Convertitore Audio  
Matvej Rossi

# Indice

[1 Indice 2](#_Toc185427327)

[2 Introduzione 3](#_Toc185427328)

[2.1 Informazioni sul progetto 3](#_Toc185427329)

[2.2 Scopo 3](#_Toc185427330)

[3 Analisi 4](#_Toc185427331)

[3.1 Analisi del dominio 4](#_Toc185427332)

[3.2 Analisi e specifica dei requisiti 4](#_Toc185427333)

[3.3 Use case 8](#_Toc185427334)

[3.4 Pianificazione 9](#_Toc185427335)

[3.5 Analisi dei mezzi 10](#_Toc185427336)

[3.5.1 Software 10](#_Toc185427337)

[3.5.2 Hardware 10](#_Toc185427338)

[4 Progettazione 11](#_Toc185427339)

[4.1 Design dell’architettura del sistema 11](#_Toc185427340)

[4.2 Diagramma di flusso 12](#_Toc185427341)

[4.3 Design delle interfacce 13](#_Toc185427342)

[4.4 Design delle classi 13](#_Toc185427343)

[5 Implementazione 15](#_Toc185427344)

[5.1 Classe Converter 15](#_Toc185427345)

[5.2 Classe Validator 16](#_Toc185427346)

[5.3 Classe Main 17](#_Toc185427347)

[6 Test 18](#_Toc185427348)

[6.1 Protocollo di test 18](#_Toc185427349)

[6.2 Risultati test 22](#_Toc185427350)

[6.3 Mancanze/limitazioni conosciute 23](#_Toc185427351)

[7 Consuntivo 23](#_Toc185427352)

[8 Conclusioni 25](#_Toc185427353)

[8.1 Sviluppi futuri 25](#_Toc185427354)

[8.2 Considerazioni personali 25](#_Toc185427355)

[9 Bibliografia 26](#_Toc185427356)

[9.1 Sitografia 26](#_Toc185427357)

[10 Glossario 27](#_Toc185427358)

[11 Indice delle figure 27](#_Toc185427359)

[12 Allegati 27](#_Toc185427360)

# Introduzione

## Informazioni sul progetto

* Data di inizio progetto: 04.09.2024
* Allievo: Matvej Rossi
* Scuola: SAMT
* Sezione: Informatica
* Materia: Modulo 306
* Docente responsabile: Geo Petrini
* Data di consegna: 18.12.24

## Scopo

Principalmente il progetto ha come scopo di facilitare la conversione tra file audio diversi (MP3, FLAC, WAV, OGG, AAC, M4A, …) e così semplificare la ricerca all’utente del file audio del tipo corretto. Attraverso ad una interfaccia grafica di semplice utilizzo l’utente può facilmente convertire molteplici file audio nel tipo di file che preferisce. Il progetto è a scopo didattico. L’utilizzo di questo tipo di programmi è molto comune e utile, anche con una gamma di formati audio ridotta.

# Analisi

## Analisi del dominio

Il prodotto funziona su GUI / CMD e viene utilizzato nel bisogno di convertire un file audio nel tipo corretto. Esistono sul Web molteplici prodotti simili a questo, tutti creati per aiutare gli utenti nel sostituire il tempo di ricerca del file corretto nel tempo di aspettare di scaricare un semplice file del formato corretto, il mio prodotto serve ad aiutare l’utente localmente. L’utente ha il semplice bisogno di avere il file del formato corretto, possono essere dall’utente comune o un’utente con competenze informatiche avanzate. Per poter realizzare questo prodotto serve la conoscenza riguardo a come sono creati i vari formati di file, cosa contengono e la conoscenza del bitrate dei file audio.

## Analisi e specifica dei requisiti

L’applicazione base fatta su CLI dovrà avere principalmente i seguenti requisiti:

* Realizzazione dell’input per l’inserimento dei file e cartelle.
* Filtrazione dei files/cartelle sorgenti.
* Realizzazione dell’output.
* Scelta della qualità dell’audio.
* Scelta della cartella di destinazione.
* Controllare se il file di output è già esistente e far scegliere all’utente cosa fare in quel caso
* Controllo dell’avanzamento della conversione (gestire eventuali errori nel corso della conversione)
* Gestire eventuali errori.
* Gestione del file di configurazione.
* Controllo compatibilità file.
* Creazione della GUI

Il programma parte tramite CMD e utilizza la libreria FFMPEG che verrà scaricata dall’utente quando scaricherà il programma. Consente la conversione di file audio, funziona come un classico programma di conversione ma in questo caso il programma senza l’interfaccia grafica resta non molto “user friendly” ma molto diretto nel suo utilizzo, quindi consente la scelta della cartella di destinazione e consente di convertire non solo un singolo file ma una cartella intera. Il file di configurazione del programma è utile perché serve a memorizzare le scelte dell’utente.

|  |  |
| --- | --- |
| **Req-01** | |
| **Nome** | Realizzazione dell’input per i file e cartelle sorgenti. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’input può essere sia di un file singolo che una cartella. |

|  |  |
| --- | --- |
| **Req-02** | |
| **Nome** | Filtrare l’input (sia file singolo che cartella) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Senza il Req-01 non è possibile realizzarlo  I tipi di file da filtrare sono: MP3, FLAC, WAV, OGG, AAC, M4A  Altri tipi di file non sono accettati. |
| **Sotto requisiti** | |
| **Req-02\_1** | Filtrare l’input su file |
| **Req-02\_2** | Filtrare l’input su cartella/e |
| **Req-02\_3** | Consentire all’utente di scegliere la filtrazione del tipo di file. |

|  |  |
| --- | --- |
| **Req-03** | |
| **Nome** | Realizzare l’output con i file convertiti |
| **Priorità** | 2 |
| **Versione** | 2.0 |
| **Note** | Fa vedere a schermo e anche nella conversione il risultato della conversione, se essa ha funzionato o no. |
| **Sotto requisiti** | |
| **Req-03\_1** | Realizzare il salvataggio dei file convertiti localmente. |
| **Req-03\_2** | Consentire all’utente della scelta del tipo di file per l’output. |
| **Req-03\_3** | Controllo dei permessi di scrittura nella cartella di destinazione. |

|  |  |
| --- | --- |
| **Req-04** | |
| **Nome** | Consentire scelta del Bitrate all’utente. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Fa scegliere all’utente la qualità del file di output se ne ha la possibilità. |

|  |  |
| --- | --- |
| **Req-05** | |
| **Nome** | Consentire scelta della cartella di destinazione dei files. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | La possibilità di scegliere tramite CMD (poi GUI) la cartella di destinazione.  (Default: cartella sorgente) |

|  |  |
| --- | --- |
| **Req-06** | |
| **Nome** | Controllo se il file output è già esistente nella cartella di destinazione. |
| **Priorità** | 2 |
| **Versione** | 2.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **Req-04\_1** | Sovrascrivere il file con lo stesso nome se già ne esiste uno. |
| **Req-04\_2** | Sovrascrivere file / files in base alla scelta dell’utente. |

|  |  |
| --- | --- |
| **Req-07** | |
| **Nome** | Controllo dell’avanzamento della conversione. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | - |

|  |  |
| --- | --- |
| **Req-08** | |
| **Nome** | Gestione degli eventuali errori |
| **Priorità** | 2 |
| **Versione** | 2.0 |
| **Note** | Errori che possono essere generati tramite input, output o altro.  Testo dell’errore deve essere comprensibile all’utente. |

|  |  |
| --- | --- |
| **Req-09** | |
| **Nome** | Gestione e aggiunta del file di configurazione |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il file serve per contenere un “pattern” per il salvataggio del nome dei files, il formato di Output dei file, la qualità (bitrate) e la scelta di cosa fare in caso di errore nella conversione. |

|  |  |
| --- | --- |
| **Req-10** | |
| **Nome** | Controllo compatibilità dei file |
| **Priorità** | 1 |
| **Versione** | 2.0 |
| **Note** | I file che vengono presi dall’input devono essere controllati per gestire eventuali errori che possono succedere durante la conversione.  Il controllo di compatibilità viene fatto tramite la libreria o tramite il Magic Number. |

|  |  |
| --- | --- |
| **Req-11** | |
| **Nome** | Creazione dell’interfaccia grafica. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | - |

## Use case

L’attore principale che comunica con il sistema è l’utente finale che utilizza il programma.

Il sistema quindi prevede solamente l’utente come attore attivo che rispetta i requisiti.

Come allegato ho inserito il Visio che spiega tutti gli Use-Case.

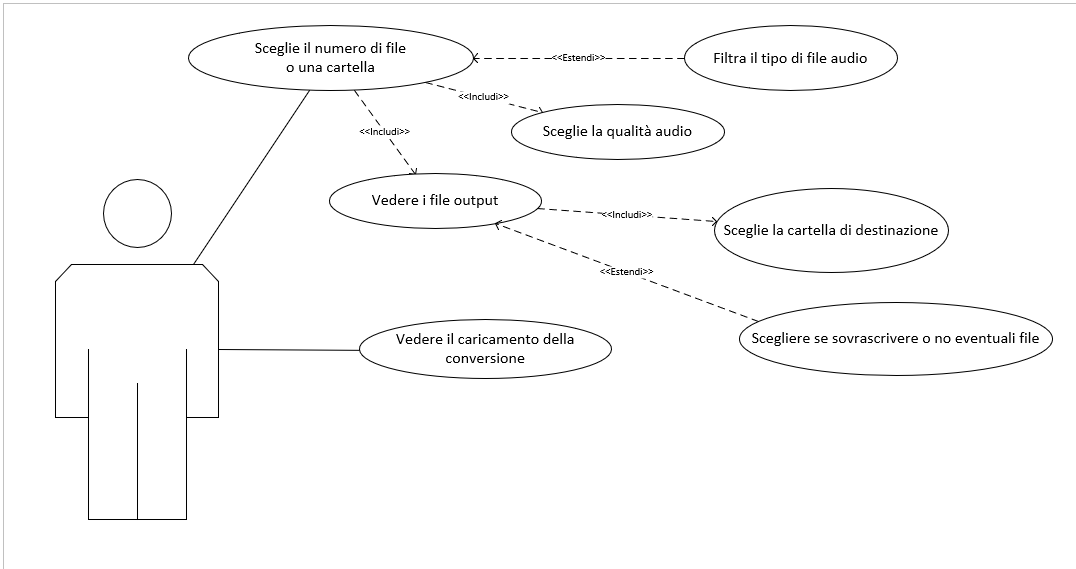


Figura 1: Use Case

## Pianificazione

La pianificazione del mio progetto è stata rappresentata mediante un diagramma di Gantt, dove vede come data di partenza del progetto il 04.09.24 e come fine il 18.12.2024. Immagine che contiene testo, schermata, numero, Parallelo

Descrizione generata automaticamente

Figura Gantt preventivo

## Analisi dei mezzi

Per la creazione del progetto mi è servito solamente un IDE e l’accesso a Internet.

### Software

Di seguito ci sono tutti i software e le relative versioni che mi sono servite per realizzare il progetto e la documentazione:

* Apache NetBeans IDE 20
* JDK: 19.0.2
* Word: 2019
* FFMPEG
* Visual Studio Code

### Hardware

Per eseguire il programma basta che su qualsiasi dispositivo, indipendentemente dal sistema operativo.

Il programma usa una particolare potenza quindi un qualsiasi PC odierno anche di bassa qualità può utilizzare il programma. Il programma può funzionare su qualsiasi sistema operativo e non presenta limitazioni.

Il PC Scolastico presentava queste specifiche di base:

* Nvidia T400 4GB
* Intel Core i7-13700 13th Gen
* Windows 10 22H2 x64
* 32 GB RAM

# Progettazione

## Design dell’architettura del sistema

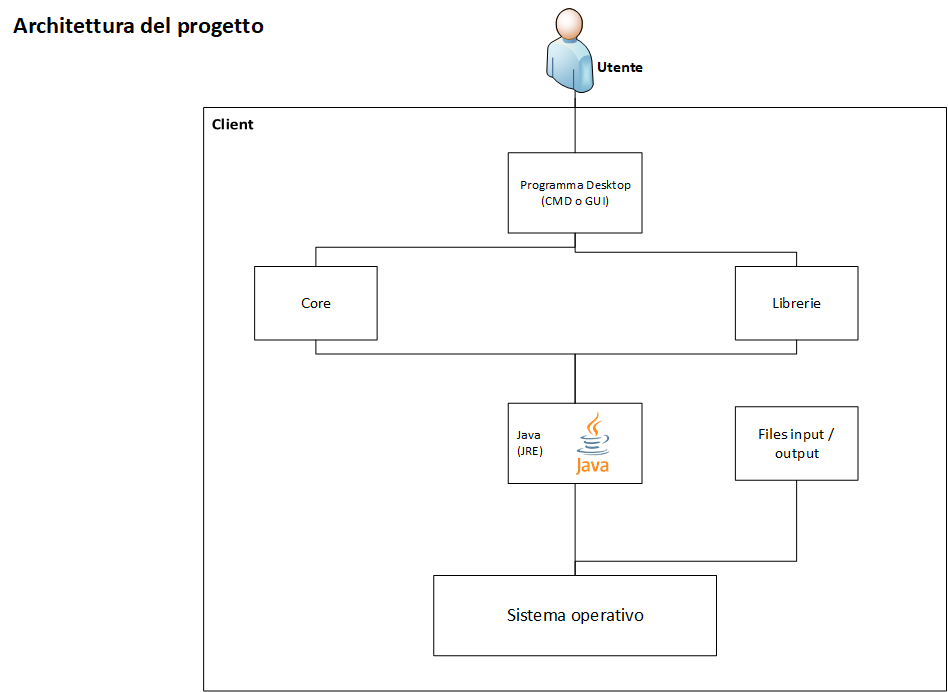


Figura 3 Architettura del progetto

## Diagramma di flusso

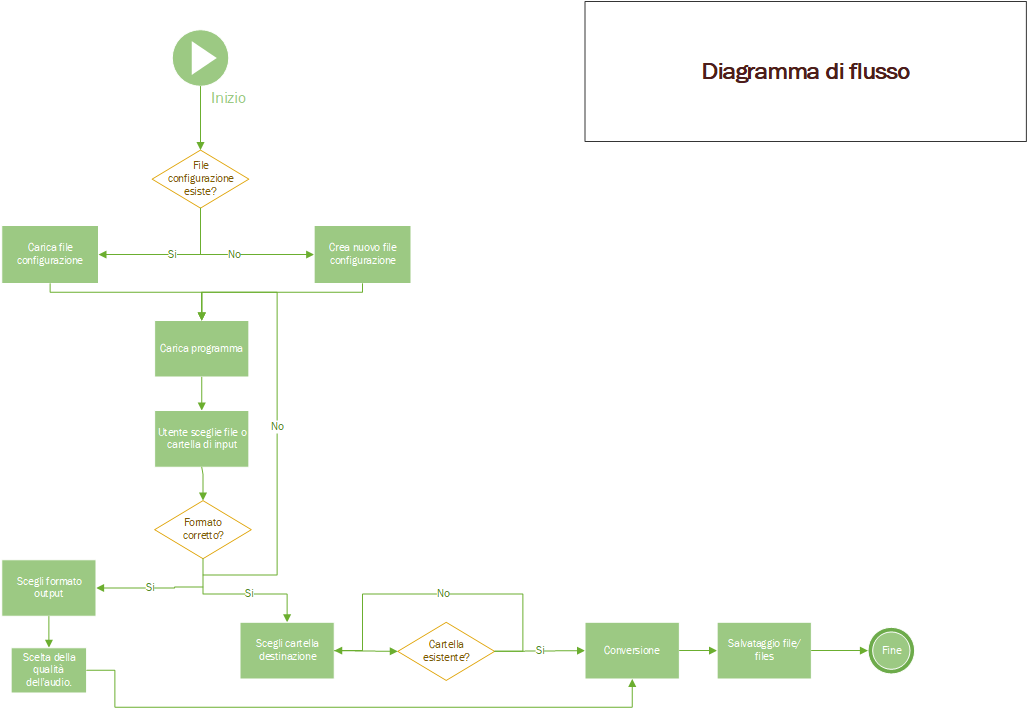


Figura Diagramma di flusso

## Design delle interfacce

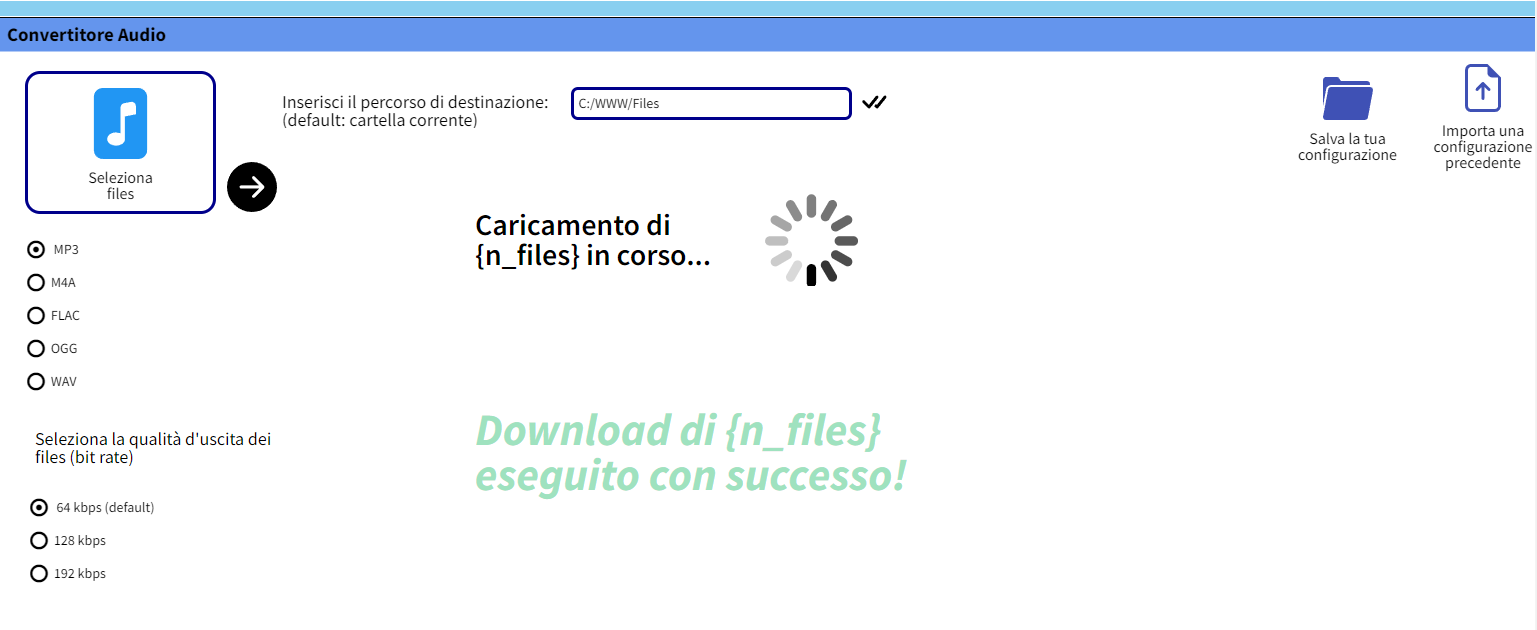


Figura Design dell'interfaccia principale

## Design delle classi

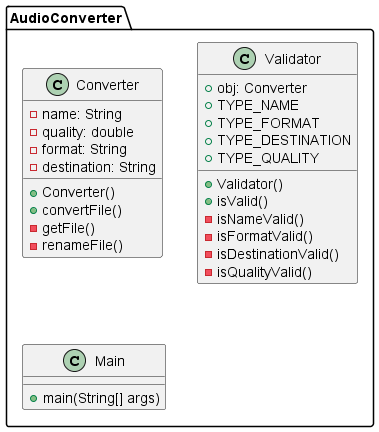


Figura Design delle classi iniziale

# Implementazione

Il progetto è stato creato utilizzando Java, con l’utilizzo di 3 classi:

* **Classe Main:** classe che si occupa di gestire tramite il classico metodo main() il ricevimento dei parametri attraverso gli “args” di Java.
* **Classe Converter**: crea l’oggetto **Converter** che mi permette attraverso i suoi attributi e i metodi privati e pubblici di prendere e memorizzare temporaneamente le informazioni del Main.
* **Classe Validator:** si occupa di controllare i dati dell’oggetto Converter e fa da ultimo step la parte principale quella di eseguire il comando passato dall’utente e passarlo a FFMPEG che farà la conversione per noi.

Poi ho scelto di utilizzare FFMPEG come programma esterno perché penso sia la soluzione più utile che fa al caso per una applicazione desktop; quindi, il mio programma di per sé fa da Wrapper attorno a FFMPEG.

## Classe Converter

Prima di tutto ho creato la classe Converter che contiene questi attributi:  
Immagine che contiene testo, Carattere, linea, schermata

Descrizione generata automaticamente

Figura attributi della classe Converter

Contiene anche tutti i getter e setter più un costruttore di default e un costruttore con tutti gli argomenti:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteFigura costruttori della classe Converter

Uno dei metodi più interessanti di questa classe è il metodo **generateFileName()**, che attraverso la classe **java.util.UUID** e il metodo **randomUUID()** crea un nome per il file unico, così se nel metodo **setOutputName()**, viene rilevato che il name di output scelto dall’utente è già nella directory di output viene chiamato il metodo **generateFileName().**

(Se il parametro **-overwrite** è settato a **true** ovviamente il file in output viene riscritto e questo metodo non viene chiamato)

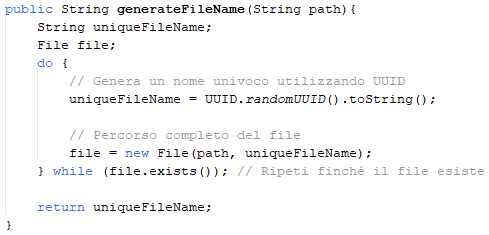


Figura Metodo generateFileName()

I metodi setter setOutputPath() e setPath() sfruttando la classe **java.io.File** e i suoi metodi (**exists(), isDirectory(), canWrite()**)per fare dei controlli sui percorsi inseriti dall’utente.

Uno dei metodi setter molto importante è anche il metodo **setFormat()** che attraverso un ciclo **foreach** riceve come parametro in entrata il nome del file in input e attraverso uno split() della stringa prende la parte dopo il **“.”** e controlla che quella stringa sia uguale a uno dei formati possibili (MP3, WAV, AAC, FLAC, M4A, OGG).

## Classe Validator

La classe Validator come detto prima si occupa di controllare ed eseguire i dati ricevuti dalla classe Converter. Ha 3 metodi principali molto auto esplicativi: **isValidConverter(), buildFFMPEGcommand(), executeCommand()**. Questi 3 metodi vengono racchiusi in un metodo unico chiamato **convertAudio()** che viene richiamato dalla classe Main.

* isValidConverter: ha il compito di controllare tutti gli input ricevuti tramite “args” dalla classe Main attraverso l’oggetto **Converter**, metodo strutturato principalmente su una base di “if”
* buildFFMPEGCommand: questo metodo letteralmente “costruisce” il comando FFMPEG, quindi, richiama il programma e costruisce la stringa da passargli.
* executeCommand(): questo programma attraverso la classe ProcessBuilder di Java

prende la stringa del comando e quindi crea il comando passo per passo.

Attraverso il metodo **directory()** imposto la cartella di lavoro di ProcessBuilder sul percorso di FFMPEG dato dalla variabile **pathExe.**

Alla fine richiamo una variabile di tipo **Process** che attraverso l’oggetto ProcessBuilder creato prima con tutte le caratteristiche impostate prima, chiama il metodo **start()** che lancia il comando.

La classe ha anche come attributo **pathExe**, una stringa che viene ricevuta grazie al metodo **System.getenv(String)** che prende dalla stringa passata la variabile d’ambiente relativa.

Questa variabile è fondamentale per prendere globalmente il percorso di **FFMPEG**, altrimenti

dovrei mettere il percorso assoluto di FFMPEG. Quindi l’utente prima di far partire il programma dovrà aver settato la variabile d’ambiente con il nome “**ffmpegPath**”.

## Classe Main

La classe Main è la classe principale che alla fine richiama tutte e 2 le classi.

Attraverso un ciclo **for** controlla ogni argomento con uno **switch:**

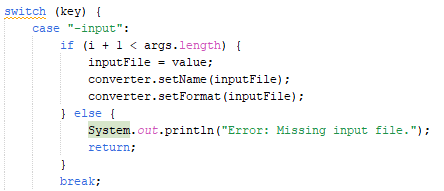


Figura switch fondamentale classe Main

Ogni parametro ha un funzionamento specifico e viene dopo passato alla classe **Converter**, dove viene impostato come attributo e alla fine controllato dalla classe **Validator.** L’unico parametro non gestito dalla classe **Converter** è il “-**overwrite**”.

Ci sono 3 parametri obbligatori da mettere per far funzionare correttamente la conversione, mentre gli altri sono facoltativi e se non inseriti verranno messi i valori di default:

* **-input**: riceve il file in input con la sua estensione (input.ext) se la stringa contiene spazi vuoti è necessario inserire le virgolette prima e dopo.
* **-outputformat**: riceve il formato di output di come si vorrà convertire il file. (Es. mp3)
* **-path**: riceve il percorso (assoluto o relativo) di dove si vuole prendere il file. Senza questo parametro il programma non sa dove andare a prendere il file. Se il percorso contiene spazi vuoti è necessario inserire le virgolette prima e dopo.

Di seguito tutti i parametri facoltativi (per evitare errori meglio inserirli in ordine di come vengono elencati)

* **-bitrate**: riceve un valore numerico intero per specificare la velocità di bit in quale si vuole convertire il proprio file.   
  Valore di default: 128.0  
  (NOTA: ovviamente se il bitrate è minore e la compressione del formato in output è attivata come per esempio nel formato MP3 il bitrate non verrà chiaramente aumentato)
* **-overwrite:** riceve un valore booleano (true | false) per dare la possibilità all’utente che se nel percorso in output c’è già un file con lo stesso nome quel file viene sovrascritto. Valore di default: false
* **-outputpath:** riceve il percorso della cartella di dove si vorrà salvare il file convertito.  
  Se il percorso contiene spazi vuoti è necessario inserire le virgolette prima e dopo.  
  Valore di default: C:\Users\utente
* **-outputname:** riceve una stringa di come si vorrà rinominare il file dopo la conversione.  
  Se **overwrite** è messo a false e nel percorso esiste già un file con lo stesso nome verrà generato dal programma un nome casuale tramite un UUID.  
  Valore di default: fileOutput

Alla fine del for e dello switch viene passato il tutto alla classe Validator tramite il suo metodo **convertAudio(converter, overwrite)** dove il parametro **overwrite** è fondamentale per controllare che l’utente non abbia scelto di sovrascrivere il file in output**.**

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-01 | **Nome** | Ricevere in input sia un file o una cartella |
| **Descrizione** | Il programma deve ricevere in input un file o una cartella tramite un parametro (-input) | | |
| **Prerequisiti** | Avere la classe Main e il relativo metodo main() con la gestione dei parametri per poter ricevere l’input | | |
| **Procedura** | 1. Avviare il programma 2. Inserire i parametri nella stringa di avvio del programma: -input file.mp3 3. Stampare alla fine del programma l’oggetto converter con il relativo file: converter.getName() | | |
| **Risultati attesi** | Output programma: **file.mp3** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-02 | **Nome** | Filtrare il tipo di formato corretto. |
| **Descrizione** | Il programma riceve in input un file con un tipo di formato e il suo compito è quello di controllare che sia uno dei formati supportati per la conversione. | | |
| **Prerequisiti** | Avere nel main() la ricezione del file (-input) | | |
| **Procedura** | 1. Avviare il programma 2. Inserire i parametri nella stringa di avvio del programma: -input file.mp3 3. Stampare il formato dall’oggetto converter: converter.getFormat(); | | |
| **Risultati attesi** | Output programma: **Not valid.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-03 | **Nome** | Conferma visuale della conversione |
| **Descrizione** | Il programma deve far vedere nell’output il nome ed il formato del file convertito. | | |
| **Prerequisiti** | La conversione del file deve essere funzionante e corretta. | | |
| **Procedura** | 1. Avviare il programma 2. Inserire i parametri e verificare che siano completamente corretti nella stringa di avvio del programma. 3. Controllare che il file sia stato convertito 4. Vedere stringa finale. | | |
| **Risultati attesi** | Output programma: **Conversion succeded. --> successfully converted " + name + " to " + nameOutput.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-04 | **Nome** | Consentire all’utente di scegliere il bitrate |
| **Descrizione** | Il programma deve poter ricevere in input il bitrate scelto dall’utente. | | |
| **Prerequisiti** | Avere nel main() la ricezione del parametro (-bitrate) | | |
| **Procedura** | 1. Avviare il programma 2. Inserire il parametro: -bitrate 256.0 3. Stampare il bitrate ricevuto: converter.getBitrate() | | |
| **Risultati attesi** | Output programma: **256.0** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-05 | **Nome** | Consentire scelta della destinazione dei files |
| **Descrizione** | Il programma deve poter ricevere in input la cartella di destinazione  (-outputpath) | | |
| **Prerequisiti** | Avere nel main() la ricezione del parametro (-outputpath) | | |
| **Procedura** | 1. Avviare il programma 2. Inserire il parametro: -outputpath C:\Test-Files\ 3. Verificare che la path sia corretta attraverso i metodi nella classe Converter() 4. Stampare la path ricevuta: converter.getOutputPath() | | |
| **Risultati attesi** | Output programma: **C:\Test-Files\** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-06 | **Nome** | Controllo se il file di output sia già presente nella cartella di destinazione ed eventualmente rinominare il file |
| **Descrizione** | Il programma deve consentire all’utente la scelta di rinominare il file o no se un file uguale è già presente nella cartella. | | |
| **Prerequisiti** | Avere nel main() la ricezione di tutti i parametri necessari | | |
| **Procedura** | 1. Avviare il programma 2. Inserire i parametri necessari 3. Verificare che la conversione sia andata a buon fine. 4. Controllare che nella directory ci sia un file uguale a quello in output | | |
| **Risultati attesi** | Output programma: **Warning: output name was not valid --> has been set to 5352558f-2744-4e46-943e-78fb61bd0bea.flac**  **Conversion succeded. --> successfully converted music.wav to 5352558f-2744-4e46-943e-78fb61bd0bea.flac** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-007  REQ-08 | **Nome** | Controllo e gestione degli eventuali errori |
| **Descrizione** | Il programma per ogni errore trovato come un mancato file in una cartella o un mancato parametro deve segnalarlo all’utente | | |
| **Prerequisiti** | Il programma deve riuscire a fare la conversione. | | |
| **Procedura** | 1. Avviare il programma 2. Inserire i parametri necessari ma mancare un parametro fondamentale come -outputformat 3. Verificare l’output del programma. | | |
| **Risultati attesi** | Output programma: **Error: Output format was not valid.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-008  REQ-10 | **Nome** | Controllo compatibilità dei file. |
| **Descrizione** | Il file in input può non essere un file in formato audio anche se ha un’estensione corretta | | |
| **Prerequisiti** | Il programma deve riuscire a fare la conversione e stampare eventuali errori. | | |
| **Procedura** | 1. Avviare il programma 2. Inserire come parametro -input un file di un qualsiasi tipo ma con estensione .mp3 (test.mp3) 3. Provare a eseguire la conversione. | | |
| **Risultati attesi** | Output programma: **[in#0 @ 00000279f51860c0] Error opening input: Invalid data found when processing input**  **Error opening input file D:\test-files\not-working.mp3.**  **Error opening input files: Invalid data found when processing input.** | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | **file.mp3** | Passato |
| TC-002 | **“Not valid”** | Passato |
| TC-003 |  | Passato |
| TC-004 | **256.0** | Passato |
| TC-005 | **C:\Test-files\** | Passato |
| TC-006 |  | Passato |
| TC-007 | **Error: Output format was not valid** | Passato |
| TC-008 |  | Passato |

## 

## Mancanze/limitazioni conosciute

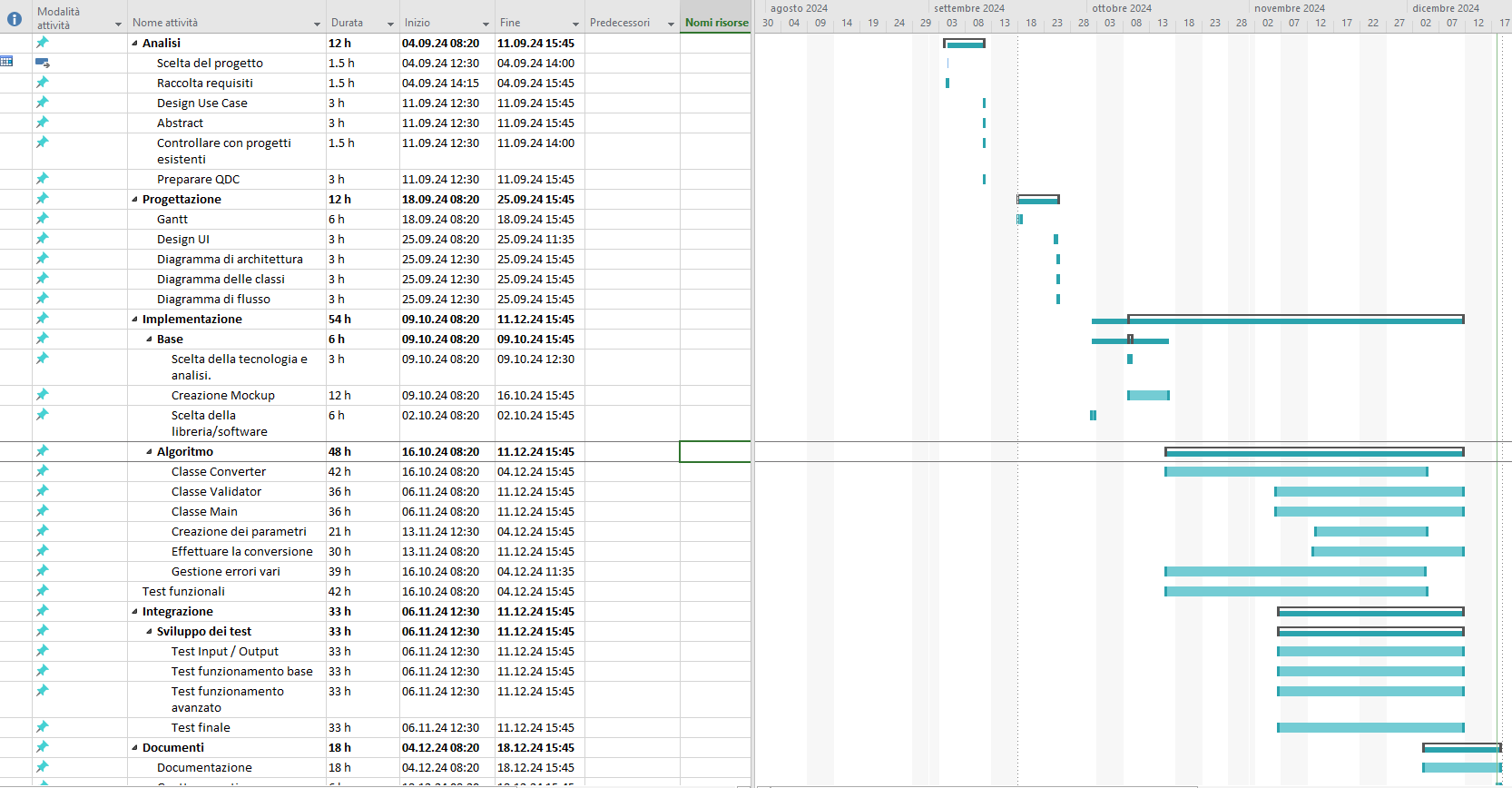
Ho riscontrato vari problemi come la creazione della GUI dopo la creazione del programma via CLI, quindi il REQ-11 non è stato rispettato.

Un altro problema riscontrato e che per far partire FFMPEG ho bisogno di impostare la cartella lavorativa tramite una variabile d’ambiente chiamata “ffmpegPath” e se questo l’utente non lo fa comporta problemi nel programma.

Il REQ-1 non è stato completamente rispettato visto che non sono riuscito a implementare la conversione anche su una cartella ma solamente su un file unico.

Di conseguenza il REQ-7 non può essere applicato.

# Consuntivo



CONSUNTIVO

Figura Gantt consuntivo

# Conclusioni

Il programma si è rivelato arduo da implementare, non come me l’aspettavo. Sono convinto che il progetto alla fine sia un buon prodotto, anche se vari requisiti sono stati messi da parte per colpa di vari fattori come il tempo o la difficoltà. Grazie a questo progetto mi sono reso conto di un aspetto fondamentale in quanto la “teoria”, ovvero la fase di preparazione del codice, sia una delle parti più complicate e che secondo me richiede molta attenzione.

Ho avuto anche la possibilità di sviluppare ulteriori conoscenze riguardo ai vari File Audio e le proprie particolarità.

## Sviluppi futuri

Molte migliorie che si possono applicare al prodotto sono già elencate come requisiti, visto che certi non ho avuto la possibilità di realizzarli o ho avuto delle difficoltà, come per esempio implementare il prendere come input una cartella e non un file unico.

## Considerazioni personali

Il progetto ha consolidato la parte di pianificazione che riguarda un progetto, ovvero saper gestire il tempo in modo corretto e saper individuare prima di fare un programma le parti principali e le parti meno importanti. Sfruttare al meglio la parte di pianificazione attraverso la creazione di tutti i diagrammi che abbiamo imparato è una delle cose che il progetto ti fa imparare. La parte di implementazione è stata molto più confusionaria di come l’avevo progettata e questo fa realizzare, come detto prima, che una delle fasi più importanti è la pianificazione.

# Bibliografia

## Sitografia

* <https://www.tutorialspoint.com/>
* <https://stackoverflow.com/>
* <https://mockflow.com/>
* <https://www.ffmpeg.org/ffmpeg.html>, *FFMPEG Documentation*
* <https://www.checkfiletype.com/check-meta>, *File MetaData*
* <https://www.geeksforgeeks.org/java-lang-processbuilder-class-java/>, *ProcessBuilder class in Java*
* <https://pixabay.com/sound-effects/search/audio-files/> *Free audio files*
* <https://trac.ffmpeg.org/wiki/audio%20types>, *Audio Files*

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| Bitrate | Velocità dei bit, nei file audio viene intesa come qualità di un file. |
| CLI | Command Line Interface: Terminale. |
| CMD | Il terminale di Windows. |
| GUI | Graphical User Interface: Interfaccia grafica. |
| IDE | Integrated Development Environment: una suite software che racchiude in un'unica interfaccia utente grafica i principali strumenti di sviluppo per codificare software. |
| Magical Number | Sequenza di Bytes che serve per identificare un tipo di file specifico. |

# Indice delle figure

[Figura 1: Use Case 8](#_Toc185427310)

[Figura 2 Gantt preventivo 9](#_Toc185427311)

[Figura 3 Architettura del progetto 11](#_Toc185427312)

[Figura 4 Diagramma di flusso 12](#_Toc185427313)

[Figura 5 Design dell'interfaccia principale 13](#_Toc185427314)

[Figura 6 Design delle classi iniziale 13](#_Toc185427315)

[Figura 7 attributi della classe Converter 15](#_Toc185427316)

[Figura 8 costruttori della classe Converter 15](#_Toc185427317)

[Figura 9 Metodo generateFileName() 16](#_Toc185427318)

[Figura 10 switch fondamentale classe Main 17](#_Toc185427319)

[Figura 11 Gantt consuntivo 24](#_Toc185427320)

# Allegati

* 4\_Diari
* 5\_Applicativo
* 1\_Qdc
* AudioConverter
* GanttPreventivo\_AudioConverter.mpp
* GanttConsuntivo\_AudioConverter.mpp
* Use-Case.vsdx