**Documentazione Funzionalità “Cinema”**

Daniele Pellegrini 285240

Riccardo Fava 287516

|  |  |
| --- | --- |
| **Definizione** | **Funzionamento** |
| USER / ADMIN | |
| bool Registration(bool isAdmin, string username,  string password, string name, string surname) | * TRUE: Registrazione avvenuta con successo * FALSE:   + Username (PK) già esistente   + Errore Query   + Connessione mancante   + Eccezione |
| bool Login(bool isAdmin, string username,   string password) | * TRUE: Username e Password corretti * FALSE:   + Username e password sbagliati per il login User o Admin   + Errore Query   + Connessione mancante   + Eccezione |
| User GetUser(bool isAdmin, string username) | Ritorna l’oggetto User:   * new User() : errore esecuzione Query |
| List<User> GetUsersList() | Restituisce la lista degli Utenti registrati del cinema.   * new List<User>(){}:se non ci sono utenti registrati o in caso di errore. |
| List<User> GetSubscribersList() | Restituisce la lista degli Utenti Abbonati al cinema.   * new List<User>(){}:se non ci sono utenti abbonati o in caso di errore. |
| bool DeleteUser(string username) | Elimina l’utente dal Database Cinema, e dalle altre tabelle.   * TRUE: Utente eliminato dal Database con successo. * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| bool EditUser(string oldUsername, string newUsername, string newPassword, string newName, string newSurname) | INPUT:   * oldUsername: attuale username dell’utente di cui si vogliono modificare i dati * newAttribute: nuovi dati da sostituire a quelli dell’Utente che si vuole modificare.   OUTPUT:   * TRUE: Utente modificato con successo * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| void CheckConnection() | Funzione utilizzata dal Server per verificare la connessione con il Database Manager. Viene semplicemente rilanciata nel Client per informare l’Utente qualora ci fossero problemi di comunicazione tra Client-Server o tra Server-Database. |
| bool CheckStringFK(string value, string valueType) | Controlla se la chiave primaria in formato varchar inserito per una tabella esiste.  INPUT:   * valueType: Nome della tabella di cui si vuole controllare la chiave primaria * value: valore corrispondente alla chiave primaria della tabella   OUTPUT:   * TRUE: la chiave primaria esiste * FALSE: la chiave primaria non esiste |
| FILM | |
| bool AddFilm(string title, int year, string direction,  int duration, DateTime releaseDate, string genre) | * TRUE: Film aggiunto con successo * FALSE:   + releaseDate in un formato errato   + Errore Query   + Connessione mancante   + Eccezione |
| bool DeleteFilm(int filmCode) | * TRUE: Film eliminato dal Database con successo. * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| Film GetFilm(int filmCode) | Ritorna l’oggetto Film:   * new Film() : errore esecuzione Query |
| List<Film> GetFilmList() | Restituisce la lista dei Film del cinema.   * new List<Film>(){}:se non ci sono film all’interno del database o in caso di errore |
| EVENTO | |
| bool AddEvent(string usernameAdmin, DateTime dateTime,  int filmCode, int hallCode, decimal price) | * TRUE: Evento aggiunto con successo * FALSE:   + dateTime in un formato errato   + price in un formato errato (prezzo dell’ordine delle centinaia)   + Errore Query   + Connessione mancante   + Eccezione |
| bool DeleteEvent(int eventCode) | * TRUE: Evento eliminato dal Database con successo. * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| Event GetEvent(int eventCode) | Ritorna l’oggetto Evento:   * new Event() : errore esecuzione Query |
| List<Event> GetEventsList() | Restituisce la lista degli Eventi del cinema.   * new List<Event>(){}:se non ci sono eventi all’interno del database o in caso di errore |
| List<Show> GetShowsList() | La struct Show rappresenta una vista particolare per l’Utente che desidera comprare un biglietto per uno spettacolo.  Restituisce la lista degli Spettacoli del database.   * new List<Show>(){}:se non ci sono spettacoli all’interno del database o in caso di errore |
| HALL | |
| List<Hall> GetHallsList() | Restituisce la lista delle Sale del cinema.   * new List<Hall>(){}:se non ci sono sale all’interno del database o in caso di errore. |
| string DrawHall(int eventCode) | Restituisce una stringa contenente un ascii art contenente una rappresentazione della Sala in cui si svolge lo spettacolo. |
| PLACE | |
| List<Place> GetAvailablePlacesList(int eventCode) | Restituisce la lista dei Posti disponibili della sala del cinema in cui si svolge uno spettacolo.   * new List<Place>(){}:se non ci sono posti all’interno del database o in caso di errore. |
| bool CheckPlace(int eventCode, int placeNumber) | Controlla che il posto scelto dall’utente sia valido.  TRUE:   * Il posto è valido:   + esiste nella sala   + non è già stato comprato   FALSE:   * Il posto non è valido:   + non esiste all’interno della sala   + è già stato comprato * Errore Query * Connessione mancante * Eccezione |
| bool CheckIntFK(string value, string valueType) | Controlla se la chiave primaria in formato int inserita per una tabella esiste.  INPUT:   * valueType: Nome della tabella di cui si vuole controllare la chiave primaria * value: valore corrispondente alla chiave primaria della tabella   OUTPUT:   * TRUE: la chiave primaria esiste * FALSE: la chiave primaria non esiste |
| PRENOTAZIONE | |
| bool AddPrenotation(DateTime dateTime, string  usernameUser, int eventCode, int placeNumber) | * TRUE: Prenotazione aggiunta con successo * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| bool DeletePrenotation(int prenotationCode) | * TRUE: Prenotazione eliminato dal Database con successo. * FALSE:   + Errore Query   + Connessione mancante   + Eccezione |
| bool AddSubscription(string username) | Sottoscrive un abbonamento al CInema per un Utente.  INPUT:   * username dell’utente che si vuole abbonare   OUTPUT:   * TRUE: Abbonamento registrato con successo * FALSE:   + username non esistente   + Errore Query   + Connessione mancante   + Eccezione |
| bool DeleteSubscription(string username) | Cancella la sottoscrizione di un abbonamento attivo al CInema per un Utente.  INPUT:   * username dell’utente che si vuole abbonare   OUTPUT:   * TRUE: Abbonamento registrato con successo * FALSE:   + username non esistente   + Errore Query   + Connessione mancante   + Eccezione |
| bool GetSubscription(string username); | Controlla se l’utente è presente nella tabella “Abbonamento”, e quindi se l’utente è abbonato.  INPUT:   * username dell’utente che si vuole abbonare   OUTPUT:   * TRUE: Utente abbonato * FALSE:   + Utente non abbonato   + Errore Query   + Connessione mancante   + Eccezione |
| List<Prenotation> GetPrenotationsList() | Restituisce la lista dei Prenotazioni del Cinema.   * new List<Prenotazione>(){}:se non ci sono prenotazioni all’interno del database o in caso di errore. |
| List<Ticket> GetTicketsList(string username) | La struct Ticket rappresenta un oggetto contenente diverse informazioni su Film, Evento, Prenotazione e Posto in riferimento alla Prenotazione dell’Utente. Rappresenta dunque il suo biglietto, che può consultare.  Restituisce la lista dei biglietti di un Utente del database.   * new List<Ticket>(){}:se non ci sono biglietti acquistati dall’utente o in caso di errore |