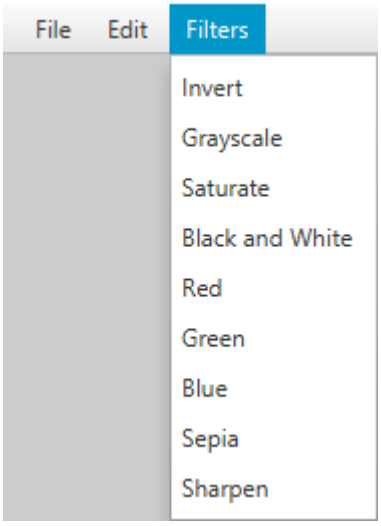

Figura 4.8 – Lista de filtre

5 ESTIMAREA COSTURILOR PROIECTULUI

Sistemul propus spre implementare are drept scop implementarea unei aplicații simple ce va utiliza Soft-uri diferite cu scopul final de obținere a unei aplicații gratis. Cheltuielile esențiale v-or fi pe întreținerea serverului de unde se poate descărca aplicația și pe lucrul pentru elaborarea ei. Pentru a dezvolta sistemul informațional propus este necesar de elaborat și de stabilit sarcinile prioritare ale proiectului, pentru aceasta s-a elaborat diagrama WBS. WBS este un instrument pentru împărțirea tuturor activităților proiectului în: pachete de lucru gestionabile, definite, care permit atingerea unui nivel de detaliu al informațiilor furnizate care răspund nevoilor proiectului.

În figura 5.1 este reprezentată Structura decompoziției lucrărilor (WBS), care este un grafic ce prezintă rezultatele și componentele proiectului, este folosit pentru a oferi claritate cu privire la ceea ce proiectul trebuie să furnizeze și in ce ordine. Realizarea acestui proiect este împărțită in 5 etape: gestiunea, definirea cerințelor, elaborarea, testarea si finalizarea.

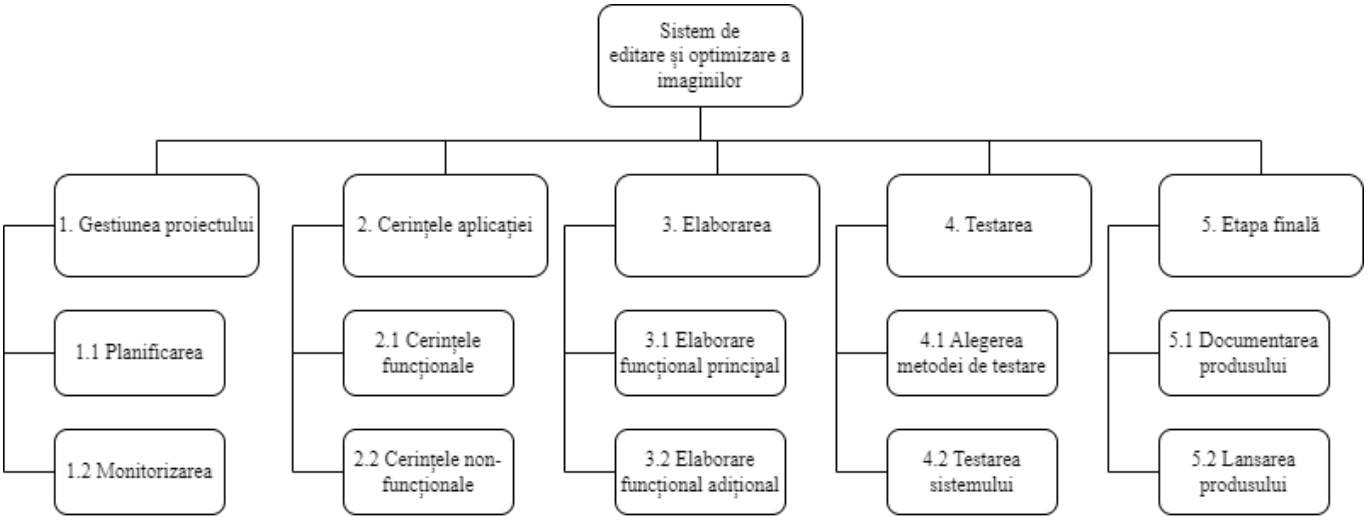

Figura 5.1 – Decompoziția lucrărilor

Graficul Gantt este o metodă de planificare și urmărire a proiectelor, care constă într-un diagramă cu bare care reprezintă sarcinile și activitățile proiectului, dispuse pe axa orizontală a timpului. Fiecare bară indică durata activității și perioada în care aceasta se desfășoară. Importanța graficului Gantt în managementul proiectelor constă în faptul că oferă o imagine de ansamblu asupra tuturor activităților care trebuie efectuate, perioadele de timp necesare pentru a le finaliza și ordinea în care acestea trebuie efectuate. Acest lucru facilitează planificarea și urmărirea proiectului, îmbunătățind eficiența și eficacitatea managementului proiectului. Graficul Gantt poate fi utilizat pentru a identifica sarcinile critice ale proiectului, adică acele sarcini care trebuie finalizate la timp pentru a asigura finalizarea proiectului la termenul stabilit. De asemenea, poate fi folosit pentru a identifica sarcinile care sunt dependente unele de altele, astfel încât managerul de proiect să poată coordona eforturile echipei în mod corespunzător. În

aceasta lucrare am împărțit termenul de lucru in 5 etape distincte cu un termen finit de timp, fiecare etapa având câteva sub etape care sunt incluse in termenul limita. În figura 5.2 am reprezentat acest grafic ce cuprinde limitele de timp și sarcinile corespunzătoare care urmează a fi îndeplinite.

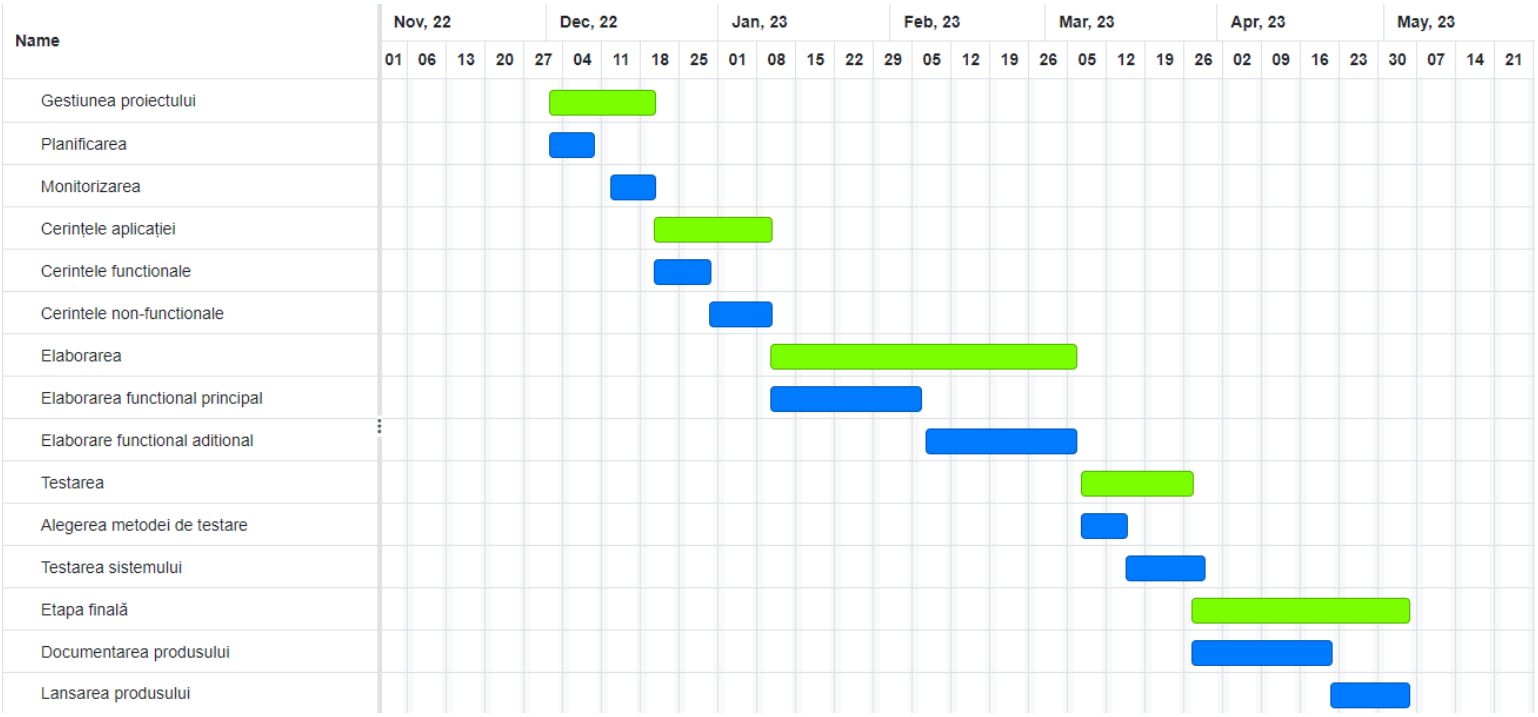

Figura 5.2 – Graficul Gantt al sistemului

Graficul Gantt este o unealtă valoroasă în managementul proiectelor, deoarece oferă o imagine de ansamblu clară și detaliată asupra activităților și sarcinilor implicate în proiect, ajutând managerul de proiect să planifice, să coordoneze și să urmărească cu succes proiectul.

Cheltuielile esențiale v-or fi pe întreținerea serverului unde se poate descărca aplicația și pe întreținerea viitoare a aplicației. Cheltuielile pentru arendarea unui server sunt indicate în Figura 5.3.

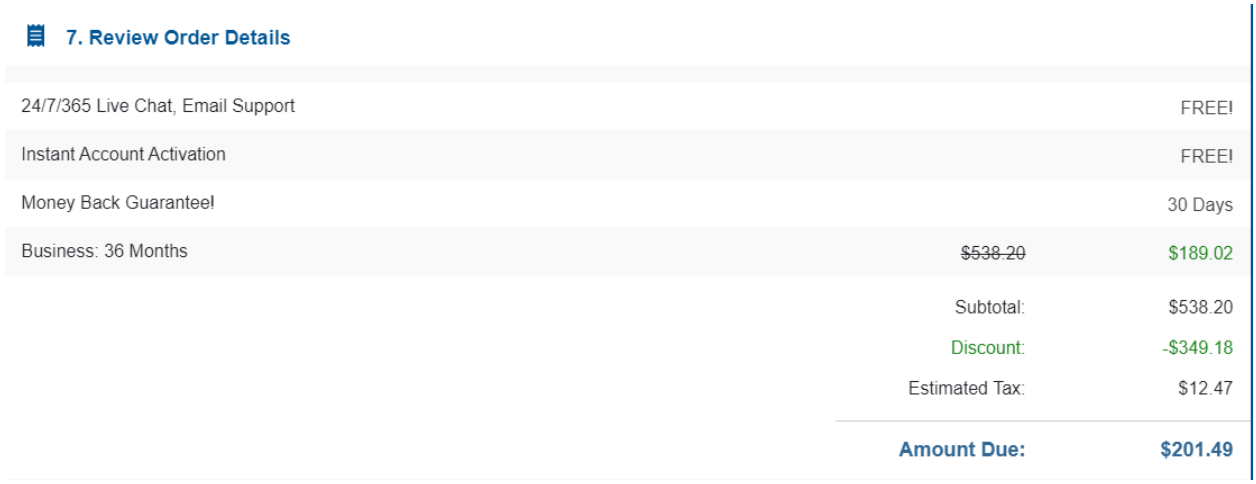

Figura 5.3 - Estimarea costului total pentru arendarea unui server.

Arendarea serverului reprezintă doar o parte din cheltuieli. O alta secțiune este depozitarea aplicației finale pe un server, costul acestei acțiuni este arătat în Figura 5.4. Arendarea lunara este obligatoriei deoarece aici este stocată aplicația și de unde poate fi accesată ea pentru a fi descărcată. Prețul pentru un an depășește suma de 30$.

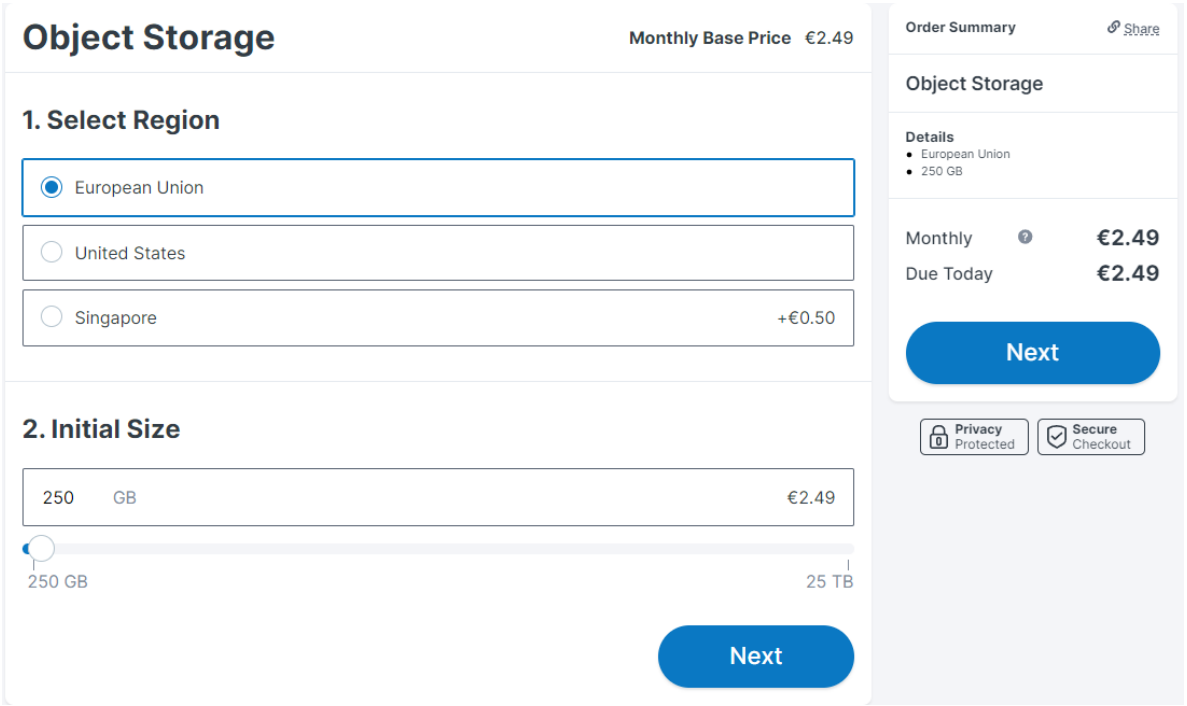

Figura 5.4 - Estimarea costului serviciului de stocare.

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 persoana 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 sa 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 tranzacție 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 vizuala a fluxurilor s-a realizat o diagrama de activitate unde este descris un proces din cadrul aplicației. Stările de tranzacție 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 utilizarea a aplicației. Scenariile de utilizare ajuta 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 clasa, pentru aceasta a fost realizată diagramă de clasă a aplicației unde au fost indicate clasele principale, relațiile între ele si 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.

Capitolul patru este destinat documentarii produsului, în acest capitol au fost realizate instrucțiuni cu privire la utilizarea aplicației. S-a descris cum aplicația poate fi instalată pe PC-ul personal, de asemenea au fost indicate funcționalitățile principale și cum ele pot fi accesate. Pentru fiecare funcționalitate aparte au fost indicați pașii ce urmează a fi efectuați pentru efectuarea acesteia.

Ultimul capitol este dedicat estimării costurilor pentru elaborarea unui astfel de proiect, în acest capitol au fost descrise toate cheltuielile ce urmează a fi efectuate în procesul elaborării dar și create diagramele Gantt și WBS pentru sistemul dat.

BIBLIOGRAFIE

1. Adriana Bogdan, ISTORIA PRIN APLICAŢII MULTIMEDIA; [online] – Disponibil: http://www.historica-cluj.ro/anuare/AnuarHistorica2014/24.pdf

2. Grafică digitală; [online] – Disponibil: https://ro.wikipedia.org/wiki/Grafic%C4%83_digital%C4%83

3. An introduction to raster images; [online] – Disponibil: https://shorthand.com/the-craft/raster-images/index.html

4. What is image compression and how does it work; [online] – Disponibil: https://www.techtarget.com/whatis/definition/image-compression

5. Desktop Applications Vs. Web Applications; [online] – Disponibil: https://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html

6. Pros and Cons of Desktop Apps; [online] – Disponibil: https://outsourcenz.com/pros-and-cons-of-web-apps-and-desktop-apps/

7. Best Photo Editing Software; [online] – Disponibil: https://www.softwaretestinghelp.com/best-free-photo-editing-software/

8. History of Photoshop; [online] – Disponibil: www.designbyfire.com/pdfs/history_of_photoshop.pdf