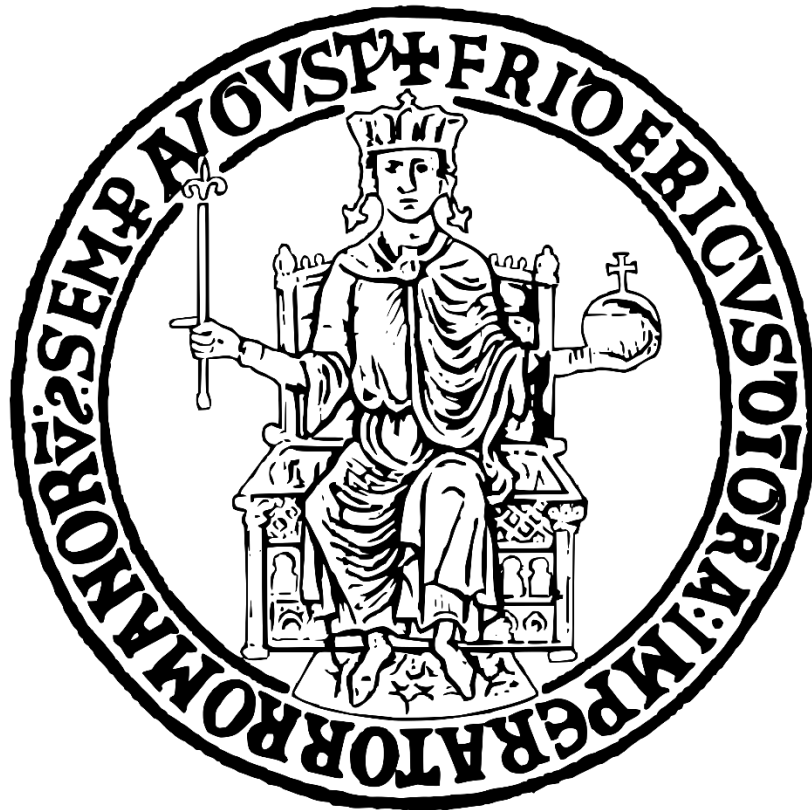


**UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II**

**SCUOLA POLITECNICA E DELLE SCIENZE DI BASE**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE**



**BISCIOR SOUND**

**SISTEMA DI GESTIONE LIBRERIA MUSICALE DI UN GRUPPO DI UTENTI**

**TRACCIA 1**

Vincenzo Meloni : N86003897

Loris Zannini : N86003744

Anno Accademico 2021/22

Docente:

Porfirio Tramontana



# Capitolo 1

## Introduzione

### 1.1 Descrizione del progetto

BisciorSound è un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI , per la gestione delle librerie musicali di un gruppo di utenti. Il sistema permette di gestire le singole tracce degli utenti, gli album di appartenenza ,gli ascolti delle tracce, i remastering , le cover , una lista di preferiti delle tracce e delle cover che l'utente salva. L'utente può accedere nella home, dove andrà ad interagire con la libreria musicale , effettuando il log-in o registrandosi (qualora non fosse dotato di account) inserendo il proprio username e la password. L'Admin è già registrato nel sistema e vi accederà tramite un'interfaccia a parte che lo reindirizzerà nella home Admin , permettendogli di effettuare le stesse operazioni di un normale utente con l'aggiunta di una sezione riservata esclusivamente ad egli, le "Opzioni Admin" , dentro le quali si potranno visionare tutti gli utenti registrati nel database (inclusi gli admin). Inoltre un amministratore avrà la possibilità di visionare , tramite una ricerca testuale , gli utenti che hanno effettuato più ascolti data una determinata traccia e potrà individuare la fascia oraria dove un preciso utente ha effettuato più ascolti differenziando tracce e cover (se nel sistema è presente la cover di un determinato brano).

### 1.2 Requisiti identificati

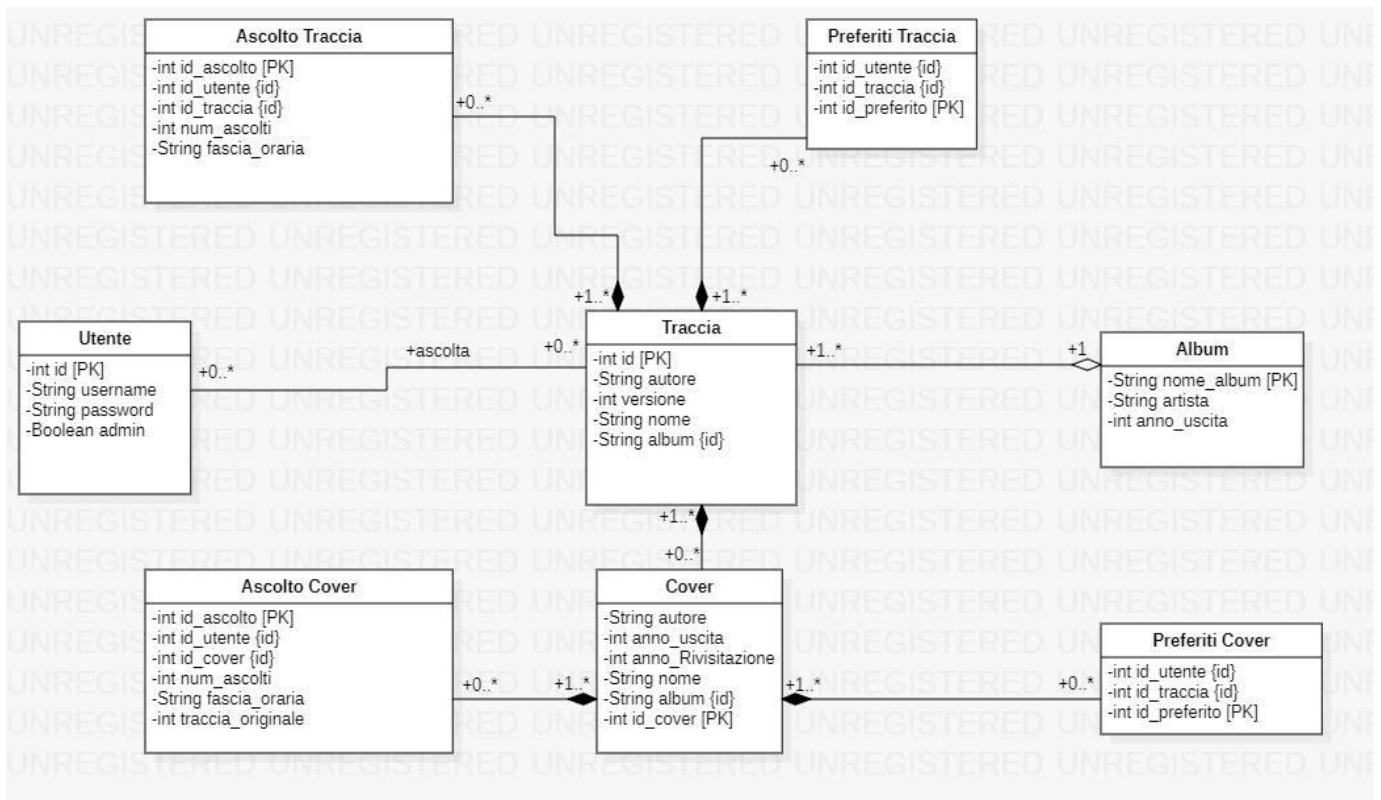
In primo luogo abbiamo suddiviso il progetto in pacchetti , ovvero: "ConfigurazioneDB, Controller, DAO , GUI, ImplementazioniPG\_DAO, Model". Successivamente nel package di configurazione abbiamo creato una classe "ConnessioneDB", che oltre a connettere l'applicativo con la base di dati, genera un' istanza della connessione da passare in tutte le classi in cui vengono effettuate query sql. Il primo requisito individuato è stato quello di creare una classe "Utente" nel package "Model"; all'interno di essa abbiamo dichiarato le variabili in maniera coerente con la tabella "Utente" situata all'interno del database. Una volta definita la classe Utente , abbiamo creato una sua interfaccia nel package "DAO" con il nome di "UtenteDAO", inserendo le signature delle funzioni implementate nel package "ImplementazioniPG\_DAO" con il nome di "UtenteImplementazionePG\_DAO()" . Lo stesso procedimento è stato eseguito per tutte le entità presenti nel package Model. Tutte le implementazioni e i DAO vengono invocate nel Controller in modo tale da avere un unico package che gestisce l'intero funzionamento del sistema. L'Utente (che ha uno username univoco), una volta effettuato il log-in o la registrazione, ha la possibilità di visionare la lista delle tracce presenti, gli album, le cover e gli ascolti . Per ogni traccia l'utente può scegliere se riprodurla o aggiungerla ai preferiti. Le tracce hanno un proprio album di appartenenza visionabile nella sezione "Album". Nella sezione "Preferiti" , l'utente può vedere le tracce salvate e in caso rimuoverle. Esistono anche le Cover, che hanno le stesse funzionalità delle Tracce e che si riferiscono a Tracce già presenti nel sistema. Nella sezione "Ascolti" l'utente può vedere le tracce che ha riprodotto con il numero di ascolti affiancato per ogni traccia e/o cover ascoltata. L'admin è in grado di visionare tutto ciò che vede un

normale utente ed in più può fare altre operazioni: data una certa traccia o cover, può controllare gli utenti che hanno effettuato più ascolti. Inoltre potrà effettuare una ricerca immettendo un nome utente presente nel sistema , andando a differenziare per tracce e cover la fascia oraria in cui l'utente digitato ha effettuato più ascolti. L'admin ha a portata di mano una lista completa di Utenti e amministratori registrati nel sistema.

# Capitolo 2

## Progettazione Concettuale

### 2.1 Class Diagram

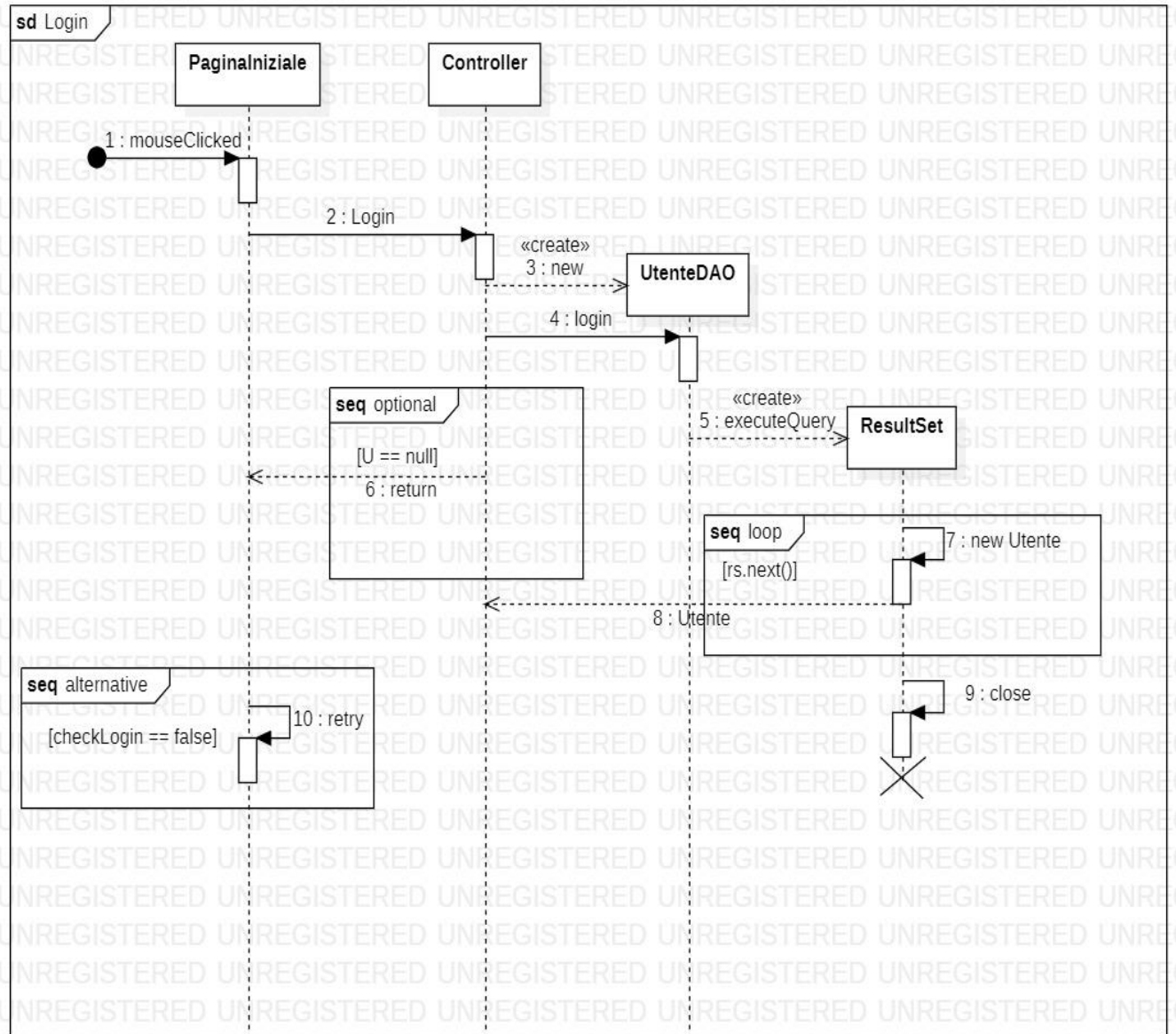


The image displays a series of UML diagrams for a music management system, organized into four main sections: GUI, Model, Controller, and Implementation (DAOs).

- GUI Diagram:** Shows the user interface components. It includes a **GUIMAN** class at the top, which interacts with **AdminLogin**, **UserLogin**, **AdminCover**, and **UserLoginCover**. Below these are **PageManager**, **HomeAdmin**, **HomeUser**, **SecondAdmin**, **SecondUser**, **FavoriteAdmin**, and **FavoriteUser**. Further down are **RegisterAdmin**, **RegisterUser**, **ManagerAdmin**, **ManagerUser**, **LoginAdmin**, **LoginUser**, **LoginFavoriteAdmin**, and **LoginFavoriteUser**. At the bottom are **AdminCover**, **AdminFavoriteCover**, **UserCover**, and **UserFavoriteCover**. A **Image** class is also shown.
- Model Diagram:** Shows the data model classes. It includes **AdminFavorite**, **Admin**, **Favorite**, **UserFavorite**, **Album**, **Cover**, and **AdminCover**. Relationships are indicated by arrows with multiplicity and association names.
- Controller Diagram:** Shows the control logic. It includes a **Controller** class with methods for **AdminFavorite**, **Admin**, **Favorite**, **UserFavorite**, **Album**, **Cover**, and **AdminCover**. The methods are implemented using DAOs and the Model classes.
- Implementation (DAOs):** Shows the data access objects. It includes **AlbumDAO**, **AdminCoverDAO**, **AdminFavoriteDAO**, **CoverDAO**, **FavoriteDAO**, **FavoriteUserDAO**, **UserDAO**, and **UserFavoriteDAO**. Each DAO has a **Connect** method and a **Disconnect** method. A **ConfigPrivate DB** and **Connection\_DB** class are also shown.

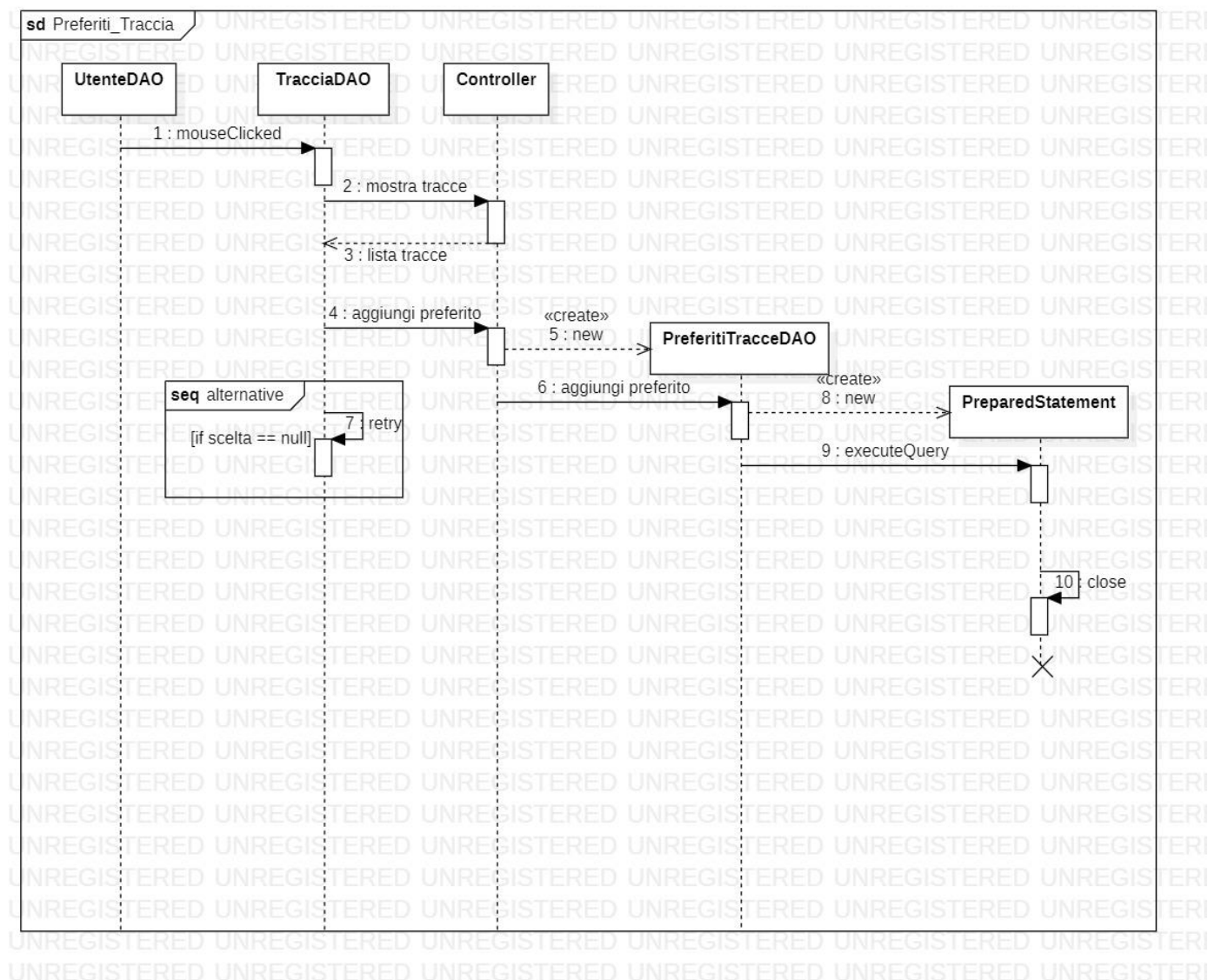
## 2.3 Sequence Diagram

### 2.3.1 Log-In



Quando viene cliccato il bottone di log-in, viene creato un Utente tramite la funzione `userData()` gestita dal controller, che con l'ausilio di una funzione booleana verifica se le credenziali sono corrette andando ad eseguire una query finché il `resultSet` risulta non nullo, altrimenti si rimane sulla schermata iniziale avvisando l'utente che le credenziali sono errate tramite uno `JOptionPane`.

### 2.3.2 Preferiti Traccia



Nel momento in cui un utente , scegliendo una traccia, vuole aggiungerla ai preferiti, viene eseguita la funzione `aggiungi_preferito()`, gestita dal controller, che permette di inserire la traccia selezionata nei Preferiti tramite il `PreparedStatement` , che esegue la query di INSERT ed inserisce la traccia nei preferiti se quest'ultima non è già presente, altrimenti viene sollevata un'eccezione.