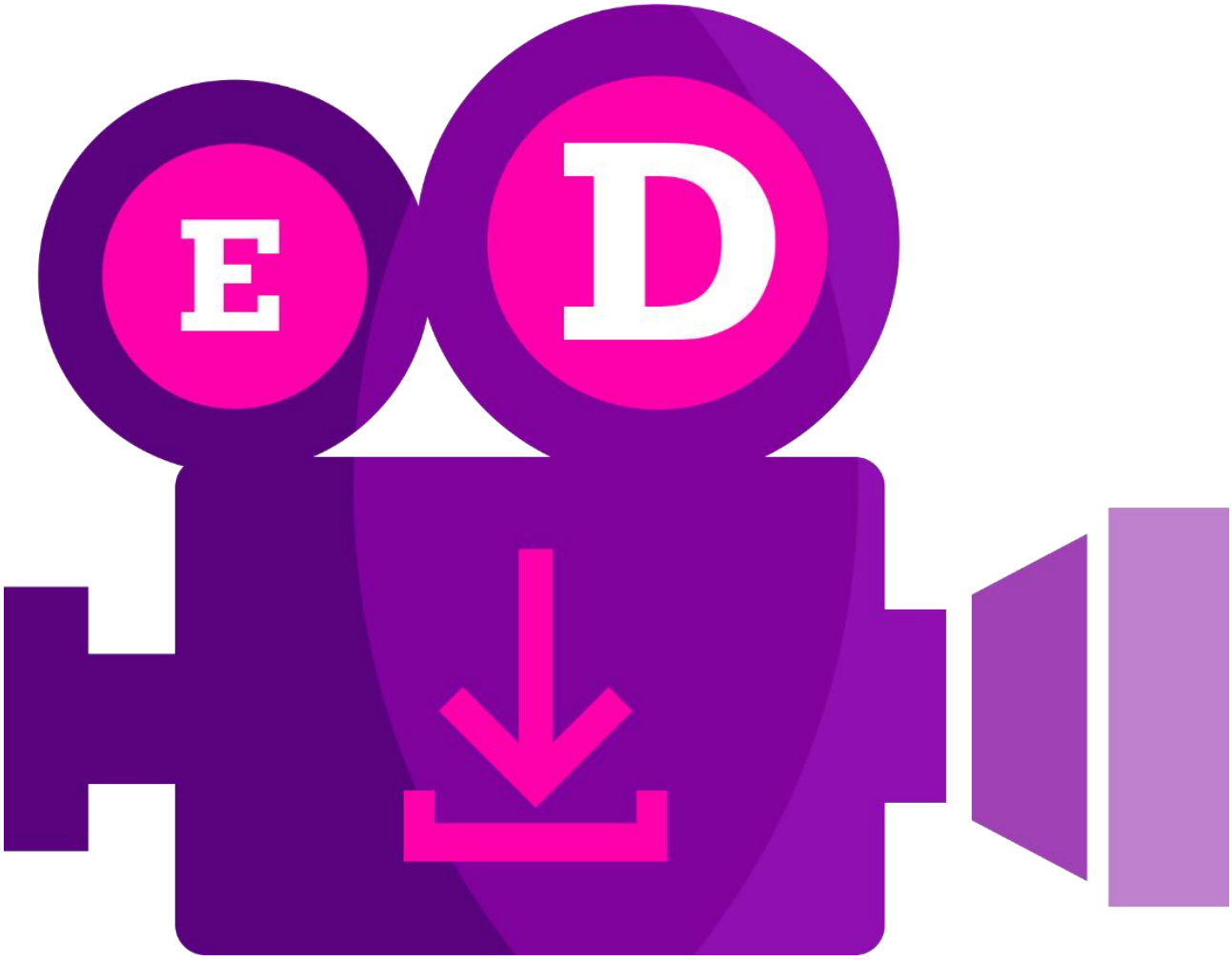
Università degli Studi di Napoli Parthenope

Dipartimento di Scienze e Tecnologie

Corso di Programmazione 3

# *ED Films*




Emanuele D’Ambrosio MAT - 0124002587

Domenico Zeno MAT - 0124002479

# Obiettivo

L'obbiettivo dell'applicazione sviluppata è permettere la visione di film gratuiti attraverso un'interfaccia di facile interpretazione. L'amministratore potrà inserire, modificare o eliminare il film e le informazioni ad esso correlato.

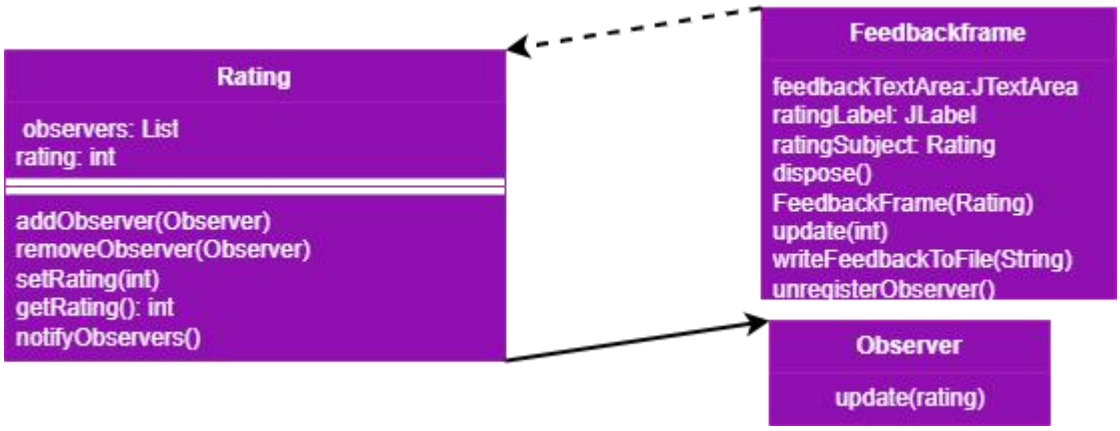
# Specifiche

Per la creazione di questa applicazione abbiamo utilizzato il linguaggio Java applicando diversi design pattern:

- Observer pattern
- Strategy pattern
- MVC
- Builder pattern

# Pattern

Observer:




**1. Observer**

L'interfaccia Observer definisce il metodo update che verrà chiamato dal Subject per notificare i cambiamenti agli osservatori:

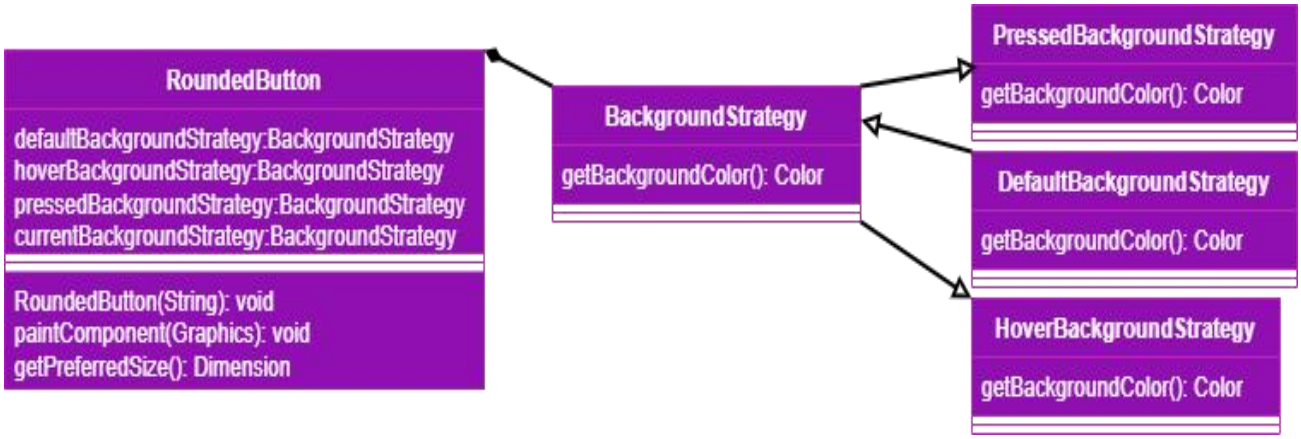
**2. Observable**

La classe Rating rappresenta il Subject. Mantiene una lista di osservatori e notifica loro quando il rating cambia:

### 3. ConcreteObserver

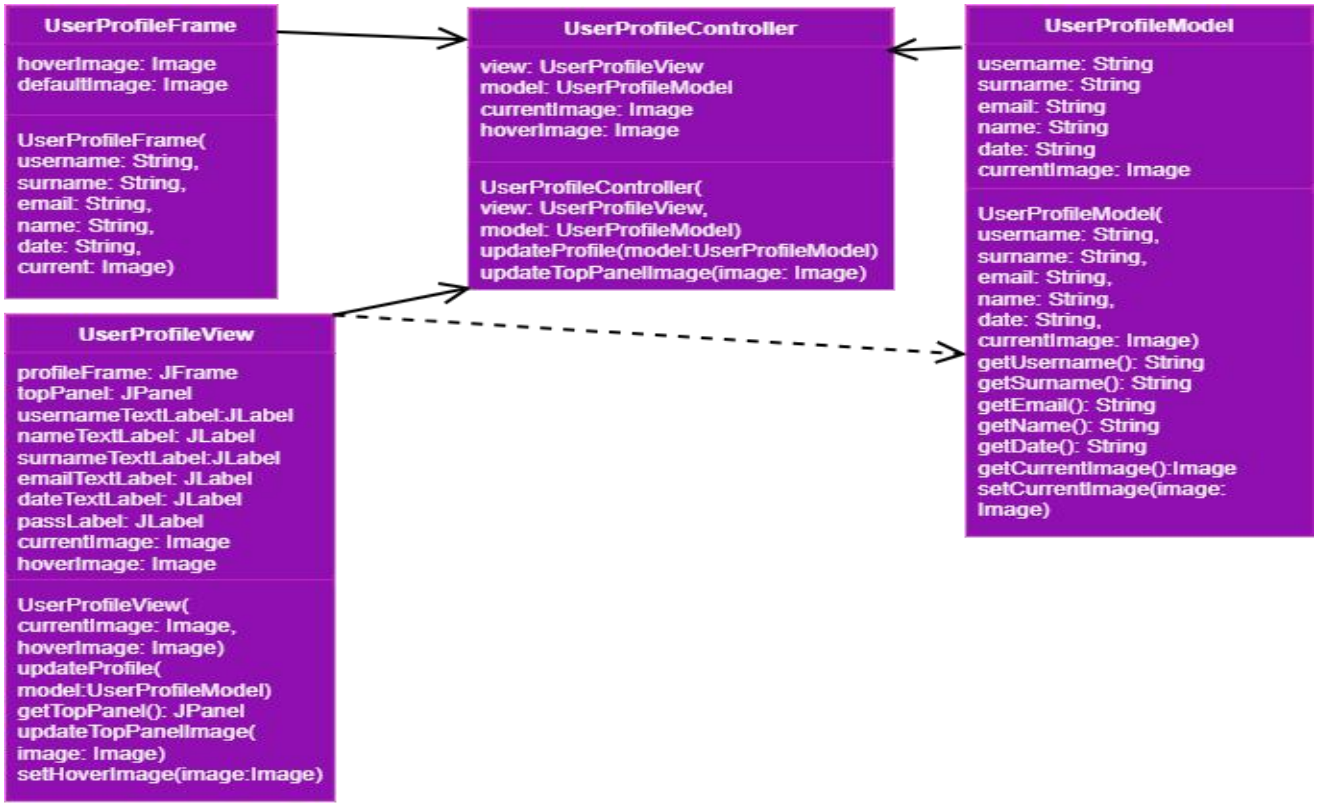
La classe FeedbackFrame rappresenta il ConcreteObserver. Implementa l'interfaccia Observer e aggiorna la UI quando riceve notifiche di cambiamento dal Subject

Strategy pattern:



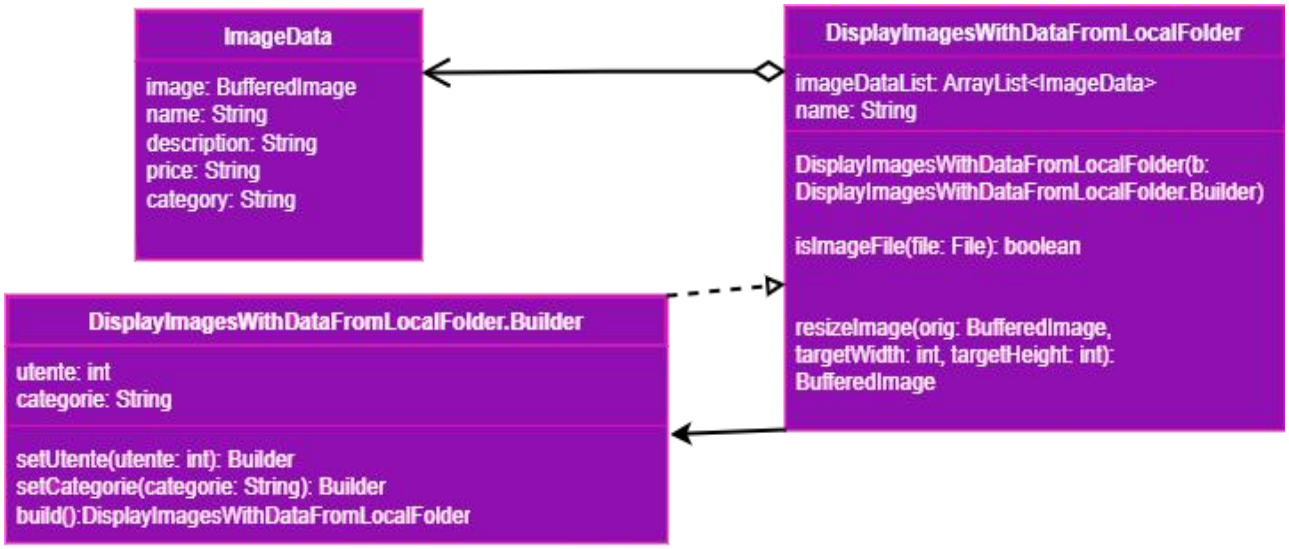

- **Interfaccia Strategy (**BackgroundStrategy**)**: Definisce un contratto comune (getBackgroundColor()) che tutte le strategie di sfondo devono implementare.
- **Implementazioni delle Strategie: (**DefaultBackgroundStrategy**,** HoverBackgroundStrategy**,** PressedBackgroundStrategy**)**: Forniscono implementazioni specifiche per ottenere il colore di sfondo in diversi stati del pulsante (default, hover, cliccato).
- **Classe Context (**RoundedButton**)**: Questa classe (RoundedButton) utilizza un'istanza di BackgroundStrategy (currentBackgroundStrategy) per determinare dinamicamente quale strategia di sfondo applicare in base agli eventi del mouse (mouseEntered, mouseExited, mousePressed, mouseReleased).
- **Utilizzo delle Strategie**: Ogni volta che un evento del mouse viene rilevato, il colore di sfondo del pulsante (currentBackgroundStrategy) viene impostato sulla strategia appropriata e successivamente viene chiamato repaint() per aggiornare l'aspetto del pulsante.

MVC(Model-View-Controller):



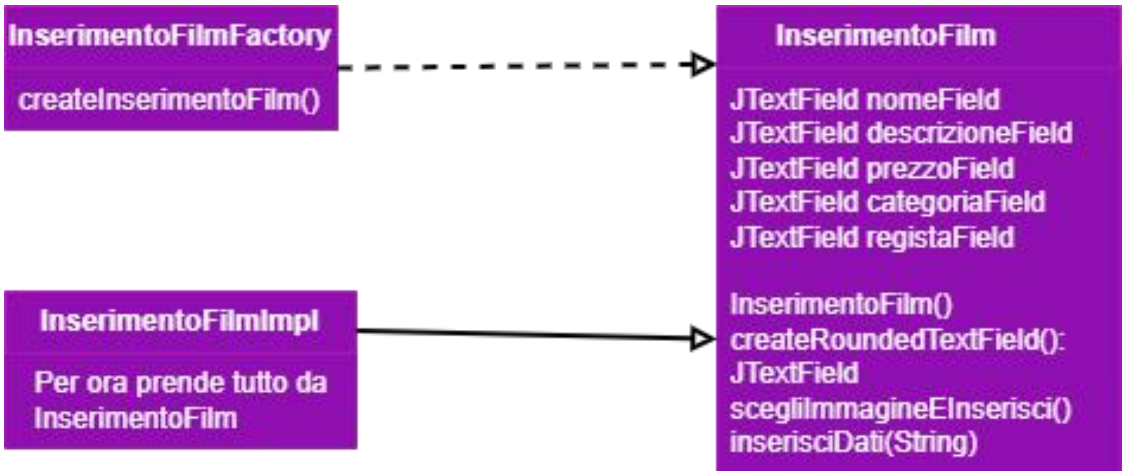

- **Model** :La classe UserProfileModel è il modello dei dati dell'utente. Contiene gli attributi username, surname, email, name, date e currentImage, e fornisce metodi per ottenere e impostare questi valori. Questo rappresenta lo stato dell'applicazione e la logica di business.
- **View** :La classe UserProfileView gestisce l'interfaccia utente. Contiene il frame JFrame, i vari pannelli e le etichette per visualizzare i dati dell'utente. Inoltre, fornisce metodi per aggiornare la vista quando i dati nel modello cambiano.
- **Controller** :La classe UserProfileController gestisce l'interazione tra il modello e la vista. Si occupa di aggiornare il modello quando l'utente interagisce con la vista (ad esempio, selezionando una nuova immagine), e di aggiornare la vista quando il modello cambia.
- **Inizializzazione**:La classe UserProfileFrame configura l'applicazione creando istanze del modello, della vista e del controller, e collegandoli insieme.

Builder pattern:




- **Classe Builder**: E' una classe interna statica Builder con metodi che impostano variabili specifiche (come setUtente e setCategorie), e un metodo build che costruisce l'oggetto principale DisplayImagesWithDataFromLocalFolder.
- **Metodo** build: Il metodo build della classe Builder crea una nuova istanza di DisplayImagesWithDataFromLocalFolder, passando this (l'istanza del builder) al costruttore privato della classe principale.
- **Costruttore privato**: La classe principale DisplayImagesWithDataFromLocalFolder ha un costruttore privato che accetta un oggetto Builder come parametro, utilizzando i dati forniti dal builder per inizializzare i propri campi.

Questi sono i vari diagrammi UML di alcuni dei pattern utilizzati, un altro pattern che si potrebbe implementare è il factory methods :



Il codice attuale non sfrutta appieno i vantaggi del Factory Method Pattern, dato che la classe InserimentoFilmImpl non aggiunge alcuna funzionalità specifica rispetto a InserimentoFilm. Tuttavia, il design attuale è preparato per un utilizzo futuro nel quale potremmo inserire film in modi differenti, creando nuove implementazioni di InserimentoFilm.

## Funzionalità di base

Le funzionalità base che la nostra applicazione mette a disposizione sono:

- Registrazione
- Login
- Visione dei film
- Feedback della pagina
- Ricerca per generi
- Ricerca per nome

## Registrazione

La registrazione è un'operazione fondamentale per il funzionamento dell'applicazione. Solo gli utenti registrati potranno visionare i film.

```
registerButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Recupera i dati dai campi di input
        String name = nameField.getText();
        String Username = usernameField.getText();
        String surname = surnameField.getText();
        String email = emailField.getText();
        String password = new String(passwordField.getPassword());
        String dateOfBirth = dobField.getText();

        // Chiama il metodo per salvare i dati
        saveUserData(name, surname, email, password,Username,dateOfBirth);

        // Invia l'email in modo asincrono per confermare via email
        Thread emailThread = new Thread(() -> {
            EmailSender emailSender = new EmailSender();
            emailSender.sendEmail(email);
        });
        emailThread.start();
        // Crea un'istanza di SitoFrame e passa i dati necessari
           sitoFrame = new SitoFrame(name, surname, email, Username, dateOfBirth,
currentImage, faviconIcon,null);
```

```
        // Crea un'istanza di UserProfileFrame e passa i dati necessari
          UserProfileFrame userProfileFrame = new UserProfileFrame(name, surname,
email, Username, dateOfBirth, currentImage);
        // Chiudi la finestra di registrazione
        registrationFrame.dispose();
    }
});
```

Questa parte di codice si occupa di prendere i dati dell'utente inseriti in input e con saveUserData, salvarli nel file utenti.txt. In seguito, verrà inviata un'e-mail di conferma di registrazione e successivamente l'utente verrà portato al sitoFrame.Verrà anche aperto lo UserprofileFrame per permettere all'utente di modificare la sua immagine profilo e per ultimo chiude il frame di registrazione.

## Login

Il login è l'operazione che si occupa di confrontare i dati dell'utente con quelli presenti in utenti.txt e in seguito aprire SitoFrame passando i dati immessi dall'utente.

```
private void login() {
    // prendo i campi username e password
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());
    //controllo che l'username e la password siano corretti
    if (authenticate(username, password)) {
        dispose(); // Chiudi la finestra di login
    } else {
        JOptionPane.showMessageDialog(null, "Accesso fallito. Riprova.");
    }
}
```

## Visione dei film

La visione dei film avviene cliccando sul pulsante riproduci nella classe SitoFrame che richiama DisplayImage.

```
button.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseEntered(MouseEvent e) {
                ImageIcon hoverIcon = new ImageIcon("7B.png");
                 Image hoverImage = hoverIcon.getImage().getScaledInstance(120, 120,
Image.SCALE_SMOOTH);
                button.setIcon(new ImageIcon(hoverImage));
            }
```

```
            @Override
            public void mousePressed(MouseEvent e) {
                File videoFile = new File("uploads/" + imageData.name+".mp4");
                // Crea un oggetto File con il nome del video

                SwingUtilities.invokeLater(() -> {
                // Ottieni l'istanza di default del toolkit
                Toolkit toolkit = Toolkit.getDefaultToolkit();
                // Ottieni le dimensioni dello schermo
                Dimension screenSize = toolkit.getScreenSize();
                int screenWidth = (int) screenSize.getWidth();
                // Converti double in int
                int screenHeight = (int) screenSize.getHeight();
                // Converti double in int
                //prendiamo il nome dell'immagine che sarà uguale al nome del video
                JFrame videoFrame = new JFrame(imageData.name);
                //diamo delle impostazioni di base alla dimensione del video
                videoFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                videoFrame.setSize(screenWidth, screenHeight-35);

                VideoPlayer videoPlayer = new VideoPlayer(videoFile);
                videoFrame.add(videoPlayer);
                videoFrame.setVisible(true);

            });
            }
            @Override
            public void mouseExited(MouseEvent e) {
                ImageIcon hoverIcon = new ImageIcon("7A.png");
                 Image hoverImage = hoverIcon.getImage().getScaledInstance(120, 120,
Image.SCALE_SMOOTH);
                button.setIcon(new ImageIcon(hoverImage));
            }
        });
```

## Feedback della pagina

Il feedback ha due componenti la prima è in alto ed è una valutazione con dei pulsanti a stella , mentre in basso c'è uno spazio riempibile con del testo che l'utente può inviare e verrà salvato nel file feedback.txt

```
        // area di testo per il feedback
        feedbackTextArea = new JTextArea(10, 40);
        JScrollPane scrollPane = new JScrollPane(feedbackTextArea);

        // Creazione delle stelle
        JPanel ratingPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
```

```
        for (int i = 1; i <= 5; i++) {
            JButton starButton = new JButton("\u2665"); // Simbolo stella
            starButton.setFont(new Font("Arial", Font.PLAIN, 24));
            int finalI = i; // Variabile finale per il listener
            starButton.addActionListener(e -> setRating(finalI));
            ratingPanel.add(starButton);
        }
        ratingLabel = new JLabel("Valutazione: " + userRating + " stelle");
        //pulsante per inviare il feedback
        JButton sendButton = new JButton("Invia Feedback");

        sendButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String feedback = feedbackTextArea.getText();
                if (!feedback.isEmpty()) {
                    writeFeedbackToFile(feedback);
                          JOptionPane.showMessageDialog(null, "Feedback inviato con
successo!");
                    feedbackTextArea.setText("");
                    dispose();
                } else {
                        JOptionPane.showMessageDialog(null, "Inserisci del feedback
prima di inviare.");
                }
            }
        });
```

## Ricerca per generi

La ricerca per generi permette tramite checkbox di vedere solo i film di un determinato genere.

```
public void mousePressed(MouseEvent e) {
            // Crea un nuovo frame per le categorie
            JFrame categorieFrame = new JFrame("Categorie");
            categorieFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            categorieFrame.setSize(400, 300);//dimensione del frame

            // Creazione dei JCheckBox per i tipi di video
            JCheckBox avventuraCheckBox = new JCheckBox("Avventura");
            JCheckBox storicoCheckBox = new JCheckBox("Storico");
            JCheckBox animazioneCheckBox = new JCheckBox("Animazione");
            JCheckBox animeCheckBox = new JCheckBox("Anime");
            JCheckBox azioneCheckBox = new JCheckBox("Azione");
            JCheckBox commediaCheckBox = new JCheckBox("Commedia");
            JCheckBox pauraCheckBox = new JCheckBox("Paura");
```

```
            JCheckBox thrillerCheckBox = new JCheckBox("Thriller");
            JCheckBox fantascienzaCheckBox = new JCheckBox("Fantascienza");

            Container contentPane = categorieFrame.getContentPane();
            contentPane.setLayout(new GridLayout(0, 1));
            // Layout a colonna per i checkbox
            contentPane.add(avventuraCheckBox);
            contentPane.add(storicoCheckBox);
            contentPane.add(animazioneCheckBox);
            contentPane.add(animeCheckBox);
            contentPane.add(azioneCheckBox);
            contentPane.add(commediaCheckBox);
            contentPane.add(pauraCheckBox);
            contentPane.add(thrillerCheckBox);
            contentPane.add(fantascienzaCheckBox);
            JButton confermaButton = new JButton("Conferma");
            confermaButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    // Raccolta delle selezioni delle checkbox
                    StringBuilder selezioni = new StringBuilder();
                                                if  (avventuraCheckBox.isSelected())
selezioni.append("Avventura:");
                    if (storicoCheckBox.isSelected()) selezioni.append("Storico:");
                                                if  (animazioneCheckBox.isSelected())
selezioni.append("Animazione:");
                    if (animeCheckBox.isSelected()) selezioni.append("Anime:");
                    if (azioneCheckBox.isSelected()) selezioni.append("Azione:");
                                                if  (commediaCheckBox.isSelected())
selezioni.append("Commedia:");
                    if (pauraCheckBox.isSelected()) selezioni.append("Paura:");
                                                if  (thrillerCheckBox.isSelected())
selezioni.append("Thriller:");
                                            if  (fantascienzaCheckBox.isSelected())
selezioni.append("Fantascienza:");

                    // Rimuove gli ultimi due punti se presente
                    if (selezioni.length() > 0) {
                        selezioni.deleteCharAt(selezioni.length() - 1);
                    }
                    // Controlla se sono state selezionate categorie
                    if (selezioni.length() == 0) {
                        // Nessuna categoria selezionata, gestisci questo caso qui
                        sitoFrame.dispose();
                          SitoFrame newSitoFrame = new SitoFrame(username, surname,
email, name, date, image, faviconIcon, null);
                        newSitoFrame.setVisible(true);
                    } else {
                        // Altrimenti, chiamiamo SitoFrame2 e passiamo le selezioni
                        sitoFrame.dispose();
```

```
                        SitoFrame newSitoFrame = new SitoFrame(username, surname,
email, name, date, image, faviconIcon, selezioni.toString());
                        newSitoFrame.setVisible(true);
                    }
                    // Chiudi il frame delle categorie
                    categorieFrame.dispose();
                }
            });

      // Aggiunta del pulsante "Conferma" al contentPane del frame delle categorie
            contentPane.add(confermaButton);
            categorieFrame.setVisible(true);
        }
```

Ogni volta che viene premuto il pulsante di conferma viene ricaricato SitoFrame ma passa la stringa della selezione in modo da non caricare film di altre categorie.

## Possibili miglioramenti:

Questo è solo uno scheletro di come potrebbe diventare questa applicazione in quanto vi possono essere fatti vari miglioramenti e aggiunte come l'aggiunta di una comunicazione tra i vari utenti e una media di valutazione per film in modo da creare una community , oppure aggiungere anche una cronologia dei film visti e in base alle preferenze mostrare i film delle stesse categorie.