UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA:Tehnologia informației

iMPACtViewer

Proiect la disciplina
Ingineria programării (IP)

Studenți
Paul-Cristian Brînză
Andrei-Iulius Covaș
Mircea-Constantin Dobreanu

Iași, 2019

Cuprins

1. Specificarea cerințelor (SRS)

1.1 Introducere

1.1.1 Scopul documentului

Scopul documentului este acela de a prezenta specificațiile produsului software cu numele iMPACt Viewer. Acest produs are ca funcționalitate principală, posibilitatea vizualizării imaginilor cu diferite extensii(.jpg, .bmp, .gif, etc.) stocate pe un anumit computer.

1.1.2 Convențiile documentului

Documentul respectă standardul de formatare stabilit de IEEE pentru dezvoltarea de produse software.

1.1.3 Audiența țintă şi sugestii de citire

Documentul se adresează oricărei persoane ce dorește să se informeze cu privire la specificațiile produsului software dezvoltat. Acesta conține o parte de introducere, descrierea generală a produsului în care sunt indicate funcționalitățile și caracteristicile produsului, cerințele externe ale interfeței, caracteristici de sistem, precum și cerințele aplicației.

1.1.4 Scopul produsului

Scopul produsului software iMPACt Viewer și domeniul de aplicare al acestuia se regăsesc în documentația programului.

1.1.5 Referințe

Acest document SRS are la bază informații despre faza de analiză în dezvoltarea unui produs software, din cursul dr. Florin Leon, profesor universitar al Facultății de Automatică și Calculatoare, Iași, http://florinleon.byethost24.com/Curs_IP/IP03_Analiza.pdf

1.2 Descriere generală

1.2.1 Perspectiva Produsului

Produsul iMPACt Viewer a fost dezvoltat din dorința de a putea oferi oricărui utilizator al unui computer, posibilitatea vizualizării colecției personale de fotografii într-un mod util și plăcut, idee susținută prin prezența unei interfețe simpliste dar în același timp intuitive.

1.2.2 Funcțiile produsului

Aplicația pune la dispoziția utilizatorului următoarele funcționalități principale:

- Posibilitatea indicării unui director ce conține imagini(*.jpg, *.bmp, *gif, etc.)
- Navigarea prin lista de imagini existente în directorul indicat
- Posibilitatea rotirii imaginii curente, aflată pe panel-ul de afișare

1.2.3 Utilizatori și caracteristici

Entitatea principală a acestei aplicații, prezentă și în diagrama cazurilor de utilizare (Figura 2.1: Diagrama de use-case) este utilizatorul, căruia îi sunt permise toate facilitățile oferite de aplicație.

1.2.4 Mediul de operare

Aplicația ce permite vizualizarea imaginilor dintr-un director ales, folosește platforma .NET de la Microsoft și este concepută pentru a rula pe sistemul de operare Windows.

1.2.5 Design și constrângeri de implementare

Design-ul aplicației este unul minimalist, cu o interfață ușor de interacționat care oferă facilitatea principală a programului, aceea de a vizualiza fișiere cu extensii specifice imaginilor.

1.2.6 Documentația utilizatorului

Documentația utilizatorului constă dintr-un help integrat în cadrul aplicației. Acesta conține toate informațiile necesare pentru utilizarea aplicației.

1.2.7 Ipoteze și dependințe

Aplicația dezvoltată care face obiectul acestui document, este dependentă de platformă, ea putând fi lansată în execuție numai de pe sistemul de operare Windows.

1.3 Cerințele interfeței externe

1.3.1 Interfața cu utilizatorul

Interfața cu utilizatorul este alcătuită dintr-un meniu, în partea superioară a ferestrei cu câmpurile „File”, „Edit” și „Help”, dintr-un panel central în care vor fi afișate imaginile precum și din butoanele „Browse” pentru încărcarea listei de imagini, „Next”, „Previous” pentru navigarea prin listă și două butoane pentru funcția de rotație.

1.3.2 Interfețele hardware

Aplicația poate rula pe orice computer cu o configurație hardware de funcționare, dar care are instalată platforma .NET oferită de Microsoft.

1.3.3 Interfețele software

Pentru crearea aplicației a fost utilizat mediul de dezvoltare Microsoft Visual Studio 2015, ce pune la dispoziție framework-ul .NET pentru realizarea aplicațiilor de tip Windows Forms, un API grafic pentru afișarea datelor și gestiunea interacțiunilor utilizatorilor.

1.3.4 Interfețele de comunicare

Aplicația preia datele de pe mediul de stocare al computerului prin listarea tuturor fișierelor aflate în directorul indicat anterior, de către utilizator.

1.4 Caracteristicile de sistem

Caracteristicile sistemului sunt prezentate pe larg în capitolul intitulat Utilizarea aplicației (Modul de utilizare; help).

1.4.1 Vizualizarea imaginilor

1.4.1.1 Descriere și prioritate

Această caracteristică a sistemului are prioritatea cea mai ridicată, fiind principalul scop al aplicației dezvoltate.

1.4.1.2 Secvențele Stimul/Răspuns

- Click Browse/Open din meniu: deschiderea unei ferestre de dialog specializată, ce are rolul de alege fișiere sau directoare
- Click Help din meniu: deschiderea unei noi ferestre cu instrucțiunile pentru utilizatori
- Click Next/Previous: navigarea prin lista de imagini
- Click Rotate left/right: efectuarea rotației, aplicată imaginii, la stânga sau la dreapta cu câte 90°

1.4.1.3 Cerințe funcționale

- REQ-1: Selectarea unui anumit folder din fereastra de dialog afișată în urma apăsării butonului Browse este necesară pentru afișarea unei imagini în panel-ul specializat.
- REQ-2: Apăsarea butoanelor de rotire și a celor de navigare, au ca efect afișarea unor Message Box-uri cu un mesaj descriptiv.

1.5 Alte cerințe nefuncționale

1.5.1 Cerințe de performanță

Aplicația își propune ca cerințe de performanță posibilitatea încărcării unei imagini într-un timp rezonabil de așteptare, mai mică decât 300-400 ms.

1.5.2 Cerințe de siguranță

Principala cerință de siguranță este aceea a siguranței datelor, prin eventuala existență a riscurilor de deteriorare a datelor accesate cu ajutorul produsului descris.

1.5.3 Cerințe de securitate

Funcționarea corectă a aplicației depinde de asemenea, de drepturile de acces asupra imaginilor existente în partițiile de lucru al computerului utilizat, pentru a lansa aplicația în execuție.

1.5.4 Atribute ce țin de calitatea software

Principalele calități ale produsului software dezvoltat sunt extensibilitatea, corectitudinea, operabilitatea și testabilitatea.

1.5.5 Reguli de afaceri

Orice utilizator poate accesa imaginile existente pe mediul de stocare, atât timp cât sistemul de operare dispune de accesul de citire al fișierelor respective.

2. Proiectarea aplicației (șablon, UML)

În proiectarea aplicației am folosit șablonul Model View Presenter(MVP) fiindcă este un șablon cu o funcționalitate ușor de înțeles și totodată aplicabil pe scopul final al aplicației alese pentru dezvoltare. De asemenea, vizualizarea pasivă folosită la actualizarea modulului View al aplicației, reprezintă o metodă vaforabilă testabilității aplicației.

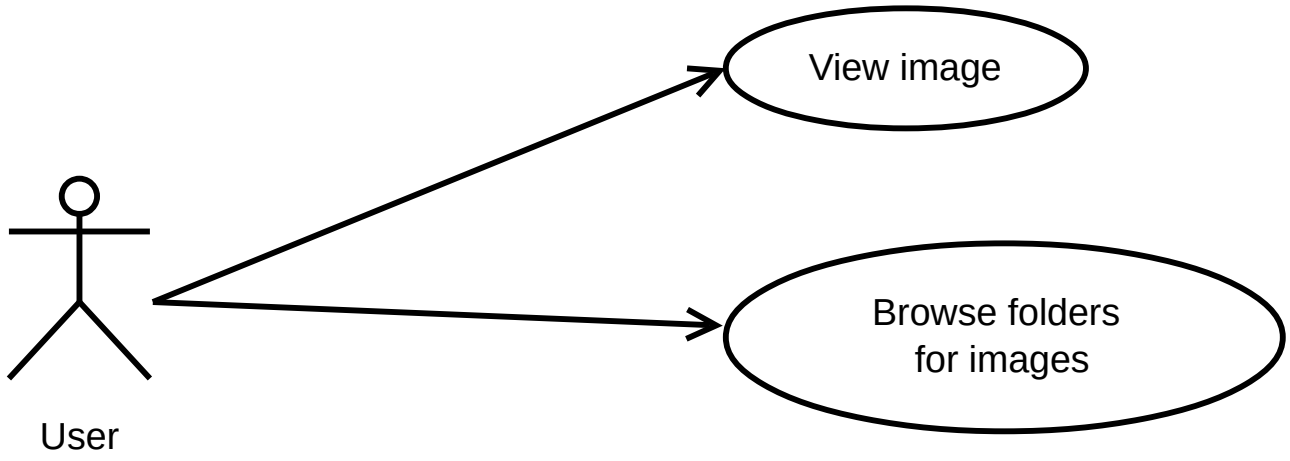

Figura 2.1: Diagrama de use-case

După cum se poate vedea și în Figura 2.1 scopul aplicației este de a expune imaginile disponibile pe sistemul de fișiere al user-ului.

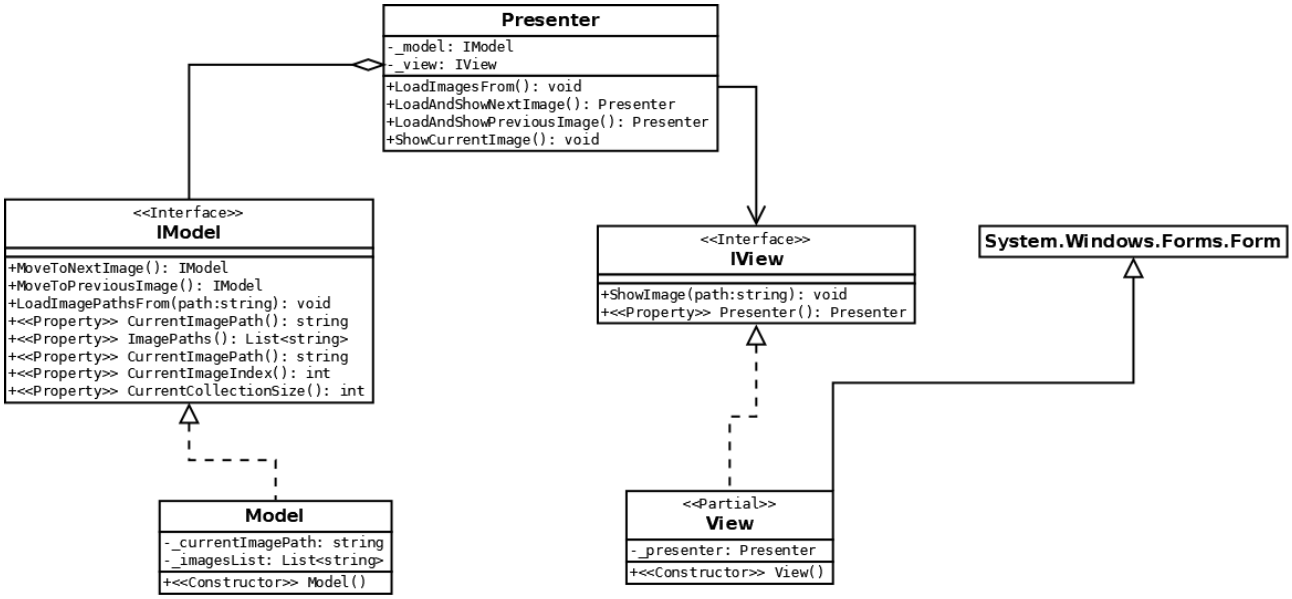

Figura 2.2: Diagrama de clase

După cum este ilustrat și în Figura 2.2 clasa Model este responsabilă de menținerea stării (în cazul concret prezentat, modelul reprezintă un director care poate conține sau nu imagini). Presenter-ul se ocupă cu tranziția dintr-o stare într-alta, adică manipulează modelul. De asemenea, Presenter-ul actualizează informațiile afișate de către clasa View. Clasa View reprezintă interfața cu utilizatorul și trimite mai departe către Presenter cererile utilizatorului (deschide un director, afișează o imagine, etc).

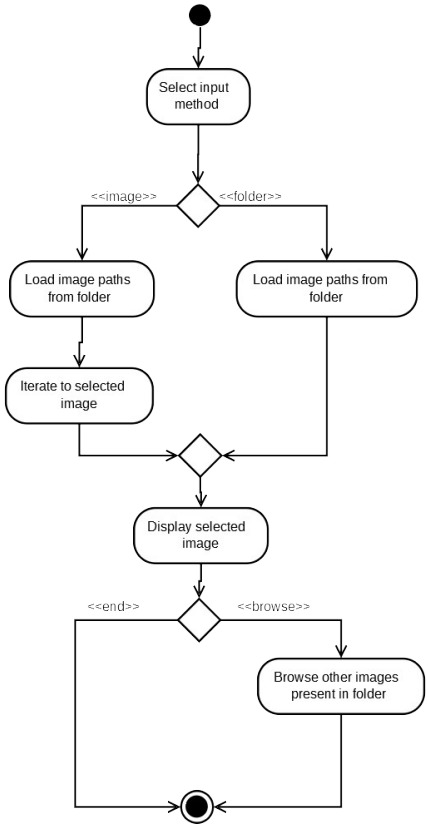

Figura 2.3: Diagrama de activitate

În Figura 2.3 sunt ilustrați pașii parcurși prin rularea aplicației în funcție de alegerile utilizatorului.

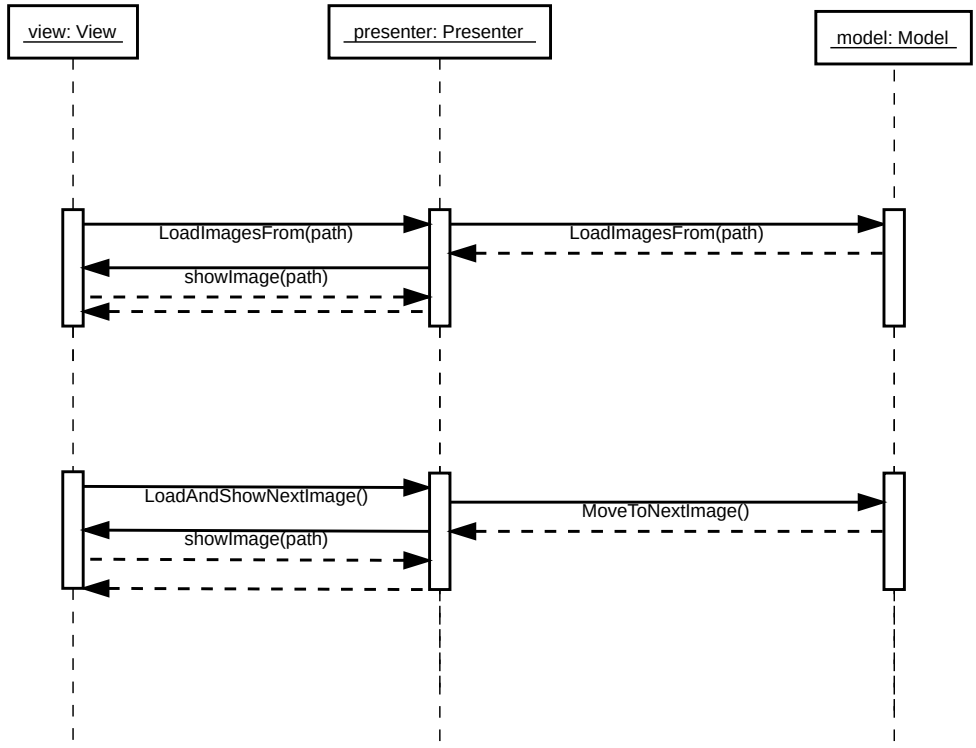

Figura 2.4: Diagrama de secvență

În Figura 2.4 se ilustrează în mod dinamic interacțiunea dintre cele trei componente menționate anterior. O cerere a utilizatorului (o interacțiune cu view-ul) determină o cerere de la view la Presenter, care modifică starea curentă și apoi propagă schimbările către View.

3. Utilizarea aplicației (Modul de utilizare; help)

Cerințe de sistem

Aplicația poate rula pe orice computer cu o configurație hardware de funcționare, dar care are instalată platforma .NET oferită de Microsoft.

Funcționalitate

Aplicația pune la dispoziția utilizatorului următoarele funcționalități principale:

1. Posibilitatea indicării unui director ce conține imagini (*.jpg, *.bmp, *.gif, etc.)
2. Navigarea prin lista de imagini existente în directorul indicat
3. Posibilitatea rotirii imaginii curente, aflată pe panel-ul de afișarea

Componente

Interfața cu utilizatorul este alcătuită dintr-un meniu, în partea superioară a ferestrei cu câmpurile File și Help, dintr-un panel central în care vor fi afișate imaginile precum și din butoanele Browse pentru încărcarea listei de imagini, Next, Previous pentru navigarea prin listă și două butoane pentru funcția de rotație.

Meniu

Figura 3.1: Captură de ecran cu meniul aplicației

Panel de afișare

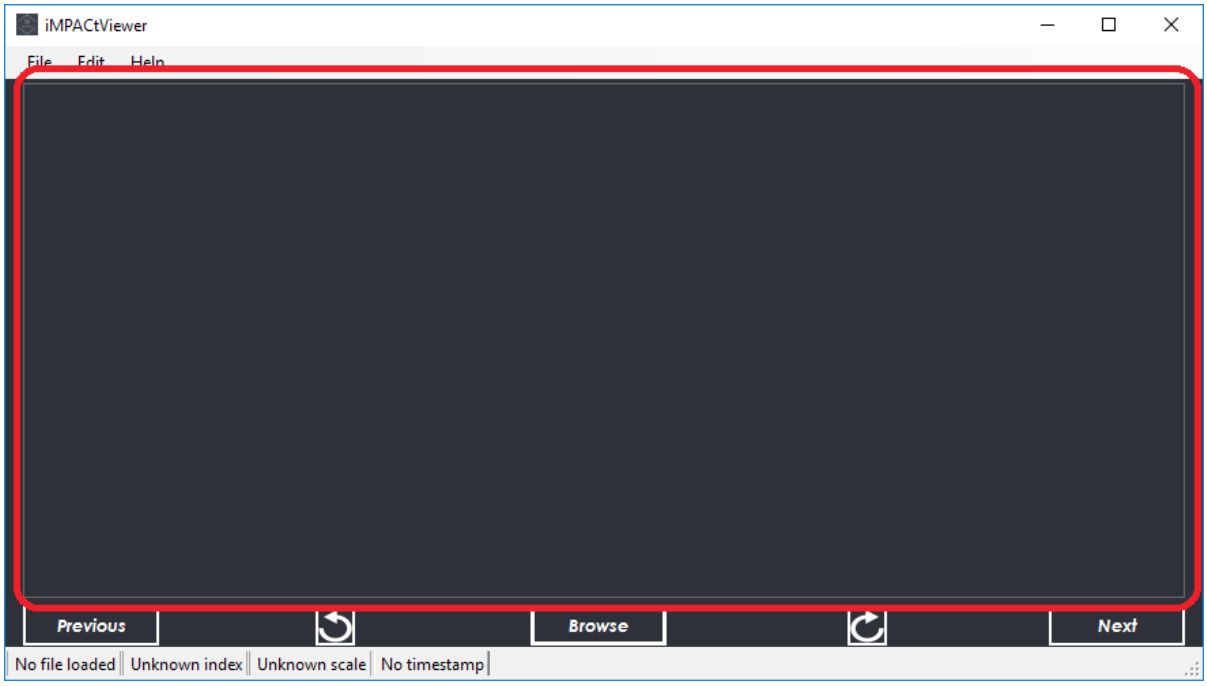

Figura 3.2: Captură de ecran cu panel-ul de afișare

Butoane

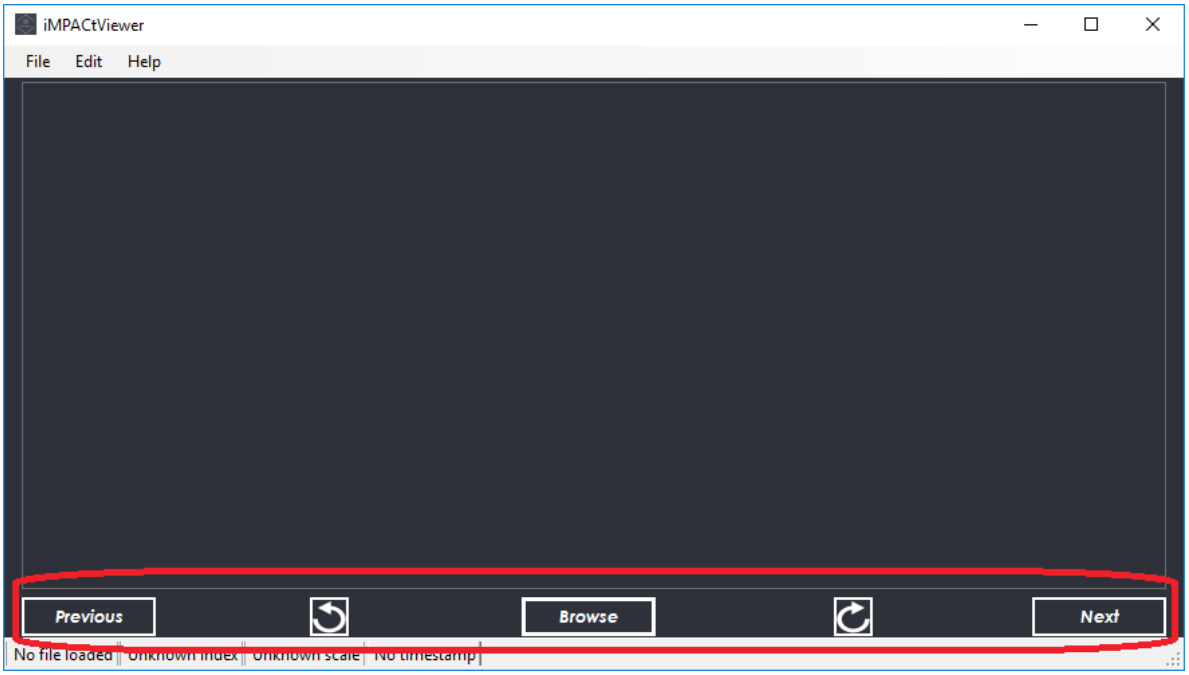

Figura 3.3: Captură de ecran cu butoanele aplicației

Detaliile imaginii afișate

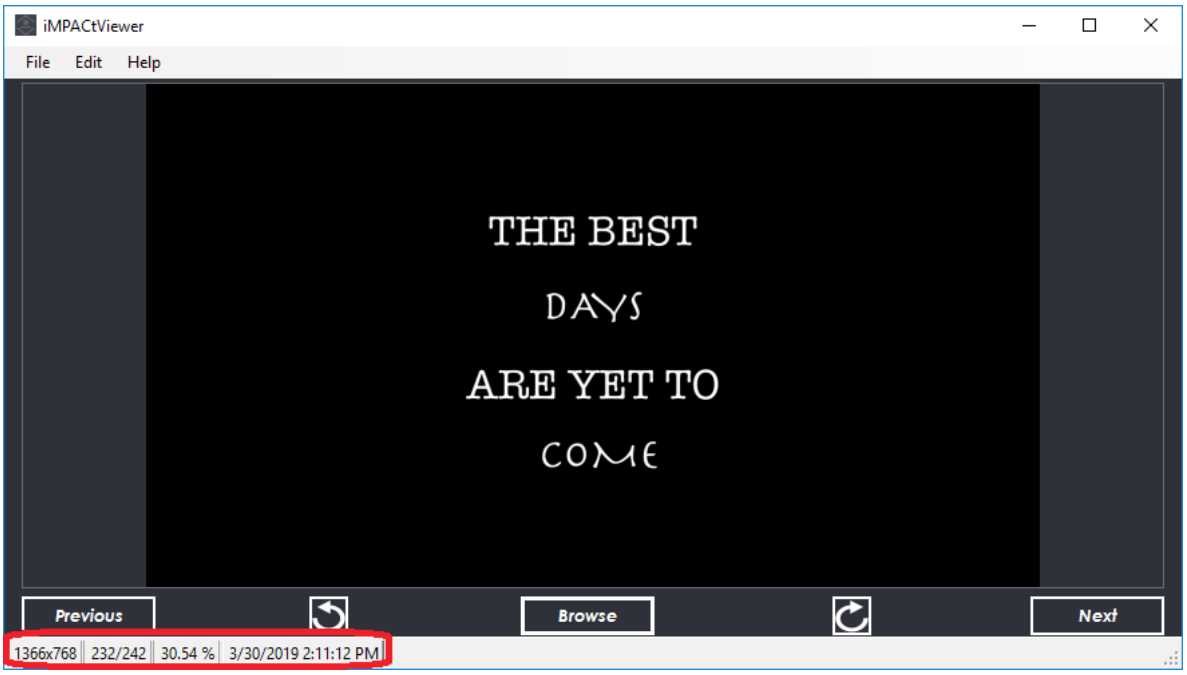

Figura 3.4: Captură de ecran cu detaliile imaginii

Utilizare

- *Open:*

Deschiderea unei anumite imagini dintr-un folder se realizează prin selectarea opțiunii Open din meniul aplicației.

- *Open folder:*

Deschiderea unei liste de imagini se realizează prin selectarea opțiunii Open Folder din meniul aplicației.

- *Help:*

Deschiderea manualului cu instrucțiunile pentru utilizatori se realizează prin selectarea opțiunii Help»View Help din meniul aplicației.

- *Browse:*

Deschiderea unei ferestre de dialog specializată, ce are rolul de alege fișiere sau directoare.

- *Next:*

Afișarea pe panel-ul de afișare a imaginii următoare din lista de imagini.

- *Previous:*

Afișarea pe panel-ul de afișare a imagini anterioare din lista de imagini.

- *Rotate left/right:*

Efectuarea rotației, aplicată imaginii, la stânga sau la dreapta cu câte 90°.

4. Exemple de rulări

Mai jos vor fi expuse câteva capturi cu programul în rulare:

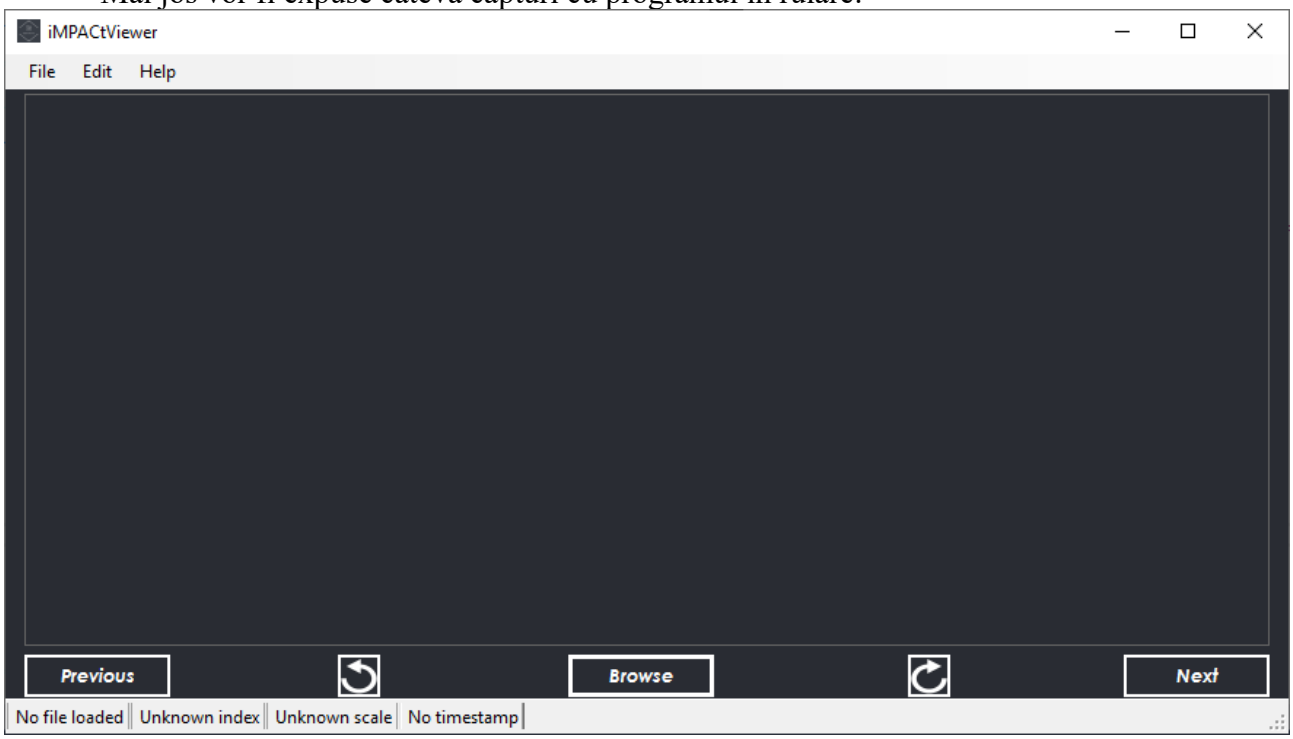

Figura 4.1: Programul când nu s-a încărcat nimic

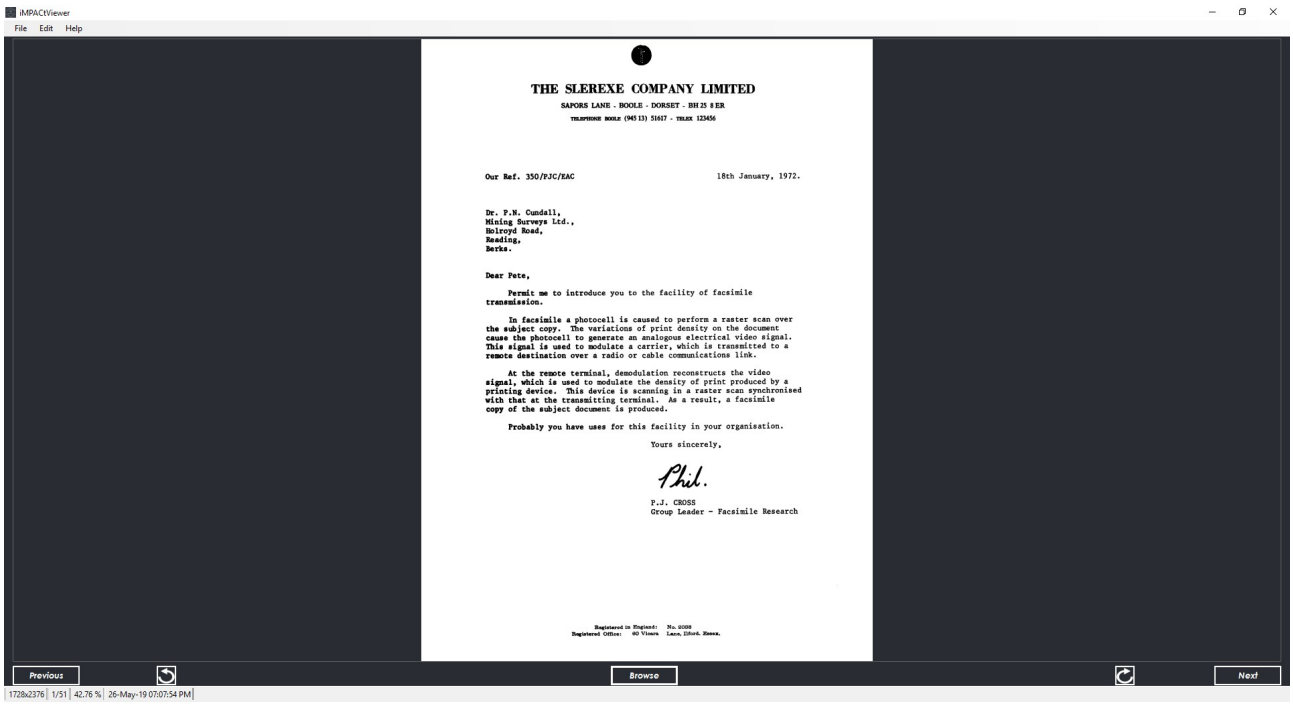

Figura 4.2: Folder cu 51 de imagini și un .Tiff expus

Figura 4.3: Programul după ce s-a iterat prin imagini (.jpg)

Anexe.

Fragmente semnificative ale codului sursă

Model.cs

```
/**
   * <summary>This function loads images from a path.</summary>
   */
  public void LoadImagePathsFrom(string path)
  {
      List<string> filePaths = Directory.GetFiles(path).ToList();
      LoadImagePathsFrom(filePaths);
  }

  public void LoadImagePathsFrom(List<string> filePaths)
  {
      _imagePathsList = new List<string>();
      CurrentImagePath = null;

      foreach (string filePath in filePaths)
      {
          if (IsImage(filePath))
          {
              _imagePathsList.Add(filePath);
              if (CurrentImagePath == null)
              {
                  CurrentImagePath = filePath;
              }
          }
      }
  }

  /**
   * <summary>This function specifies if a path points to an image.</summary>
   */
  public static bool IsImage(string path)
  {
      if (path == null)
      {
          return false;
      }

      try
      {
          string extension = Path.GetExtension(path);
          return ImageExtensions.Contains(extension.ToLowerInvariant());
      }
      catch (ArgumentException)
      {
          return false;
      }
  }

  /**
   * <summary>This function moves the iterator to the next image.</summary>
   */
  public IModel MoveToNextImage()
  {
```

```
            int index = -1;
            bool found = false;

            while (index < _imagePathsList.Count && found != true)
            {
                index++;

                if (index < _imagePathsList.Count &&
_imagePathsList[index].Equals(_currentImagePath))
                {
                    index++;
                    index %= _imagePathsList.Count;

                    _currentImagePath = _imagePathsList[index];
                    found = true;
                }
            }

            return this;
        }

        /**
         * <summary>This function moves the iterator to the previous image.</summary>
         */
        public IModel MoveToPreviousImage()
        {
            int index = _imagePathsList.Count;
            bool finded = false;

            while (index > 0 && finded != true)
            {
                index--;

                if (_imagePathsList[index].Equals(_currentImagePath))
                {
                    index--;
                    if (index < 0)
                    {
                        index = _imagePathsList.Count - 1;
                    }
                    _currentImagePath = _imagePathsList[index];
                    finded = true;
                }
            }

            return this;
        }
```

Presenter.cs

```
public class Presenter
    {
        private readonly IView _view;
        private IModel _model;

        public Presenter(IView view, IModel model)
        {
            if (view == null || model == null)
                throw new ArgumentNullException("Null argument passed to presenter
constructor.");
```

```
        this._view = view;
        this._model = model;

        this._view.Presenter = this;
    }

    public string CurrentImagePath {
        get { return this._model.CurrentImagePath; }
    }


    /**
     * <summary>
     * Function responsible with loading a folder.
     * </summary>
     */
    public Presenter LoadImagesFrom(string path)
    {
        _model.LoadImagePathsFrom(path);
        ShowCurrentImage();
        return this;
    }

    /**
     * <summary>
     * Convenience function responsible with displaying the next image.
     * </summary>
     */
    public Presenter LoadAndShowNextImage()
    {
        _model.MoveToNextImage();
        ShowCurrentImage();
        return this;
    }

    /**
     * <summary>
     * Convenience function responsible with displaying the previous image.
     * </summary>
     */
    public Presenter LoadAndShowPreviousImage()
    {
        LoadNextImage();
        ShowCurrentImage();
        return this;
    }
    public Presenter LoadNextImage()
    {
        LoadPreviousImage();
        return this;
    }

    public Presenter LoadPreviousImage()
    {
        _model.MoveToPreviousImage();
        return this;
    }

    public void ShowCurrentImage()
```

```
        {
            _view.ShowImage(_model.CurrentImagePath);
        }

        public string GetCurrentImagePositionInCollection()
        {
            return String.Format("{0}/{1}", 1 + _model.CurrentImageIndex,
_model.CurrentCollectionSize);
        }
    }
```

View.cs

```
public void ShowImage(string path)
        {
            if (path != null)
            {
                Bitmap bitmap = new Bitmap(path);
                this.imageResolutionStatusLabel.Text = String.Format("{0}x{1}",
bitmap.Width, bitmap.Height);

                Size clientSize = this.mainPictureBox.ClientSize;

                this.indexStatusLabel.Text =
_presenter.GetCurrentImagePositionInCollection();
                this.scaleStatusLabel.Text = String.Format("{0:0.00} %", 100.0 *
clientSize.Width / bitmap.Width * clientSize.Height / bitmap.Height);
                this.imageTimestampStatusLabel.Text =
File.GetCreationTime(path).ToString();

                this.mainPictureBox.Image = bitmap;
            }

            else
            {
                MessageBox.Show("No images were loaded!", "Invalid operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

Listă de activități

Brînză Paul-Cristian:

- SRS
- Implementare modul DLL
- Șablon proiectare
- Antente în fișiere sursă

Covaș Andrei-Iulius:

- Help
- Respectarea standardelor
- Interfață grafică
- Documentație

Dobreanu Mircea-Constantin:

- Testare
- UML

- Tratarea situațiilor excepționale
- Respectarea standardului de scriere a codului
- Comentarii