Politecnico di Torino

DIPARTIMENTO DI AUTOMATICA E INFORMATICA

Laurea Magistrale in Computer Engineering

Progetto 2.1: Backup di emergenza

Candidati:

Giacomo Ponzuoli

Riccardo Marco Miracapillo

Vito Silvetri

Anno Accademico
2023/2024

Indice

Capitolo 1

Introduzione

1.1 Funzionalità e Obiettivi dell'Applicazione

Questo documento presenta un'applicazione software sviluppata in Rust, progettata per facilitare il processo di backup dei dati su dischi esterni, come ad esempio chiavette USB. L'applicativo consente agli utenti di effettuare backup in modo intuitivo attraverso comandi basati sull'interfaccia grafica, permettendo l'interazione tramite mouse.

Una volta attivato il comando di backup, tramite il riconoscimento di una specifica forma selezionata nell'apposita GUI, è richiesta una conferma da parte dell'utente. Tale conferma avviene mediante il tracciamento di una seconda forma, anch'essa selezionata nella medesima GUI. La sorgente del backup può essere configurata in fase di definizione dell'applicativo, o anche successivamente, rendendo il processo flessibile e personalizzabile in base alle esigenze dell'utente.

L'applicazione è progettata per essere attiva in background durante il funzionamento del PC, garantendo un basso consumo di CPU. A tal fine, è previsto un monitoraggio del consumo di CPU ogni due minuti, con registrazione delle informazioni in un file di log. Al termine di ogni operazione di backup, l'applicativo genera un ulteriore file all'interno del path di destinazione, che specifica la dimensione complessiva dei file salvati e il tempo di CPU impiegato per completare l'operazione.

Con queste caratteristiche, l'applicativo mira a fornire un'esperienza utente semplice ed efficiente, soddisfacendo le necessità di backup in modo pratico e accessibile.

1.2 Rust: Una Scelta Strategica per lo Sviluppo dell'Applicativo

Il presente applicativo, progettato per consentire la gestione e l'esecuzione di backup in modo efficiente e sicuro, è stato realizzato utilizzando il linguaggio di programmazione Rust. La scelta di Rust non è casuale: questo linguaggio rappresenta una soluzione moderna e affidabile per lo sviluppo di applicazioni che richiedono un alto grado di sicurezza, prestazioni elevate ed una gestione ottimale della concorrenza. Attraverso le sue caratteristiche uniche, Rust garantisce la solidità e la robustezza necessarie per un sistema dedicato alla protezione e alla salvaguardia dei dati, aspetti fondamentali per un'applicazione di backup.

1.3 Funzionalità del Software

L'applicativo è stato sviluppato per offrire agli utenti tre funzionalità principali, concepite per garantire semplicità d'uso, efficienza e completezza nel processo di gestione dei backup. Queste funzionalità coprono l'intero ciclo di vita del software, dall'installazione all'esecuzione delle operazioni principali e al monitoraggio delle risorse, come descritto di seguito:

1. **Installazione e Disinstallazione**: Questa funzione consente di installare l'applicativo come servizio in background sulla macchina dell'utente. Per farlo, viene eseguito un file Bash che, in base al sistema operativo e all'architettura della macchina (ad esempio, nel caso di macOS), fornisce le opportune release dell'applicativo. Successivamente, l'installazione viene completata tramite l'esecuzione

di un apposito eseguibile. Durante il processo, viene resa disponibile una GUI di configurazione che permette agli utenti di personalizzare il comportamento dell'applicazione. Analogamente, è possibile disinstallare il software attraverso un comando dedicato.

2. **Backup**: Il processo di backup rappresenta il nucleo operativo dell'applicativo. Gli utenti possono avviare il backup dei dati selezionati tramite il riconoscimento di forme tracciate con il mouse sullo schermo. La prima forma attiva il processo, mentre la seconda serve come conferma, garantendo così un doppio livello di controllo.

3. **Logging**: Questa funzionalità permette di monitorare l'utilizzo della CPU da parte dell'applicativo, registrando periodicamente i dati raccolti in un file di log memorizzato sul Desktop. Tale meccanismo fornisce agli utenti una chiara visione del consumo di risorse e contribuisce a garantire trasparenza ed efficienza operativa.

Queste tre funzionalità lavorano sinergicamente per offrire una soluzione di backup affidabile, flessibile e orientata all'utente.

Capitolo 2

Installazione (e Disinstallazione)

La portabilità rappresenta un elemento centrale nella progettazione dell'applicativo, poiché garantisce la possibilità di eseguirlo su diverse piattaforme e sistemi operativi senza richiedere modifiche al codice sorgente. Tale caratteristica è fondamentale per assicurare una diffusione capillare e un accesso agevole da parte di una vasta platea di utenti.

Per agevolare questo processo, l'applicativo include una directory denominata `release`, all'interno della quale sono disponibili gli eseguibili specifici per ciascun sistema operativo supportato. Per avviare la fase di installazione, l'utente dovrà eseguire il comando appropriato da terminale, posizionandosi nella directory principale dell'applicativo e digitando l'eseguibile corrispondente al sistema operativo in uso sulla propria macchina.

2.1 Windows

- **Installazione**: per avviare il processo di installazione, è necessario eseguire il seguente comando nel terminale:

```
release\windows\Group-35.exe
```

In caso di difficoltà o esigenze specifiche, è possibile eseguire singolarmente i vari eseguibili relativi ai processi dell'applicativo utilizzando i seguenti comandi:

- `release/windows/gui.exe`: consente di avviare l'interfaccia grafica (GUI) di configurazione, permettendo all'utente di modificare il percorso sorgente/destinazione o le forme utilizzate per il riconoscimento.
- `release/windows/main.exe`: permette di eseguire il software per il backup manualmente, qualora l'utente preferisca non utilizzare la modalità in background.
- `release/windows/Group-35.exe`: consente di effettuare nuovamente l'installazione dell'applicativo, ripristinando lo stato iniziale del sistema.

- **Modifica**: Qualora fosse necessario apportare modifiche al codice, è consigliabile procedere prima con la disinstallazione dell'applicativo e successivamente eseguire lo script per rigenerare gli eseguibili relativi ai vari processi. Per eseguire tale operazione, utilizzare il seguente comando dalla linea di comando:

  ```
  windows_build_release.bat
  ```

  Nel caso in cui si riscontrino difficoltà durante l'esecuzione, è opportuno verificare che i permessi di esecuzione siano impostati correttamente. A tal fine, è possibile eseguire i seguenti comandi da terminale:

  ```
  chmod +x windows_build_release.bat
  windows_build_release.bat
  ```

- **Disinstallazione**: Per avviare il processo di disinstallazione dell'applicativo e terminare tutti i relativi processi, eseguire il seguente comando:

  ```
  release/windows/uninstall.exe
  ```

2.2 MacOS

- **Installazione**: per avviare il processo di installazione, è necessario eseguire il comando nel terminale in base all'architettura dell'elaboratore (Intel o ARM).
  In caso di **ARM**:

  ```
  release/macos-arm/Group-35
  ```

  In caso di **Intel**:

  ```
  release/macos-intel/Group-35
  ```

  In caso di difficoltà o esigenze specifiche, è possibile eseguire singolarmente i vari eseguibili relativi ai processi dell'applicativo utilizzando i seguenti comandi (sempre in base al tipo di architettura):

  - `release/macos-arm/gui` o `release/macos-intel/gui`: consente di avviare l'interfaccia grafica (GUI) di configurazione, permettendo all'utente di modificare il percorso sorgente/destinazione o le forme utilizzate per il riconoscimento.
  - `release/macos-arm/main` o `release/macos-intel/main`: permette di eseguire il software per il backup manualmente, qualora l'utente preferisca non utilizzare la modalità in background.
  - lanciare `release/macos-arm/Group-35` o `release/macos-intel/Group-35`: consente di effettuare nuovamente l'installazione dell'applicativo, ripristinando lo stato iniziale del sistema.

- **Modifica**: Qualora fosse necessario apportare modifiche al codice, è consigliabile procedere prima con la disinstallazione dell'applicativo e successivamente eseguire

lo script per rigenerare gli eseguibili relativi ai vari processi. Per eseguire tale operazione, utilizzare il seguente comando dalla linea di comando:

```
macos_build_release.sh
```

Nel caso in cui si riscontrino difficoltà durante l'esecuzione, è opportuno verificare che i permessi di esecuzione siano impostati correttamente. A tal fine, è possibile eseguire i seguenti comandi da terminale:

```
chmod +x macos_build_release.sh
macos_build_release.sh
```

- **Disinstallazione**: Per avviare il processo di disinstallazione dell'applicativo e terminare tutti i relativi processi, eseguire il comando nel terminale in base all'architettura dell'elaboratore (Intel o ARM).
  In caso di **ARM**:

  ```
  release/macos-arm/uninstall.exe
  ```

  In caso di **Intel**:

  ```
  release/macos-intel/uninstall.exe
  ```

2.2.1 Prerequisiti

Il comando `osascript` dovrebbe essere preinstallato nel sistema operativo. Qualora non fosse operativo, si consiglia di verificare la sua presenza eseguendo il seguente comando nel terminale:

```
which osascript
```

Se `osascript` è correttamente installato, il terminale restituirà il percorso del file eseguibile. In caso contrario, non verrà fornito alcun risultato.

2.3 Linux

- **Installazione**: per avviare il processo di installazione, è necessario eseguire il seguente comando nel terminale:

  ```
  release/linux/Group-35
  ```

  In caso di difficoltà o esigenze specifiche, è possibile eseguire singolarmente i vari eseguibili relativi ai processi dell'applicativo utilizzando i seguenti comandi:

  - `release/linux/gui`: consente di avviare l'interfaccia grafica (GUI) di configurazione, permettendo all'utente di modificare il percorso sorgente/destinazione o le forme utilizzate per il riconoscimento.
  - `release/linux/main`: permette di eseguire il software per il backup manualmente, qualora l'utente preferisca non utilizzare la modalità in background.
  - `release/linux/Group-35`: consente di effettuare nuovamente l'installazione dell'applicativo, ripristinando lo stato iniziale del sistema.

- **Modifica**: Qualora fosse necessario apportare modifiche al codice, è consigliabile procedere prima con la disinstallazione dell'applicativo e successivamente eseguire lo script per rigenerare gli eseguibili relativi ai vari processi. Per eseguire tale operazione, utilizzare il seguente comando dalla linea di comando:

  ```
  linux_build_release.sh
  ```

  Nel caso in cui si riscontrino difficoltà durante l'esecuzione, è opportuno verificare che i permessi di esecuzione siano impostati correttamente. A tal fine, è possibile eseguire i seguenti comandi da terminale:

  ```
  chmod +x linux_build_release.sh
  linux_build_release.sh
  ```

- **Disinstallazione**: Per avviare il processo di disinstallazione dell'applicativo e terminare tutti i relativi processi, eseguire il seguente comando:

  ```
  release/linux/uninstall.exe
  ```

2.3.1 Prerequisiti

Prima di utilizzare l'applicazione su sistemi Linux, è necessario assicurarsi che siano installate le librerie richieste. Se non sono già presenti nel sistema, eseguire i seguenti comandi:

```
# Aggiorna l'elenco dei pacchetti
sudo apt update

# Installa le librerie necessarie
```

```
sudo apt install glib-2.0
sudo apt install libgtk2.0-dev
sudo apt install libgtk-3-dev
sudo apt install libx11-dev
sudo apt install librust-also-sys-dev
```

2.4 Struttura del codice di installazione

Il processo di installazione dell'applicativo è gestito attraverso uno script che rileva il sistema operativo dell'utente e configura l'applicazione di conseguenza. La struttura del codice è suddivisa in vari passaggi che permettono l'esecuzione dei processi necessari per l'installazione, la configurazione e l'autostart dell'applicativo. Di seguito vengono descritti i principali componenti ed operazioni eseguite dal codice:

- **Rilevamento del sistema operativo:** Il codice inizia rilevando il sistema operativo corrente tramite la variabile `env::consts::OS`, che restituisce una stringa identificativa del sistema in uso. Questo permette al codice di adattarsi alle specifiche esigenze dei vari sistemi operativi supportati, come Windows, macOS e Linux.

- **Determinazione del percorso di installazione:** Successivamente, viene determinato il percorso della cartella `Desktop` dell'utente, utilizzando la funzione `dirs::desktop_dir()`. Il percorso completo della cartella di installazione dell'applicativo viene quindi costruito concatenando la directory `Group-35/release` alla cartella Desktop dell'utente.

- **Gestione dell'autostart:** Una parte fondamentale del processo di installazione riguarda la configurazione dell'applicazione per l'avvio automatico al boot del sistema. A seconda del sistema operativo, vengono configurati diversi meccanismi di autostart:

- **Windows:** Se il sistema operativo è Windows, viene aggiunta l'estensione `.exe` al percorso dell'eseguibile e l'applicazione viene configurata per l'avvio automatico tramite l'uso del costruttore `AutoLaunchBuilder`. Questo strumento consente di abilitare l'avvio automatico senza l'intervento di agenti di sistema, come i Launch Agent di macOS.
- **macOS:** Per macOS, la configurazione dell'autostart avviene anch'essa tramite `AutoLaunchBuilder`, ma senza l'uso di Launch Agents. In questo caso, l'applicazione viene configurata per avviarsi automaticamente all'accensione del sistema tramite un meccanismo integrato nel sistema operativo.
- **Linux:** Per Linux, viene utilizzato un approccio simile, ma con una gestione diversa. In particolare, viene creata una cartella `autostart` all'interno della directory `/.config`, se non già esistente. Al suo interno viene creato un file `.desktop` che contiene le informazioni necessarie per avviare l'applicazione automaticamente al login dell'utente. Il contenuto del file `.desktop` include il nome dell'applicazione, il percorso dell'eseguibile e altre configurazioni necessarie per l'esecuzione automatica.

- **Verifica dell'esistenza di `output.csv`:** Prima di avviare la GUI, il codice verifica se un file chiamato `output.csv` esiste già nella directory di destinazione. Se il file è presente, la GUI non viene avviata, altrimenti viene lanciata la GUI per la configurazione iniziale dell'applicativo. La GUI viene eseguita tramite il comando associato al file `gui.exe` o `gui` a seconda del sistema operativo.

- **Esecuzione dell'applicazione di backup:** Una volta completata la fase di configurazione, il codice verifica se l'applicazione di backup (`main.exe` o `main`) è già in esecuzione. Se il processo è già attivo, viene visualizzato un messaggio di avviso; in caso contrario, viene avviata l'applicazione di backup in background. Per Windows, questo viene fatto controllando la lista dei processi attivi tramite il comando `tasklist`, mentre per altri sistemi operativi viene utilizzato il comando `pgrep`.

In sintesi, il codice di installazione gestisce in modo automatizzato il processo di configurazione e avvio dell'applicativo, garantendo la portabilità su diverse piattaforme e un'esperienza utente fluida attraverso l'auto-avvio e la gestione dei processi in background.

Capitolo 3

Backup

La fase di backup è il punto focale dell'applicazione. Una volta installata, l'applicazione rimane in ascolto di specifici movimenti del mouse che possono essere definiti in fase di configurazione. Una volta riconosciuti tali movimenti, vengono effettuati dei controlli preliminari per poi completare l'effettiva esecuzione del backup.

3.1 File di configurazione

Il file di configurazione nel progetto funge da elemento centrale per la gestione delle preferenze dell'utente. Questo file viene creato e aggiornato attraverso l'interazione con l'interfaccia grafica dell'applicazione. In questo caso, per la memorizzazione delle impostazioni relative al backup, viene utilizzato un file in formato CSV denominato `output.csv`.

Dati Memorizzati nel File di Configurazione

Il file include i seguenti dati:

- **Percorso della cartella sorgente**: Indica la directory da cui saranno prelevati i file da includere nel backup.
- **Drive di destinazione**: Specifica il percorso in cui verrà salvato il backup.

- **Preferenze sui tipi di file**: Include opzioni per filtrare i file da includere nel backup, come immagini, video, musica e documenti.
- **Segnali per avviare e confermare il backup**: Permette all'utente di scegliere i segnali visivi per l'avvio e la conferma del processo di backup.

Controlli per il Funzionamento del Programma

Per garantire un corretto funzionamento, sono state implementate le seguenti verifiche:

Esistenza del file `output.csv`:

- Durante l'intera esecuzione del programma, viene monitorata la presenza del file di configurazione.
- Qualora il file non fosse presente o fosse stato eliminato, l'applicazione provvederà automaticamente a generare un nuovo file `output.csv` con valori predefiniti.

Validazione durante il salvataggio:

- Prima di salvare le impostazioni, il sistema verifica la completezza delle informazioni fornite dall'utente.
- Sono inoltre effettuati controlli sull'esistenza dei percorsi specificati per la cartella sorgente e la destinazione del backup. In caso di errori, l'utente riceverà notifiche specifiche per correggere eventuali anomalie.

Grazie a queste misure, si contribuisce a rendere il sistema più robusto nel fronteggiare eventuali anomalie, garantendo un'elevata affidabilità dell'applicazione.

3.2 User Experience

Nel contesto di questo progetto, è stato utilizzato il framework `Iced` per la realizzazione di un'interfaccia grafica pensata per facilitare l'esperienza utente.

Iced è un framework per Rust che consente di creare applicazioni con interfacce utente reattive, performanti e scalabili. Con una sintassi semplice e un modello di programmazione dichiarativo, Iced permette lo sviluppo di GUI multipiattaforma, che possono essere eseguite su sistemi operativi come Windows, macOS e Linux. Il framework si basa sul modello di programmazione reattiva, dove l'interfaccia risponde automaticamente agli eventi generati dall'utente, come clic, input da tastiera o selezione di menù.

Menu configurazione

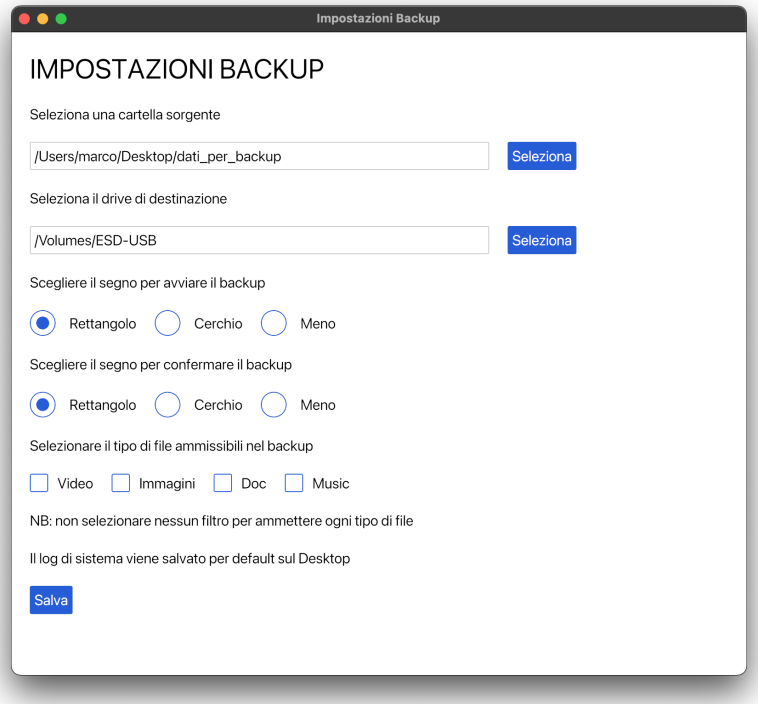

Figura 3.1: Interfaccia GUI

Per permettere all'utente di personalizzare i vari campi del file `output.csv` e scegliere i dati da includere all'interno del backup, è stata implementata un'interfaccia grafica semplice ed intuitiva, composta principalmente dai seguenti elementi:

- **Campi di input** per l'inserimento dei percorsi della cartella sorgente e della destinazione del backup.
- **Pulsanti** che consentono la selezione dei percorsi attraverso finestre di dialogo di sistema.
- **Checkbox** per la selezione dei tipi di file da includere nel backup.
- **Radio button** per la scelta dei segnali visivi (Rettangolo, Cerchio, Meno) utilizzati per avviare e confermare il backup.
- **Pulsante di salvataggio** che effettua la validazione dei dati inseriti e memorizza tutte le configurazioni.

In assenza di un file di configurazione esistente, l'applicazione avvia automaticamente il processo di configurazione con valori predefiniti, come il percorso del Desktop. In questa fase, l'interfaccia guida l'utente attraverso una serie di passaggi necessari per fornire le informazioni richieste, facilitando così la configurazione iniziale del programma.

L'interfaccia è progettata per fornire un feedback continuo all'utente. In caso di errori, come percorsi non validi o campi obbligatori non compilati, vengono visualizzati messaggi di dialogo che avvisano l'utente circa la natura dell'errore. Allo stesso modo, quando le impostazioni vengono salvate correttamente, vengono forniti messaggi di conferma che l'operazione è stata completata con successo.

Countdown

L'ultima interfaccia implementata all'interno del progetto, è la realizzazione di un timer countdown. Quest'ultima viene avviata automaticamente ogni qual volta l'utente traccia la prima figura sullo schermo, per avviare il processo di backup. Una volta avviato, l'utente avrà dieci secondi per confermare, con la seconda figura, l'esecuzione effettiva del backup.

Durante il countdown, l'utente è informato costantemente del tempo rimanente, grazie a una visualizzazione dinamica dell'orologio che viene aggiornata in tempo reale nella

finestra principale. Il messaggio visualizzato (che indica "Tempo rimasto: X secondi"), si aggiorna automaticamente ogni secondo. Se l'utente non effettua una scelta prima che il timer raggiunga lo zero, la finestra viene automaticamente chiusa (e vi è un segnale acustico di conferma), prevenendo l'interruzione o il blocco dell'applicazione.

Il timer grafico è utile anche in caso di attivazione involontaria del processo di backup, in questo modo l'utente ha una percezione visiva e può agire per non continuare con l'esecuzione.

3.3 Acquisizione stato del Mouse

Partendo dal presupposto che l'applicazione viene utilizzata in un contesto dove non è possibile utilizzare lo schermo, un modo per poter interagire con il programma ed innescare il processo di backup è attraverso le periferiche, quali mouse e tastiera. In particolar modo, si è scelto di utilizzare il mouse per poter impartire dei movimenti specifici. I movimenti scelti sono:

- Creazione di un rettangolo a tutto schermo: il rettangolo viene riconosciuto nel momento in cui il mouse mouse 'tocca' i quattro lati dello schermo.[1]
- Creazione di un cerchio: il programma salva tutti i movimenti del mouse, e controlla se entro una certa tolleranza i movimenti indicati formino un cerchio, calcolando prima un raggio medio e verificando che la distanza di tutti i punti da quel raggio sia uguale entro una determinata tolleranza.
- Creazione di una linea orizzontale: in maniera simile a quanto accade per il cerchio, il programma salva i movimenti del mouse e controlla che venga realizzata una linea orizzontale, anche qui entro una certa tolleranza.

Una volta avviato il programma, il flusso di esecuzione è il seguente:

[1] attraverso la classe Eventloop è possibile gestire diversi eventi di sistema. In questo caso permette di avere informazioni circa la risoluzione del monitor così da poterne individuar i bordi

1. Un thread secondario viene avviato per rimanere in ascolto e riconoscere precisi movimenti del mouse, sulla base della configurazione indicata.

2. Una volta riconosciuto il primo movimento, notificato tramite un feedback acustico, si ha a disposizione un limitato quantitativo di tempo (10 secondi) per effettuare un secondo movimento utilizzato per conferma.

3. Qualora anche la seconda fase dovesse andare a buon fine, la routine di backup viene chiamata e un feedback acustico acuto notificherà la corretta esecuzione del backup.

4. Qualora non si riuscisse ad effettuare in tempo il secondo riconoscimento, un feedback acustico più grave avvertirà dell'accaduto, riportando il flusso al thread principale.

3.4 Funzione di Backup

Una volta acquisiti i parametri attraverso il file di configurazione e aver correttamente riconosciuto i due movimenti indicati, la funzione:

```
   backup_execute(
&value.text_drive_destinazione,
&value.text_cartella_sorgente,
&vec!["all".to_string()],
).expect("errore nel backup");
```

si occupa di realizzare la copia dei dati all'interno del driver esterno indicato. La funzione acquisisce una serie di parametri tra cui la directory sorgente e destinazione del backup e una lista di eventuali filtri sulla tipologia di dati. Al fine di rendere il programma quanto più robusto possibile, sono stati introdotti una serie di controlli e operazioni aggiuntive, quali:

- Identificazione del corretto driver: si controlla se il drive inserito in fase di configurazione sia effettivamente ancora disponibile (ad esempio l'utente potrebbe aver

rimosso la chiavetta USB adibita al backup). Il backup può essere effettuato solo su un drive esterno, anche riconosciuto come 'Drive rimovibile' dalla maggior parte dei sistemi operativi.

- Filtraggio dei dati: l'utente ha possibilità di scegliere quali tipi di dati copiare
- Creazione di un report: alla fine del backup, nel drive destinazione viene anche creato un file di log contenente informazioni circa il tempo totale necessario al backup e la quantità di byte copiati.

In caso di errore o in caso di corretto completamento del backup il programma ritorna al processo principale per una nuova identificazione dei movimenti del mouse.

Capitolo 4

Logging

4.1 Logging

Uno dei requisiti dell'applicazione è la generazione di un log per il monitoraggio costante del consumo di CPU dovuto all'intero applicativo.

Oltre al thread principale che gestisce il backup, viene aperto un thread secondario il cui scopo è acquisire i dettagli del primo e salvarli in un file di log dedicato.

La soluzione implementata si basa sull'utilizzo dei canali della libreria MPSC[1].

Un 'ticker' viene creato per inviare un segnale ogni 120 secondi, e serve a controllare quando il thread deve scrivere un nuovo log nel file. Quando il segnale viene ricevuto:

1. Viene acquisito il PID del processo principale
2. Viene calcolato il relativo consumo di CPU
3. Il file di log viene aggiornato con il timestamp e il consumo calcolato

Esempio scrittura su file:

```
2024-09-23 14:49:52 - CPU Usage: 0.15%
2024-09-23 14:51:52 - CPU Usage: 0.17%
```

[1]Multi-producer, single-consumer FIFO queue communication primitives, gestisce una comunicaione message-based tra canali

Il thread è in un loop infinito che parte dal momento in cui il programma viene installato sul PC.

Capitolo 5

Conclusione

5.1 Architettura e Utilità del Programma

L'applicazione è stata progettata per garantire un'esperienza utente intuitiva ed efficiente, focalizzandosi sulla gestione automatizzata dei processi di backup. La possibilità di avviare il backup tramite il tracciamento di una forma sulla GUI rappresenta un approccio innovativo che semplifica l'interazione utente-software, migliorando la fruibilità e riducendo la necessità di interventi manuali. L'applicazione opera in background con un impatto minimo sulle risorse del sistema, un elemento cruciale per il suo utilizzo in contesti quotidiani e prolungati.

5.2 Risultati dei Test

I test effettuati hanno dimostrato la stabilità e la piena operatività dell'applicativo. I risultati relativi al consumo di risorse evidenziano un'ottimizzazione significativa: il consumo medio di CPU si attesta intorno allo 0% in condizioni di inattività (ovvero quando l'applicazione è in attesa del tracciamento di una forma per attivare il backup). Anche durante l'esecuzione di operazioni più complesse, come il riconoscimento delle

forme e l'avvio del backup, l'incremento del consumo è trascurabile, con un picco massimo registrato pari allo 0,15%, come illustrato dal seguente esempio di file di log:

```
2024-12-16 11:28:07 - CPU Usage: 0.00%
2024-12-16 11:30:07 - CPU Usage: 0.00%
2024-12-16 11:32:07 - CPU Usage: 0.00%
2024-12-16 11:34:07 - CPU Usage: 0.00%
2024-12-16 11:36:07 - CPU Usage: 0.02%
2024-12-16 11:38:07 - CPU Usage: 0.15%
2024-12-16 11:40:07 - CPU Usage: 0.00%
2024-12-16 11:42:07 - CPU Usage: 0.00%
2024-12-16 11:44:07 - CPU Usage: 0.00%
2024-12-16 11:46:07 - CPU Usage: 0.00%
```

Questi dati confermano la leggerezza dell'applicazione e la sua capacità di operare senza compromettere le prestazioni generali del sistema.

5.3 Accuratezza del Riconoscimento

Per quanto riguarda la precisione nel riconoscimento delle forme, si è scelto di adottare un approccio prudente per ridurre il rischio di attivazioni involontarie. L'accuratezza del riconoscimento dipende da diversi fattori, tra cui la risoluzione dello schermo, la velocità di movimento del mouse, la precisione dell'utente e il tipo di periferica di input (mouse o trackpad). Per garantire una maggiore affidabilità, è stata introdotta una soglia meno restrittiva, che minimizza la probabilità di falsi positivi, ossia l'attivazione del backup a causa di movimenti casuali del cursore. È stata inoltre offerta la possibilità di scegliere tra diverse forme, così da consentire la selezione di quella che presenta il minor numero di problematiche sulla base delle proprie abitudini di utilizzo.

5.4 Considerazioni Finali

In sintesi, l'applicazione dimostra di soddisfare pienamente gli obiettivi iniziali di progetto. L'architettura solida, la capacità di eseguire backup in modo intuitivo e l'elevata efficienza nell'utilizzo delle risorse rendono questo strumento ideale per l'utilizzo quotidiano. La stabilità operativa è garantita anche in condizioni di carico prolungato, mentre il sistema di riconoscimento basato su forme si distingue per accuratezza e affidabilità. L'applicazione può quindi essere considerata una soluzione robusta e performante per la gestione del backup su dischi esterni.