INDICE

1. Introduzione

Il seguente documento è volto all'identificazione e al raffinamento di oggetti del dominio della soluzione definiti durante la realizzazione dei sottosistemi presente all'interno del System Design. Per ciascun oggetto identificato saranno definite le signature, la visibilità e i contratti, ovvero le condizioni da soddisfare per permettere l'invocazione di una operazione e condizione sotto le quali tali comportamenti(operazioni) potrebbero generare eccezioni. Sarà quindi realizzata la specifica delle interfacce tramite attività di: identificazione di attributi mancanti e operazioni, specifica delle signature e della visibilità, individuazione di invarianti, pre-condizioni e post-condizioni. Per lo sviluppo del software si è deciso di concentrarsi su i seguenti sottosistemi: Gestione Registrazione, Gestione Account, Gestione Autenticazione, Gestione Eventi, Amministrazione Piattaforma. Non è stato trattato il sottosistema di Gestione Annunci, i cui riferimenti sono all'interno del RAD e del SDD.

1.1 Object Design trade-offs

Prestazione vs costo:

Il sistema dovrà essere in grado di garantire: la rapidità durante l'esecuzione delle funzionalità, la gestione di un elevato numero di utenze e la memorizzazione dei dati permanenti sfruttando l'utilizzo di un database. Le tecnologie utilizzate per lo sviluppo del sistema non prevedono pagamenti per il loro uso, tuttavia coloro che intendono usufruire del software dovranno disporre di dispositivi mobili che supportino la tecnologia Android, pertanto in caso di batteria quasi scarica non è garantito il corretto funzionamento del sistema.

Usabilità vs affidabilità:

Il sistema sarà sviluppato attraverso una modalità user-friendly che garantisce l'intuitività durante l'utilizzo dell'applicazione. Inoltre, il software supporterà la gestione degli input errati da parte dell'utente, delle eccezioni e delle situazioni di errore, tramite apposite schermate. Il sistema sarà inoltre in grado di ripristinare i campi precompilati di un form in caso di interruzione forzata durante la compilazione.

Leggibilità vs tempo:

Il sistema sarà implementato tramite appositi moduli che garantiscono la facilità di interazione con il codice.

Sicurezza vs efficienza: il sistema sarà sviluppato in un tempo limitato, pertanto la sicurezza è garantita tramite autenticazione, gestita con l'opportuno inserimento di mail e password.

1.2 Linee guida per la documentazione delle interfacce

I nomi delle classi dovranno iniziare con la lettera maiuscola, e tale nome dovrà essere pertinente al concetto che rappresenta. Le classi che gestiscono la logica di controllo saranno file java.

- I nomi delle variabili dovranno iniziare con la lettera minuscola, in caso di parole composte si utilizzerà la notazione "Camel Case".
- I nomi delle variabili dovranno essere descrittivi della variabile che si intende rappresentare.
- L'uso di variabili locali all'interno di un metodo prevede l'inizializzazione.
- Le variabili dichiarate di tipo final dovranno essere scritte in caratteri maiuscoli; in caso di parole composte è previsto l'utilizzo del carattere "_" come separatore.
- Il nome di un metodo deve iniziare con una lettera minuscola, in caso di parole composte si utilizzerà la notazione "Camel Case".
- Il nome che descrive un metodo deve essere un verbo.
- Le classi definite per i layout, fanno parte nelle categorie "resource" e avranno una estensione .xml, pertanto il nome dovrà iniziare con una lettera minuscola e in caso di parole composte è prevista la separazione utilizzando "_".
- I nomi dei metodi che effettuano l'accesso alle variabili di una classe saranno del tipo "getNomeVariabile()"; i nomi dei metodi che settano valori alle variabili di una classe saranno del tipo "setNomeVariabile()".
- I metodi si distinguono in due categorie: quelli privi di un valore di ritorno ("void") e quelli che restituiscono un valore in base al tipo (int,String, boolean ecc).

1.3 Definizioni, acronimi, abbreviazioni

- **SDD:** System Design Document
- **RAD:** Requirements Analysis Document
- **XML:** Extensible-Markup-Language, descrittore utilizzato per creare all'interno di AndroidStudio le interfacce utente
- **Java:** Linguaggio di programmazione ad alto livello orientato agli oggetti, utilizzato all'interno di AndroidStudio per gestire la logica di controllo
- **DBMS:** Sistema di gestione di basi di dati
- **MySQL:** Sistema di gestione di basi di dati relazionale
- **JSON** : JavaScript Object Notation, utilizzato per comunicare tra client e server.

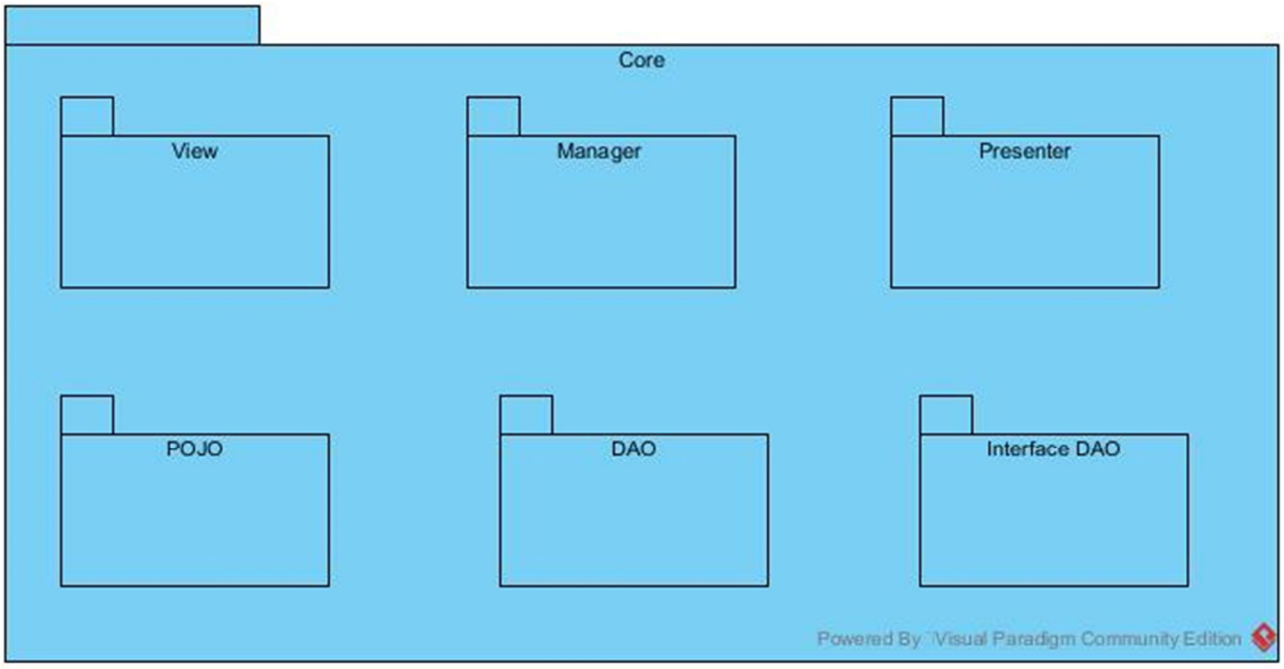
2. Packages

2.1 Introduzione

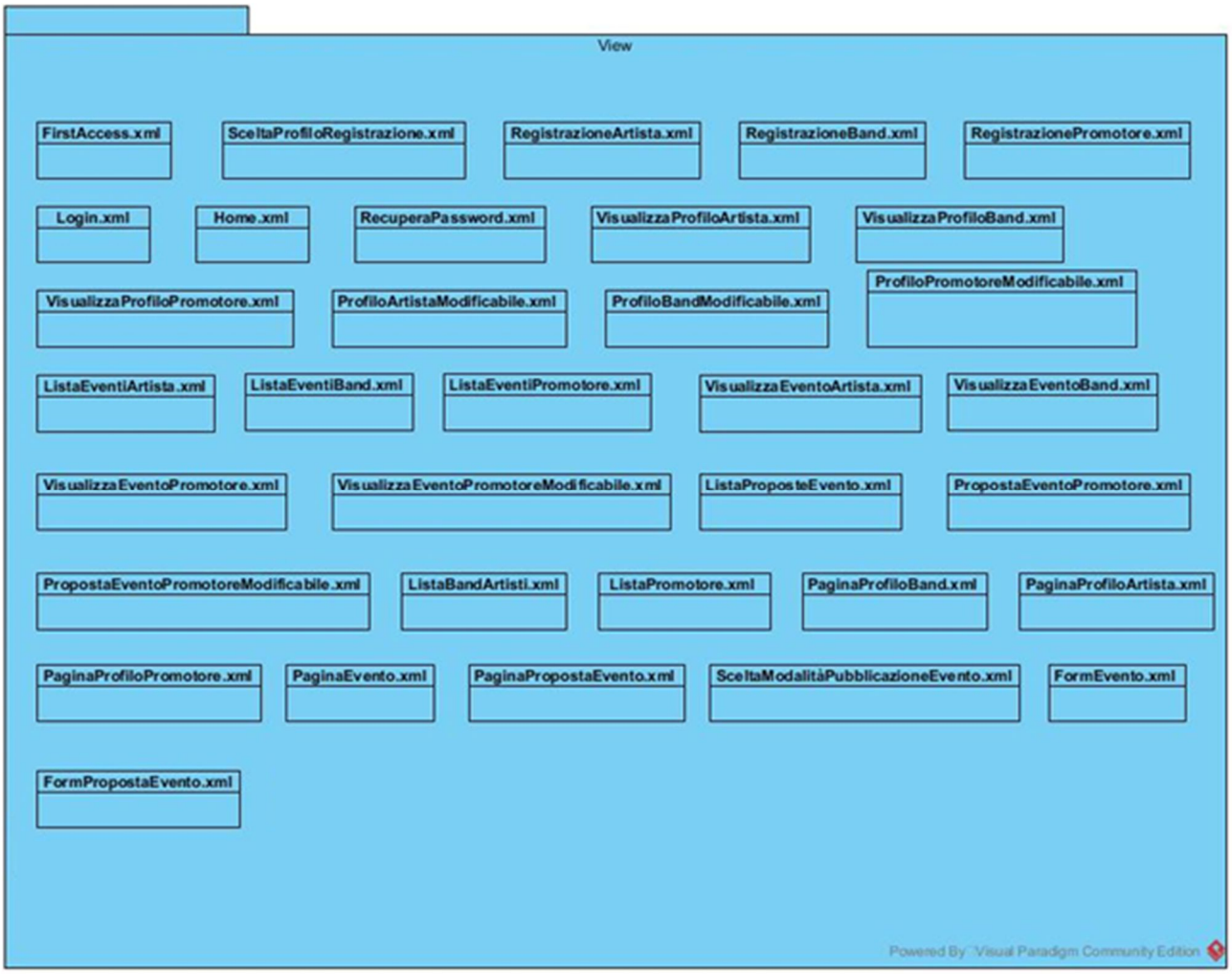
L'architettura del software realizzato è di tipo three-tier ovvero presenta 3 livelli di astrazione: Presentation layer, Application layer, Storage layer. All'interno di ogni layer sono presenti diversi packages, ciascuno volto ad un determinato scopo e che contiene una serie di classi per realizzarlo.

Presentation Layer	Comprende tutte le interfacce utente che includono i boundary objects individuati nel RAD. L'interazione con l'utente avverrà tramite un Web Server attraverso l'interazione con i layout.
Application Layer	Contiene tutti gli oggetti relativi alla logica di controllo e all'elaborazione dei dati tramite comunicazione con lo Storage Layer per generare contenuti dinamici e accedere ai dati persistenti. All'interno di questo layer avviene la gestione dei seguenti sottosistemi: • **Gestione Registrazione** • **Gestione Autenticazione** • **Gestione Account** • **Gestione Eventi** • **Amministrazione Piattaforma**
Storage Layer	Comprende le operazioni di: inserimento, aggiornamento e interrogazione sugli oggetti persistenti memorizzati all'interno del DataBase, l'interazione tra Storage Layer e il DataBase avverrà tramite il DBMS MySQL.

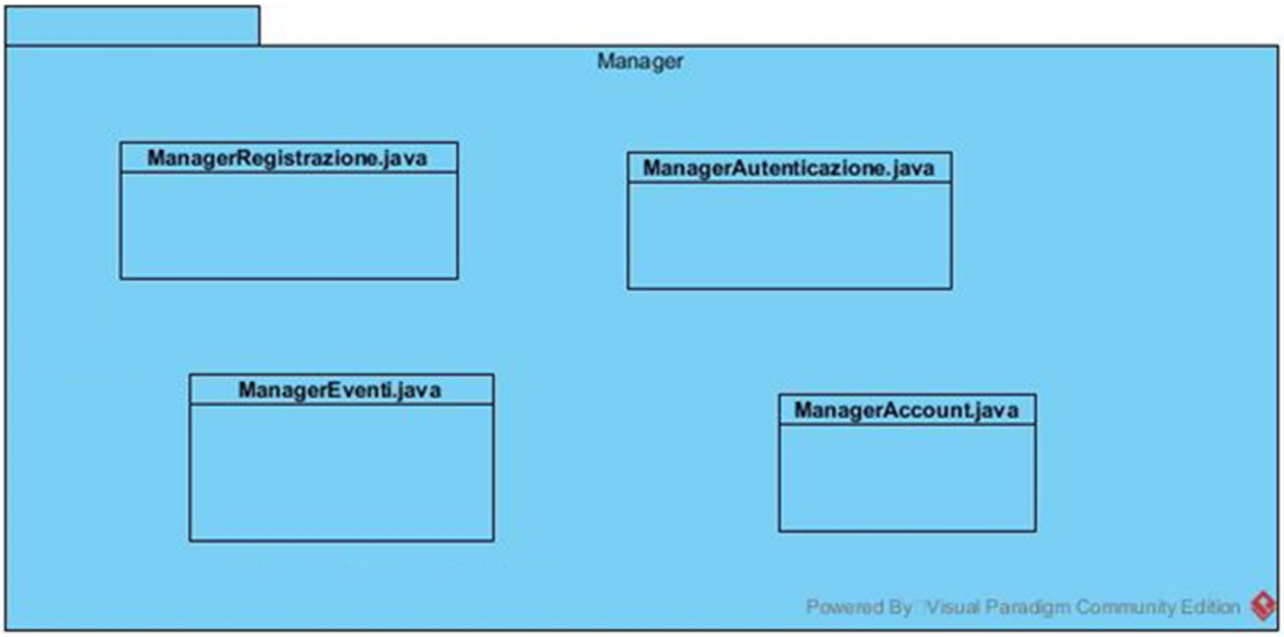
2.2 Packages core



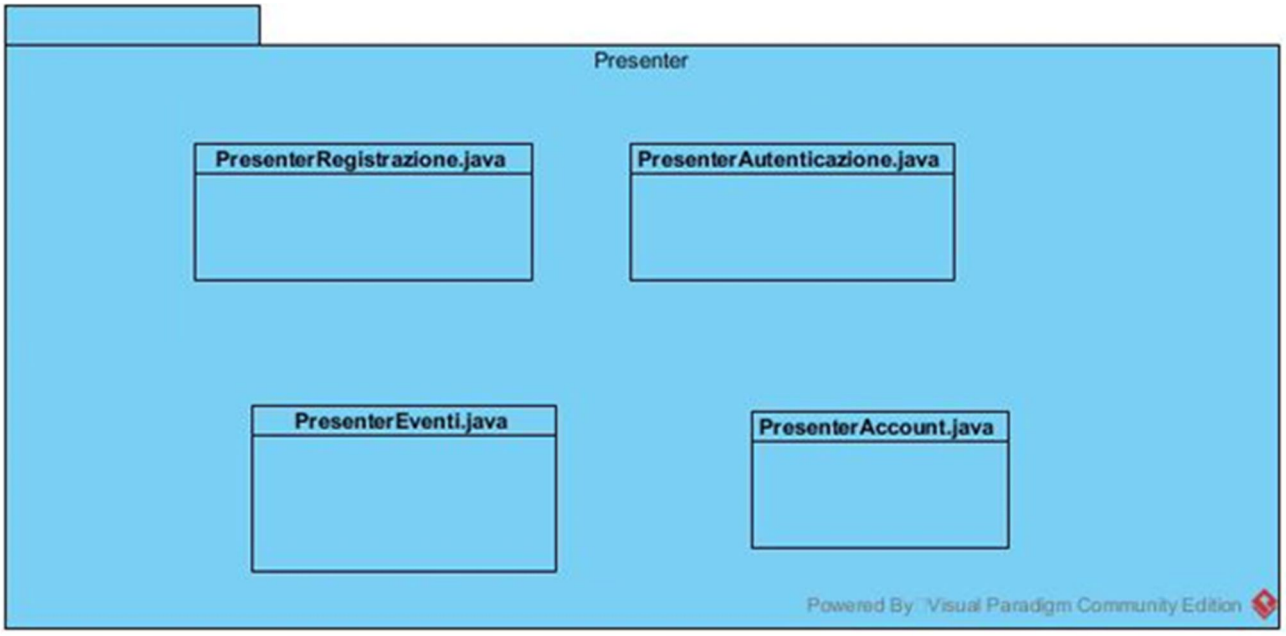
2.3 Package View



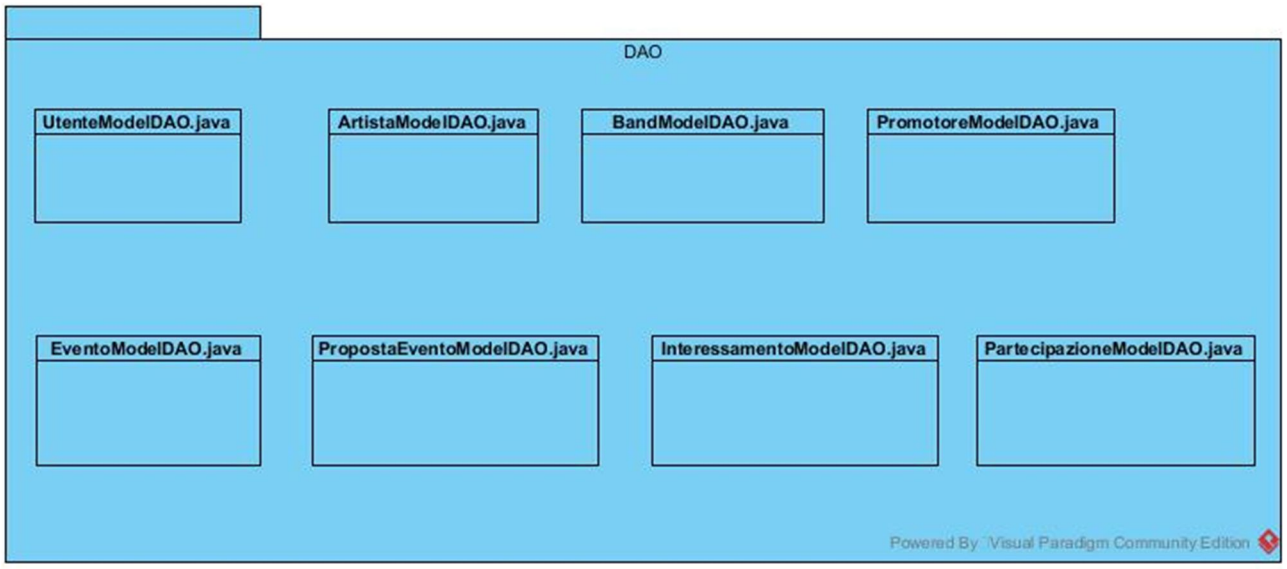
2.4 Package Manager



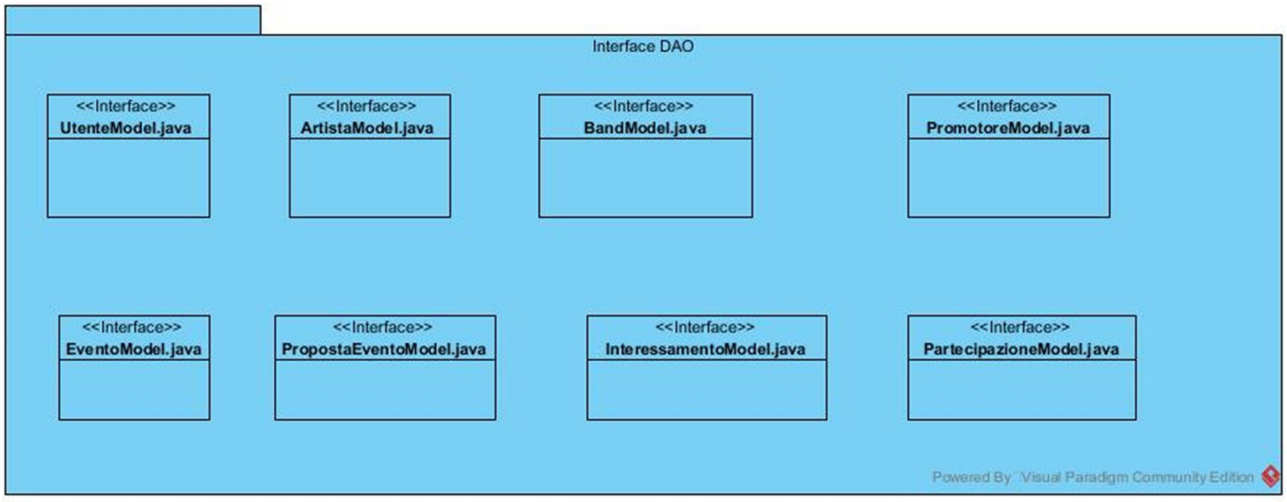
2.5 Package Presenter



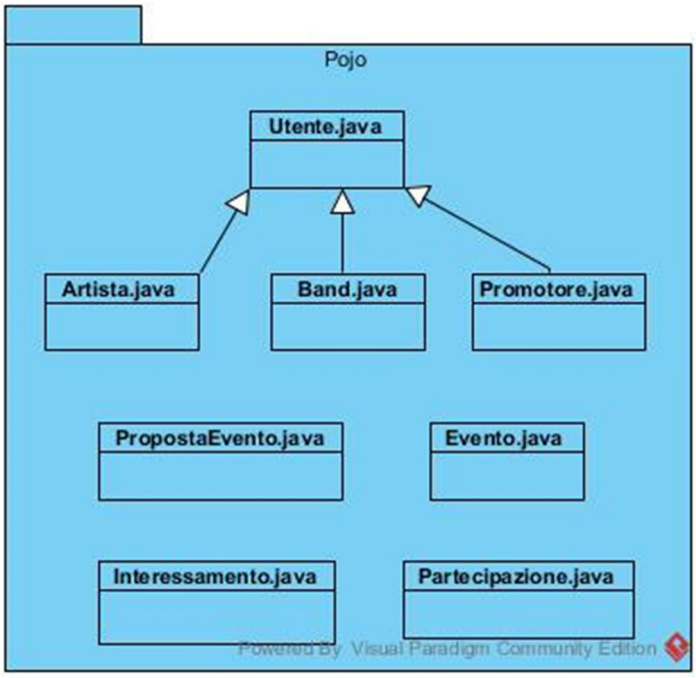
2.6 Package DAO



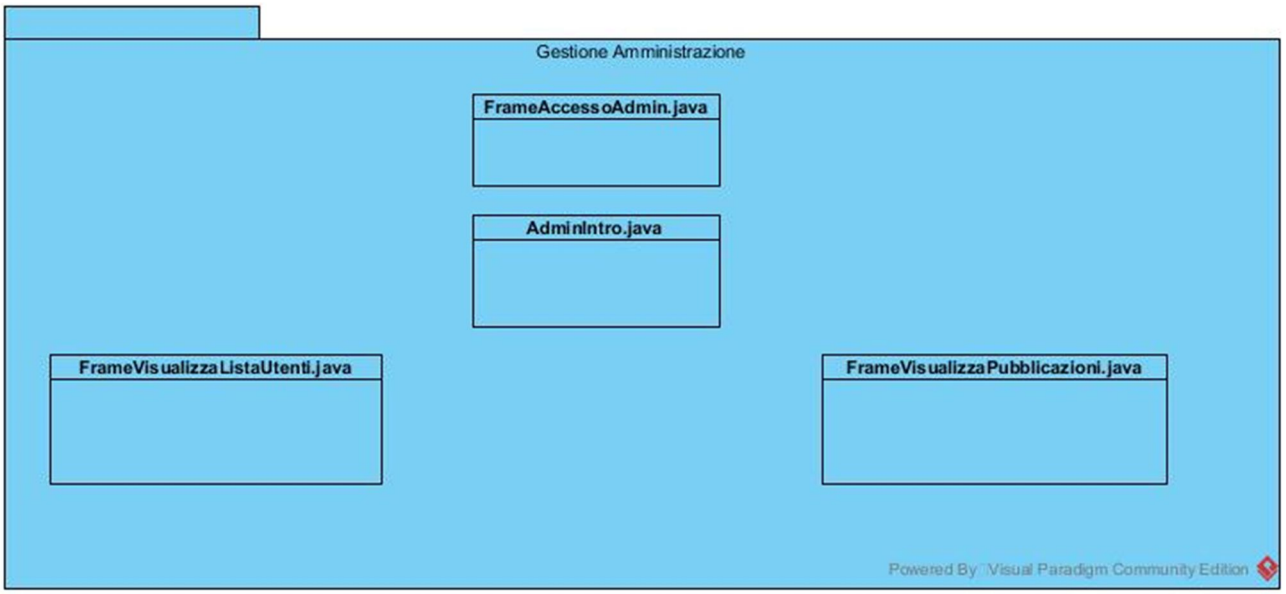
2.7 Package Interface DAO



2.8 Package POJO



Gestione Amministrazione Piattaforma:



3. Interfacce delle classi

3.1 Classi View

Classe	Descrizione
FirstAccess.xml	Visualizza la pagina dedicata alla prima volta in cui l'utente accede alla piattaforma
SceltaProfiloRegistrazione.xml	Visualizza la pagina dedicata alla scelta della modalità di registrazione
RegistrazioneArtista.xml	Visualizza la pagina relativa al form della registrazione di un artista
RegistrazioneBand.xml	Visualizza la pagina relativa al form della registrazione di una band
RegistrazionePromotore.xml	Visualizza la pagina relativa al form della registrazione di un promotore
Login.xml	Visualizza la pagina che permette di effettuare il login
Home.xml	Visualizza la pagina relativa alla HomePage della piattaforma
RecuperaPassword.xml	Visualizza la pagina dedicata al recupero della password
VisualizzaProfiloArtista.xml	Visualizza la pagina che mostra i dati del profilo di un artista
VisualizzaProfiloaBand.xml	Visualizza la pagina che mostra i dati del profilo di una band
VisualizzaProfiloPromotore.xml	Visualizza la pagina che mostra i dati del profilo di un promotore
ProfiloArtistaModificabile.xml	Visualizza la pagina che permette di modificare il profilo di un artista
ProfiloBandModificabile.xml	Visualizza la pagina che permette di modificare il profilo di una band
ProfiloPromotoreModificabile.xml	Visualizza la pagina che permette di modificare il profilo di un promotore
ListaEventiArtista.xml	Visualizza la pagina contenente l'elenco degli eventi a cui l'artista ha partecipato e parteciperà
ListaEventiBand.xml	Visualizza la pagina contenente l'elenco degli eventi a cui la band ha partecipato e parteciperà
ListaEventiPromotore.xml	Visualizza la pagina contenente gli eventi organizzati dal promotore
VisualizzaEventoArtista.xml	Visualizza la pagina del singolo evento di un artista
VisualizzaEventoBand.xml	Visualizza la pagina del singolo evento di una band
VisualizzaEventoPromotore.xml	Visualizza la pagina del singolo evento di un promotore
VisualizzaEventoPromotoreModificabile.xml	Visualizza la pagina che permette di modificare i dati di un singolo evento di un promotore
ListaProposteEvento.xml	Visualizza la pagina contente l'elenco delle proposte di evento pubblicate dai promotore
ListaProposteEventoPromotore.xml	Visualizza la pagina contenente le proposte di evento di un singolo promotore
PropostaEventoPromotore.xml	Visualizza la pagina di una singola proposta evento di un promotore
PropostaEventoPromotoreModificabile.xml	Visualizza la pagina che permette di modificare la proposta di evento di un promotore

ListaBand&Artisti.xml	Visualizza la pagina contenente la lista della band e degli artisti registrati sulla piattaforma
ListaPromotori.xml	Visualizza la pagina contenente la lista dei promotori registrati sulla piattaforma
PaginaProfiloBand.xml	Visualizza la pagina di un profilo di una band selezionata
PaginaProfiloArtista.xml	Visualizza la pagina di un profilo di un artista selezionato
PaginaProfiloPromotore.xml	Visualizza la pagina di un profilo di un promotore selezionato
PaginaEvento.xml	Visualizza la pagina di un evento selezionato
PaginaPropostaEvento.xml	Visualizza la pagina di una proposta evento selezionata
FormPropostaEvento.xml	Visualizza la pagina dedicata al form per la creazione di una proposta evento
SceltaModalitàPubblicazioneEvento.xml	Visualizza la pagina contente la modalità di pubblicazione di un evento
FormEvento.xml	Visualizza la pagina dedicata al formo per la creazione di un evento

3.2 Classi Manager

Classe	**Descrizione**
ManagerRegistrazione.java	È un'interfaccia del sottosistema che gestisce la registrazione degli utenti.
ManagerAutenticazione.java	È un'interfaccia del sottosistema che gestisce l'autenticazione di ogni utente.
ManagerAccount.java	È un'interfaccia del sottosistema che gestisce gli account degli utenti.
ManagerEventi.java	E' un'interfaccia del sottosistema che gestisce gli eventi dei promotori.

3.3 Classi Presenter

Classe	**Descrizione**
PresenterRegistrazione.java	E' una classe che gestisce la logica di controllo relativa alla registrazione degli utenti.
PresenterAutenticazione.java	E' una classe che gestisce la logica di controllo relativa all'autenticazione di ogni utente.
PresenterAccount.java	E' una classe che gestisce la logica di controllo relativa agli account.
PresenterEventi.java	E' una classe che gestisce la logica di controllo relativa agli eventi pubblicati da i promotori.
MainActivityPrincipale.java	E' una classe che permette di effettuare l'accesso ad un utente registrato
MainActivityPromotore.java	E' la classe che realizza l'activity principale del promotore, che comprende tutte le funzionalità disponibili per un utente registrato in qualità di

	promotore
MainActivityBandArtisti.java	E' la classe che realizza l'activity principale delle band e degli artisti, che comprende tutte le funzionalità disponibili per entrambi i profili

3.4 Classi DAO

Classe	Descrizione
ArtistaModelDAO.java	E' una classe che permette di interagire con la tabella Artista all'interno del database
BandModelDAO.java	E' una classe che permette di interagire con la tabella Band all'interno del database
PromotoreModelDAO.java	E' una classe che permette di interagire con la tabella Promotore all'interno del database
PropostaEventoModelDAO.java	E' una classe che permette di interagire con la tabella PropostaEvento all'interno del database
EventoModelDAO.java	E' una classe che permette di interagire con la tabella Evento all'interno del database
InteressamentoModelDAO.java	E' una classe che permette di interagire con la tabella Interessamento all'interno del database
PartecipazioneModelDAO.java	E' una classe che permette di interagire con la tabella Partecipazione all'interno del database

3.5 Classi InterfaceDAO

Classe	Descrizione
ArtistaModel.java	E' un'interfaccia che contiene i metodi CRUD relativi agli artisti.
BandModel.java	E' un'interfaccia che contiene i metodi CRUD relativi alle band.
PromotoreModel.java	E' un'interfaccia che contiene i metodi CRUD relativi ai promotori.
PropostaEventoModel.java	E' un'interfaccia che contiene i metodi CRUD relativi alle proposte di evento.
EventoModel.java	E' un'interfaccia che contiene i metodi CRUD relativi agli eventi.
InteressamentoModel.java	E' un'interfaccia che contiene i metodi CRUD relativi all'interessamento di artisti e/o band a una proposta di evento.
PartecipazioneModel.java	E' un'interfaccia che contiene i metodi CRUD relativi alla partecipazione di artisti e/o band ad un evento

3.6 Classi POJO

Classe:	**Descrizione**
Utente.java	Descrive gli utenti registrati sulla piattaforma
Artista.java	Descrive il profilo di un artista registrato sulla piattaforma
Band.java	Descrive il profilo di una band registrata sulla piattaforma
Promotore.java	Descrive il profilo di un promotore registrato sulla piattaforma
ProposteEvento.java	Descrive una proposta di evento all'interno del sistema
Evento.java	Descrive un evento all'interno del sistema
Interessamento.java	Descrive l'interesse di un artista e/o una band a una proposta di evento
Partecipazione.java	Descrive la partecipazione di un artista e/o una band a un evento

Gestione Amministrazione:

- Piattaforma Jframe lato server, per la gestione della sicurezza delle pubblicazioni.

ControlAmministrazionePiattaforma.java	E' una classe che gestisce la logica di controllo relativa all'amministrazione della piattaforma.
FrameAccessoAdmin.java	Visualizza la pagina per l'accesso alla piattaforma dal lato amministrativo
FrameVisualizzaUtenti.java	Visualizza la pagina contenente la lista degli utenti registrati sulla piattaforma, in qualità di artisti, band e promotori
FrameVisualizzaPubblicazioni.java	Visualizza la pagina contenente la lista delle pubblicazioni (proposte evento e eventi) effettuate all'interno della piattaforma
Connessione.java	E' la classe che permette di instaurare una connessione Client-Server.
ClientListener.java	E' la classe che gestisce le richieste da parte di un Client, trasportando la comunicazione lato Server.

3.7 Classi Manager

3.7.1 Manager Registrazione

E' un interfaccia del sottosistema che gestisce la registrazione degli utenti.

- public Artista registrazioneArtista()
  Questo metodo permette ad un artista di registrarsi sulla piattaforma.
- public Band registrazioneBand()
  Questo metodo permette ad una band di registrarsi sulla piattaforma.
- public Promotore registrazionePromotore()
  Questo metodo permette ad un promotore di registrarsi sulla piattaforma
- public boolean verificaMail(String email)
  Questo metodo permette di verificare che la mail inserita da colui che intende registrarsi sia già presente o meno sulla piattaforma

3.7.2 Manager Autenticazione

E' un'interfaccia del sottosistema che gestisce l'autenticazione degli utenti registrati sulla piattaforma.

- public Utente login(View v)
  Questo metodo permette agli utenti registrati di effettuare il login sulla piattaforma

3.7.3 Manager Account

E' un'interfaccia del sottosistema che gestisce gli account degli utenti registrati sulla piattaforma

- public boolean eliminaUtente(Utente utente)
  Questo metodo permette ad un utente di eliminare il proprio profilo dalla piattaforma
- public Artista visualizzaProfiloArtista(View v,MenuItem id)
  Questo metodo permette ad un artista di visualizzare il proprio profilo
- public Band visualizzaProfiloBand(View v,MenuItem id)
  Questo metodo permette ad una band di visualizzare il proprio profilo
- public Promotore visualizzaProfiloPromotore(View v, MenuItem id)
  Questo metodo permette ad un promotore di visualizzare il proprio profilo
- public Artista modificaProfiloArtista(View v,int idArtista)
  Questo metodo permette ad un artista di modificare il proprio profilo
- public Band modificaProfiloBand(View v,String emailBand)
  Questo metodo permette ad una band di modificare il proprio profilo
- public Promotore modificaProfiloPromotore(View v,String emailPromotore)
  Questo metodo permette ad un promotore di modificare il proprio profilo
- public ArrayList<PropostaEvento> visualizzaMieProposte(String emailPromotore)
  Questo metodo permette ad un promotore di visualizzare le proprie proposte di evento
- public ArrayList<Evento> visualizzaMieEventi(String emailUtente)

Questo metodo permette ad un utente di visualizzare i propri eventi.

- public ArrayList<Artista> cercaArtista(String filtro)
  Questo metodo permette di cercare un artista, restituendo una lista di artisti
- public ArrayList <Band> cercaBand(String filtro)
  Questo metodo permette di cercare una band, restituendo una lista di band
- public ArrayList<Promotore> cercaPromotore()
  Questo metodo permette di cercare un promotore, restituendo una lista di promotori.

3.7.4 Manager Amministrazione Piattaforma

E' un'interfaccia del sottosistema che gestisce l'amministrazione della piattaforma.
Questa interfaccia permette di catturare ogni click dei pulsanti dal pannello dell'amministratore.

- public ArrayList<Artista> visualizzaListaArtisti()
  Questo metodo permette all'admin di visualizzare gli artisti iscritti alla piattaforma
- public ArrayList<Band> visualizzaListaBand()
  Questo metodo permette all'admin di visualizzare le band iscritte alla piattaforma
- public ArrayList<Promotore> visualizzaListaPromotori()
  Questo metodo permette all'admin di visualizzare i promotori iscritti alla piattaforma
- public boolean eliminaArtista(Artista artista)
  Questo metodo permette all'admin di eliminare un artista iscritto sulla piattaforma
- public boolean eliminaUtente(Band band)
  Questo metodo permette all'admin di eliminare una band iscritta sulla piattaforma
- public boolean eliminaUtente(Promotore promotore)
  Questo metodo permette all'admin di eliminare un promotore iscritto sulla piattaforma
- public ArrayList<PropostaEvento> visualizzaListaProposteEvento()
  Questo metodo permette all'admin di visualizzare le proposte evento pubblicate dai promotori.
- public boolean eliminaPropostaEvento(PropostaEvento propostaEvento)
  Questo metodo permette all'admin di eliminare una proposta evento pubblicata da un promotore
- public ArrayList<Evento> visualizzaListaEventi()
  Questo metodo permette all'admin di visualizzare gli eventi pubblicate dai promotori.
- public boolean eliminaEvento(Evento evento)

Questo metodo permette all'admin di eliminare un evento pubblicato da un promotore

3.7.5 Manager Eventi

E' un'interfaccia del sottosistema che gestisce gli eventi pubblicati sulla piattaforma.

- public void pubblicaEvento(View v,PropostaEvento propostaEvento)
  Questo metodo permette ad un promotore di pubblicare un evento utilizzando una proposta di evento già pubblicata
- public void pubblicaEvento(View v, Evento evento)
  Questo metodo permette ad un promotore di pubblicare un nuovo evento
- public void pubblicaPropostaEvento(View v, PropostaEvento propostaEvento)
  Questo metodo permette ad un promotore di pubblicare una proposta evento
- public PropostaEvento modificaPropostaEvento(View v,int idPropostaEvento)
  Questo metodo permette ad un promotore di modificare una proposta evento
- public Evento modificaEvento(View v,int idEvento)
  Questo metodo permette ad un promotore di modificare un evento
- public boolean eliminaPropostaEvento(View v,int idPropostaEvento)
  Questo metodo permette ad un promotore di eliminare una proposta evento.
- public boolean eliminaEvento(View v,int idEvento)
  Questo metodo permette ad un promotore di eliminare un evento
- public ArrayList<PropostaEvento> cercaPropostaEvento(String filtro)
  Questo metodo permette di ricercare una proposta evento, restituendo una lista di proposte evento
- public ArrayList<Evento> cercaEventi(String filtro)
  Questo metodo permette di ricercare un evento, restituendo una lista di eventi
- public PropostaEvento visualizzaUnaPropostaEvento(View v, int idPropostaEvento, String nomeMetodoOnClick, Promotore promotore)
  Questo metodo permette di visualizzare una singola proposta evento
- public PropostaEvento visualizzaSingoloEvento(View v, int idEvento, String nomeMetodoOnClick, Promotore promotore)
  Questo metodo permette di visualizzare un singolo evento

4.1 Introduzione

Il sistema è stato sviluppato usufruendo dell'architettura MVP, che si ispira al pattern architetturale MVC. L'architettura MVP è definita su 3 livelli: Model,View, Presenter. Il Model si interfaccia con i dati, ovvero con gli Entity Objects; la View permette l'interazione diretta con l'utente; il Presenter è un intermediario grazie al quale viene ridotto l'accoppiamento, infatti può comunicare sia con il Model che con la View. In base all'architettura scelta, si è deciso di utilizzare due design pattern: Façade per permettere la comunicazione tra lo strato Presentation Layer e Application Layer, DAO(Data Access Object) per permettere la comunicazione fra lo Storage Layer e il Database.

4.2 Façade design pattern

Il Façade pattern permette di nascondere le complessità del sistema grazie ad un'interfaccia visibile al Presentation Layer permettendo a quest'ultimo di accedere alle funzionalità del sistema. Il software sviluppato utilizza il pattern façade per permettere la comunicazione fra le View presenti all'interno del Presentation Layer e i sottosistemi relativi alla logica di controllo. E' stata quindi realizzata un'interfaccia contenente le classi "Manager" che permettono l'accesso ai vari sottosistemi.

4.3 DAO design pattern

Il Data Access Object pattern permette la separazione tra i dati e le operazioni di basso livello dai servizi di alto livello. Questo pattern comprende: Data Access Object Interface, ossia un'interfaccia che definisce le operazioni standard da applicare su oggetti model; Data Access Object concrete class, la classe che implementa la Data Access Object Interface, è la classe responsabile di lavorare con i dati memorizzati all'interno del database; Model Object, l'oggetto POJO, ovvero il bean che descrive l'entity object in considerazione. È stato utilizzato questo pattern per permetter la comunicazione fra lo Storage Layer e il DataBase, garantendo anche una maggiore coerenza con l'architettura scelta.

4. Conclusione

Qui sotto, è stato riportato lo schema completo del sistema.