Progetto Ingegneria del Software Avanzata

Università degli Studi di Ferrara

Di Matteo Brina

Descrizione dell'applicazione

L'applicazione consiste in un client dotato di interfaccia grafica che si collega a un database MySQL e permette di riprodurre i brani al suo interno. La schermata iniziale contiene due pulsanti "Login" e "Registrati".

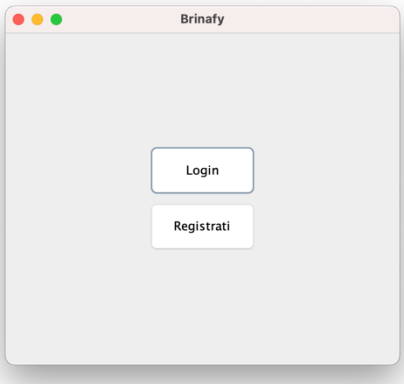


Figura 1: Schermata iniziale

Cliccando su "Registrati" viene aperta una schermata in cui sono presenti due campi di testo etichettati come "Username" e "Password" e due pulsanti "Indietro" e "Conferma". Cliccando sul pulsante "Indietro" si ritorna alla schermata precedente mentre, riempiendo i campi di testo con Username e Password e cliccando su "Conferma" possono presentarsi due scenari: Nel caso in cui l'username sia già presente all'interno del database, apparirà un banner con scritto "L'utente esiste già" e, cliccando su "OK" si rimane sulla schermata di registrazione; se invece l'username non è già presente del database apparirà un banner con scritto "Registrazione avvenuta con successo" e, cliccando su "OK" si ritornerà alla schermata iniziale.

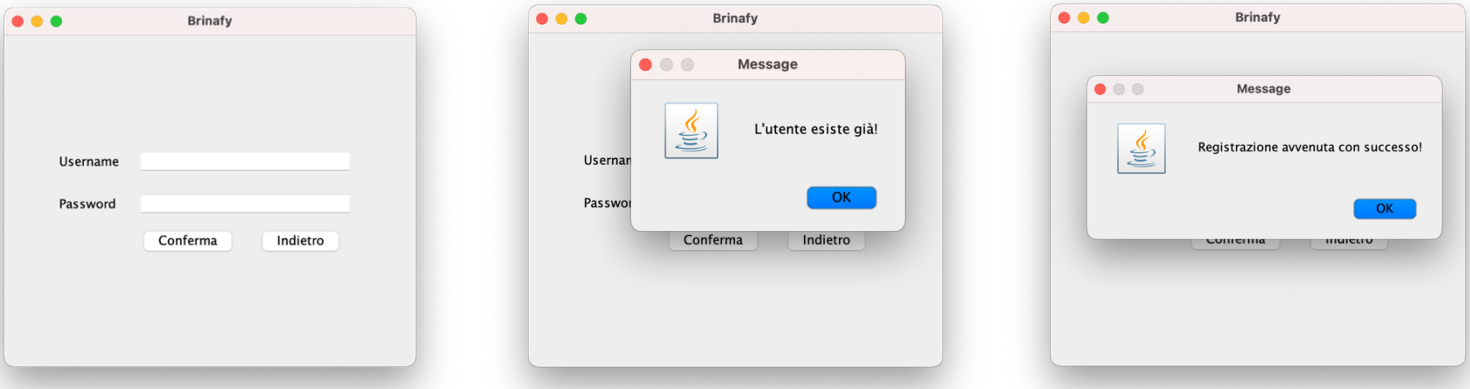


Figura 2: Schermata di registrazione

Cliccando su "Login" viene aperta una schermata analoga a quella di Registrazione, con la differenza che, cliccando su "Conferma" possono presentarsi 3 scenari: Nel caso in cui l'username

non sia presente nel database apparirà con banner con scritto “L’utente non esiste” e, cliccando sul pulsante “OK”, si rimane sulla schermata di login; nel caso in cui l’username sia presente nel database ma la password sia errata apparirà un banner con scritto “Password errata” e, cliccando sul pulsante “OK”, si rimane sulla schermata di login; nel caso in cui vengano inseriti username e password corretti, apparirà un banner con scritto “Login avvenuto con successo” e, cliccando sul pulsante “OK”, si passerà alla schermata principale di selezione dei brani.

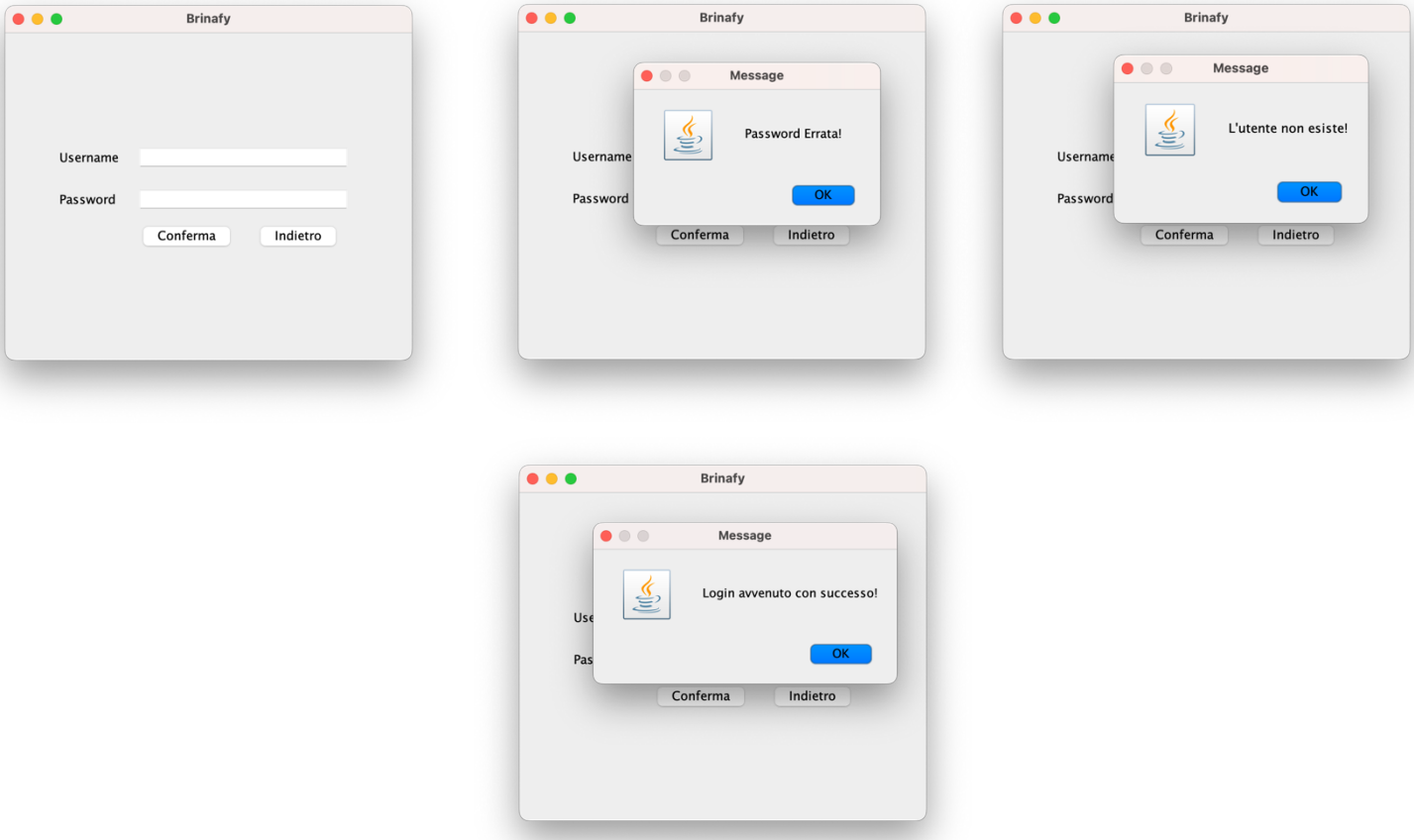


Figura 3: Schermata di Login

Questa schermata è composta in alto a sinistra da un’etichetta con scritto “Benvenuto (username)”, in alto dal pulsante “Logout” che permette, di effettuare il logout e tornare alla schermata iniziale, da un pulsante “STOP” disabilitato che si abiliterà una volta iniziata la riproduzione di un brano e la cui pressione consente di interromperla, e in alto a destra da un campo di testo e da un pulsante “conferma” tramite il quale si possono cercare brani che contengono all’interno del proprio titolo, artista o album la stringa cercata. Al centro della schermata è presente la lista di pulsanti corrispondenti alle canzoni il cui testo è costituito dai campi titolo, artista, album e ascolti corrispondenti a ogni canzone e alla cui pressione partirà la riproduzione del brano corrispondente e gli ascolti verranno incrementati di una unità. Nel caso in cui venga ricercata una stringa che non è contenuta in nessun campo di nessun brano apparirà un banner con scritto “nessun brano”

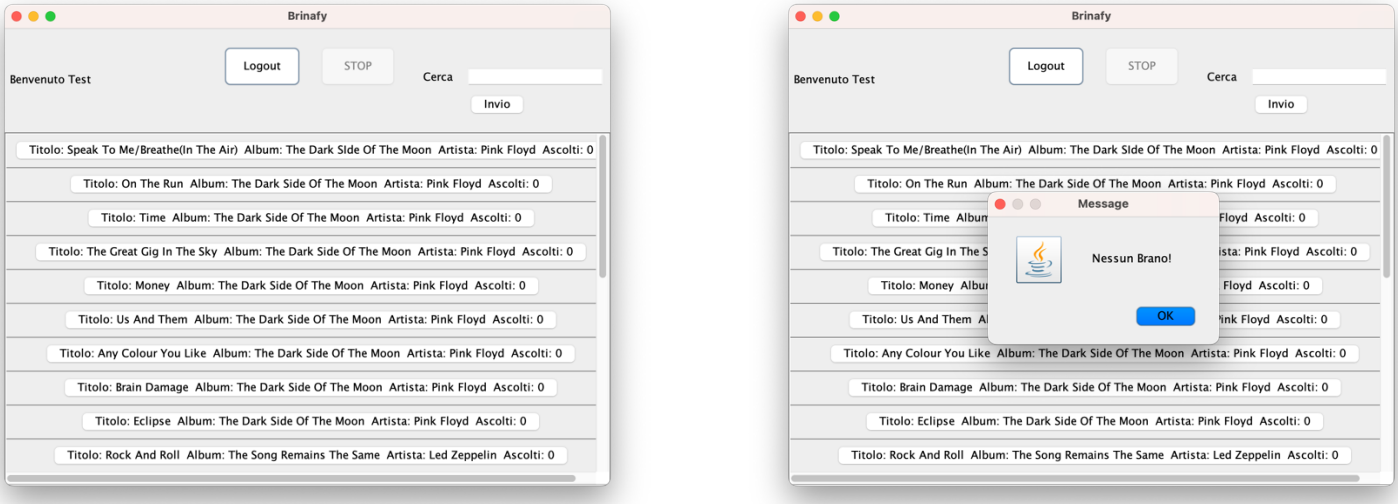


Figura 4: Schermata principale

Schema ER del database

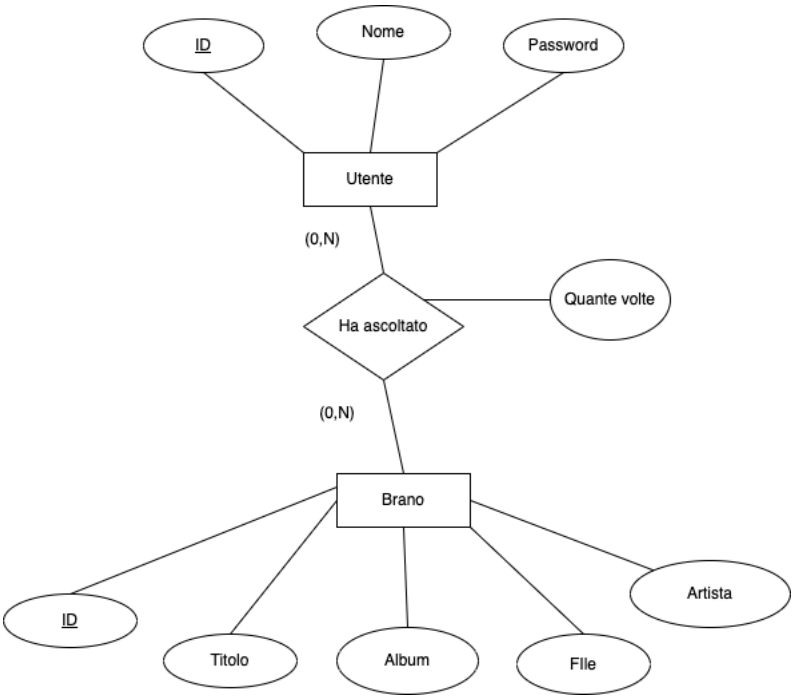


Figura 5: Diagramma ER del database

Utente

ID	Nome	Password

Brano

ID	Titolo	Album	Artista	File

Ha_ascoltato

ID_Utente	ID_Brano	Quante volte

Vincoli di chiavi esterna:

Ha_Ascoltato.ID_Utente —> Utente.ID
Ha_Ascoltato.ID_Brano —> Brano.ID

Lo schema relazionale è costituito da due entità forti “Utente” e “Brano” e da una relazione “Ha ascoltato” che li lega. L’attributo chiave di “Brano” e “Utente” è “ID” che è un intero auto incrementante che identifica univocamente ogni record nel DB. La relazione “Ha ascoltato” ha un attributo “Quante volte” che traccia il numero degli ascolti di ogni utente per ogni brano.

Sviluppo dell’applicazione

Il linguaggio utilizzato nello sviluppo dell’applicazione è stato interamente Java. Per lo sviluppo dell’applicativo si è cercato di restare il più fedele possibile al modello MVC.

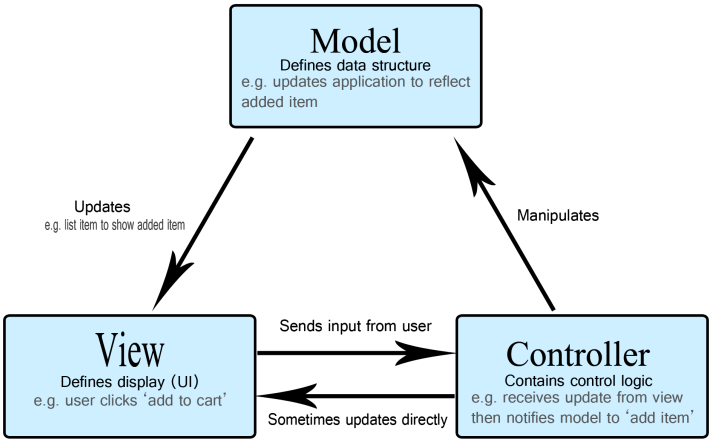


Figura 6: Pattern MVC

Il primo a essere sviluppato è stato il Model; si è proceduto a scrivere le classi che mappassero le entità forti del database ovvero la classe Brano e la classe Utente con i relativi attributi. Successivamente si è passati allo sviluppo delle classi DAOFactory e Configuration: la prima serve a instaurare la connessione con il database che verrà passata come parametro ai DataAccessObject di Brano e Utente e fornisce i metodi per fare commit e rollback delle transazioni e per chiudere la connessione; la seconda serve per parametrizzare il più possibile la connessione al database andando a fornire la stringa di connessione e il driver che, all’occorrenza possono essere modificati con facilità. Le classi UtenteDAO e BranoDAO contengono i metodi che accedono al database che a loro volta contengono le query necessarie al funzionamento dell’applicazione che vengono sottoposte tramite JDBC; un esempio può essere il metodo che crea una nuova tupla nella tabella Utente.

Le classi che costituiscono il controller invece sono ognuna associata a una view e contengono i metodi che vengono invocati a partire dalle interazioni dell’utente con quest’ultima e che la modificano dinamicamente; un esempio può essere il metodo makeButtonText della classe

MainPanelController che costruisce le stringhe dei pulsanti associati a ogni canzone dinamicamente in base alla canzone e agli ascolti effettuati da ogni utente.

Le view sono state sviluppate usando la libreria Java Swing; è stata creata una classe FirstFrame che contiene l'unico JFrame dell'applicazione e le classi FirstPanel, RegisterPanel, LoginPanel e MainPanel che contengono ognuno un JPanel corrispondente a una delle quattro schermate descritte all'inizio.

Scrittura dei test

Il codice è stato testato per intero avvalendosi di JUnit, un framework di unit testing per il linguaggio di programmazione Java. Per verificare che la copertura del codice fosse completa si è ricorsi a JaCoCo, una libreria di code coverage per Java.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
DAOFactory		92%		n/a	0	7	2	22	0	7	0	1
UtenteDAO		100%		100%	0	18	0	151	0	6	0	1
MainPanel		100%		100%	0	4	0	91	0	3	0	1
BranoDAO		100%		100%	0	10	0	93	0	5	0	1
MainPanelController		100%		100%	0	9	0	61	0	8	0	1
RegisterPanel		100%		n/a	0	2	0	53	0	2	0	1
LoginPanel		100%		n/a	0	2	0	53	0	2	0	1
FirstFrame		100%		n/a	0	1	0	31	0	1	0	1
FirstPanel		100%		n/a	0	2	0	29	0	2	0	1
FirstFrame.3.new ActionListener() {...}		100%		100%	0	3	0	12	0	2	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	14	0	2	0	1
Brano		100%		100%	0	18	0	30	0	17	0	1
MainPanel.TestPane		100%		n/a	0	2	0	11	0	2	0	1
Utente		100%		n/a	0	11	0	17	0	11	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	8	0	2	0	1
LoginPanelController		100%		100%	0	3	0	11	0	2	0	1
FirstFrame.3.2.new ActionListener() {...}		100%		100%	0	3	0	5	0	2	0	1
RegisterPanelController		100%		n/a	0	2	0	11	0	2	0	1
FirstFrame.3.new ActionListener() {...}		100%		100%	0	3	0	5	0	2	0	1
Configuration		100%		n/a	0	2	0	3	0	2	0	1
MainPanel.new ActionListener() {...}		100%		n/a	0	2	0	4	0	2	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	3	0	2	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	3	0	2	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	3	0	2	0	1
FirstFrame.new ActionListener() {...}		100%		n/a	0	2	0	3	0	2	0	1
FirstFrame.new WindowAdapter() {...}		100%		n/a	0	2	0	3	0	2	0	1
App		100%		n/a	0	3	0	5	0	3	0	1
DuplicatedObjectException		100%		n/a	0	1	0	2	0	1	0	1
WrongPasswordException		100%		n/a	0	1	0	2	0	1	0	1
MissingObjectException		100%		n/a	0	1	0	2	0	1	0	1
Total	5 of 2.811	99%	0 of 48	100%	0	124	2	730	0	100	0	30

Figura 7: Report JaCoCo

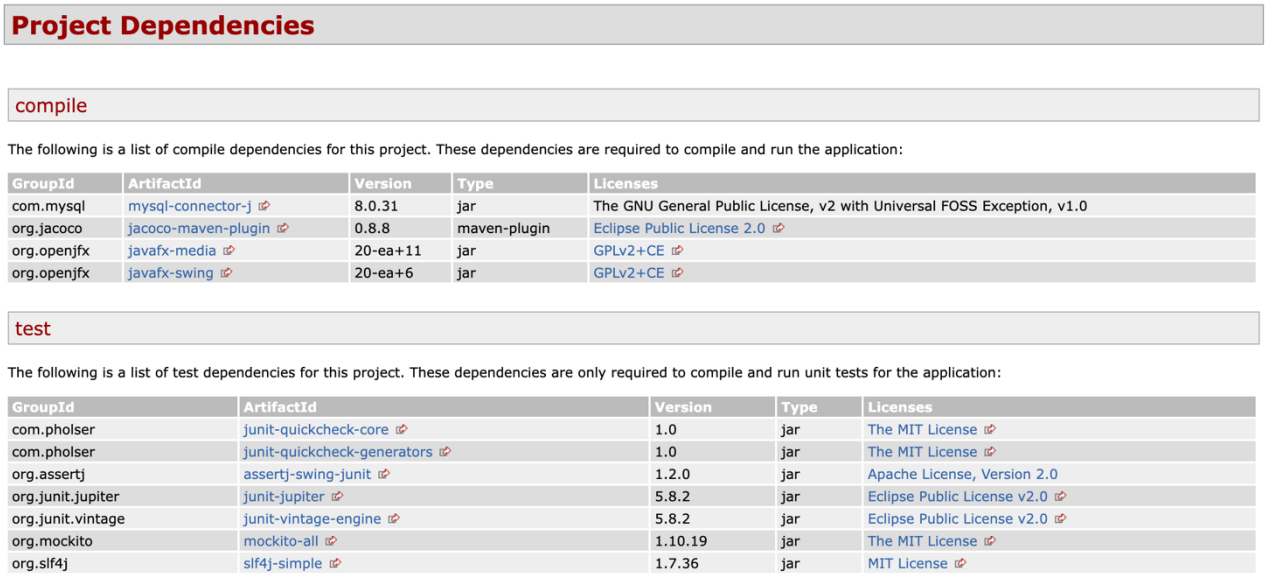
Come si può notare dal report il codice è stato testato per il 99%, ovvero 2806 istruzioni su 2811 e 48 branches su 48 (le 5 istruzioni mancanti non possono essere testate per limiti intrinseci del codice in quanto scritto in modo tale che certe eccezioni non possono essere lanciate).

Gestione delle dipendenze

Per gestire le dipendenze dell'applicazione si è fatto uso di Apache Maven, uno strumento di gestione di progetti software basati su Java e build automation. Maven usa un costrutto conosciuto come Project Object Model (POM); un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse. In questo modo

si separano le librerie dalla directory di progetto utilizzando questo file descrittivo per definirne le relazioni.

Maven effettua automaticamente il download di librerie Java e plug-in Maven dai vari repository definiti scaricandoli in locale o in un repository centralizzato lato sviluppo. Questo permette di recuperare in modo uniforme i vari file JAR e di poter spostare il progetto indipendentemente da un ambiente all'altro avendo la sicurezza di utilizzare sempre le stesse versioni delle librerie. Maven fornisce inoltre la possibilità di generare un mini-sito con informazioni riguardanti il progetto.



Project Dependencies

compile

The following is a list of compile dependencies for this project. These dependencies are required to compile and run the application:

GroupId	ArtifactId	Version	Type	Licenses
com.mysql	mysql-connector-j	8.0.31	jar	The GNU General Public License, v2 with Universal FOSS Exception, v1.0
org.jacoco	jacoco-maven-plugin	0.8.8	maven-plugin	Eclipse Public License 2.0
org.openjfx	javafx-media	20-ea+11	jar	GPLv2+CE
org.openjfx	javafx-swing	20-ea+6	jar	GPLv2+CE

test

The following is a list of test dependencies for this project. These dependencies are only required to compile and run unit tests for the application:

GroupId	ArtifactId	Version	Type	Licenses
com.pholser	junit-quickcheck-core	1.0	jar	The MIT License
com.pholser	junit-quickcheck-generators	1.0	jar	The MIT License
org.assertj	assertj-swing-junit	1.2.0	jar	Apache License, Version 2.0
org.junit.jupiter	junit-jupiter	5.8.2	jar	Eclipse Public License v2.0
org.junit.vintage	junit-vintage-engine	5.8.2	jar	Eclipse Public License v2.0
org.mockito	mockito-all	1.10.19	jar	The MIT License
org.slf4j	slf4j-simple	1.7.36	jar	MIT License

Figura 8: Porzione del mini-sito Maven che elenca le dipendenze

Controllo di versione e hosting

Per il controllo di versione si è ricorsi all'uso di Git integrato con GitHub la parte di hosting il cui link è https://github.com/matteobrina/ProgettoISA . Esistono in tutto 6 branch su github:

- Main: la versione del software così come è descritta nella relazione
- OldVersion: branch il cui ultimo commit è quello in cui si è raggiunti la prima versione stabile e funzionante del software in cui le entità Utente e Brano non erano legate dalla relazione "Ha ascoltato"
- Test: branch che deriva da OldVersion su cui venivano fatte prove e con cui veniva fatto il merge con main una volta che si arrivava a una nuova versione stabile.
- Linux: branch che deriva da main in cui sono stati risolti problemi di compatibilità con s.o. Linux.
- Player: branch che deriva da Linux, in cui è presente una variante dell'Applicazione che effettua la riproduzione dei Brani tramite il Player di default del s.o.
- DOCKER: branch che deriva da Linux con modifiche specifiche per far funzionare l'applicazione containerizzata

È inoltre stata effettuata la configurazione del file Gitignore che specifica i file intenzionalmente non tracciati che Git dovrebbe ignorare.

Deployment

Per il deployment semplificato dell'applicazione è stato utilizzato Docker, un software libero progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili chiamati container, con l'obiettivo di semplificare i processi di deployment di applicazioni software. È stato creato il Dockerfile che contiene le istruzioni per creare l'immagine del container

```
FROM ubuntu:22.04

WORKDIR home/isa

COPY pom.xml .
COPY src src/

RUN apt update && apt upgrade && apt install -y openjdk-17-jdk openjdk-17-jre && apt
install -y maven && apt install -y libxtst6 libxrender1 libxi6 && apt install -y
rhythmbox
```

A partire dall'immagine ubuntu: 22.04 viene generata la nuova immagine creando la directory home/isa all'interno della quale vengono copiati il file pom.xml e la cartella src e tutte le sue sottodirectory nella nuova cartella src. In seguito, vengono aggiornati i repository e vengono installate la Jdk e la Jre 17, maven, alcune dipendenze per l'interfaccia grafica e il riproduttore musicale predefinito di ubuntu rhythmbox.
In questo caso l'applicazione per potersi avvalere dell'interfaccia grafica ha bisogno di collegarsi al server X Window System della macchina ospitante.
Link all'immagine Docker:
https://hub.docker.com/layers/matteobrina99/isa/2022/images/sha256-bb21868cf083ef89fe4814cf78aeb13a763cd3fe803c10f78930254edc79f6c0?context=repo

È stato inoltre creato uno script bash per avviare agevolmente il container esportando le variabili di sistema necessarie al collegamento con il server X Window System.

```
#!/bin/bash
IP=$(ifconfig en0 | grep inet | awk '$1=="inet" {print $2}')
xhost +
sudo docker container run -e DISPLAY=$IP:0 -v /tmp/.X11-unix:/tmp/.X11-unix -it isa:2022
bash
```