Events Handler

*AUTORI:*

*Francesco Digirolamo*

*(e-mail: f.digirolamo8@studenti.uniba.it)*

*&*

*Vincenzo Biallo*

*(e-mail: v.biallo1@studenti.uniba.it)*

***Laboratorio di informatica***

***A.A 2020/21***

**CASO DI STUDIO: TRACCIA 4**

**d**

**Sommario**

[ANALISI 2](#_Toc75015816)

[Descrizione del software 2](#_Toc75015817)

[Requisiti funzionali 4](#_Toc75015818)

[Casi d’uso 6](#_Toc75015819)

[PROGETTAZIONE 18](#_Toc75015820)

[Progettazione dei tipi di dati e strutture 19](#_Toc75015821)

[Progettazione delle librerie e funzioni 21](#_Toc75015822)

[Dipendenza tra funzioni 23](#_Toc75015823)

[Pseudocodice 24](#_Toc75015824)

[Gestione Generale 24](#_Toc75015825)

[Gestione Utenti 25](#_Toc75015826)

[Gestione Locali 26](#_Toc75015827)

[Gestione Eventi 27](#_Toc75015828)

[Gestione Prenotazioni 30](#_Toc75015829)

[TESTING 31](#_Toc75015830)

# **ANALISI**

## Descrizione del software

*Si vuole realizzare un software per la vendita di biglietti e gestione di eventi musicali. Il software agisce a livello locale e su file in formato “Human-readable” operando in scrittura e lettura, creando, modificando e cancellando dati.*

*I dati sono memorizzati su molteplici file CSV (da allegare in maniera obbligatoria al programma che verranno caricati all’avvio del software e saranno sottoposti a una diagnostica di integrità e affidabilità).*

*L’esecuzione del software continuerà solo se superata la diagnostica, altrimenti verrà visualizzato un warning e verranno avviate delle operazioni di ripristino dati.*

*Nello specifico, i dati trattati riguardano gli utenti del sistema, nonché i possibili spettatori, gli eventi musicali, i locali in cui questi ultimi si terranno e le rispettive prenotazioni.*

*Gli* ***UTENTI*** *nel software sono caratterizzati da:*

* *Nickname (univoco)*
* *E-mail*
* *Cognome*
* *Nome*
* *Genere (M/F/ Non Specificato)*
* *Data di nascita (opzionale)*
* *Numero cellulare*
* *Genere musicale preferito (scelto tra: BLUES, CLASSICA, HEAVY METAL, HIP-POP, JAZZ, POP, PUNK, ROCK, ALTRO).*

*Gli* ***EVENTI*** *nel software sono caratterizzati da:*

* *Codice univoco*
* *Nome*
* *Data*
* *Orario di inizio evento*
* *Orario di fine evento*
* *Ospite della serata*
* *Codice locale (si è ritenuto opportuno introdurre questa variabile al posto del nome del locale e del numero posti, in quanto rappresenta una chiave esterna di un record in un altro file CSV, successivamente descritto).*
* *Costo biglietto della prima fila*
* *Costo biglietto dell’ultima fila*

*Le* ***PRENOTAZIONI*** *nel software sono caratterizzate da:*

* *Numero univoco*
* *Codice evento (prenotato)*
* *Nickname dell’utente (che ha effettuato la prenotazione)*
* *Fila prenotata*
* *Numero poltrona prenotata*
* *Importo*

*I* ***LOCALI*** *nel software sono caratterizzati da:*

* *Codice univoco*
* *Nome del locale*
* *Numero file*
* *Numero posti*

*Fondamentalmente il sistema prevede le seguenti funzioni:*

* *Registrazione nuovo utente*
* *Modifica dati di un utente già esistente*
* *Aggiunta di un nuovo evento*
* *Ricerca di un evento in base alla data o al nome dell’ospite*
* *Modifica o annullamento di un evento già organizzato*
* *Visualizzazione della lista degli spettatori prenotati a un evento*
* *Visualizzazione della lista degli eventi a cui un utente ha partecipato, in ordine di data o di costo*
* *Vendita di un biglietto ad un utente*
* *Verifica della disponibilità di una poltrona di un evento*
* *Visualizzazione dell’incasso complessivo di un evento*

## **Requisiti funzionali**

|  |  |  |
| --- | --- | --- |
| **CODICE** | **NOME** | **DESCRIZIONE** |
| RM00 | *Visualizzazione del menù di scelta* | *Il programma visualizza a schermo il menù con tutte le possibili opzioni di navigazione e attende un input corretto da parte dell'utente.* |
| RU00 | *Registrazione nuovo utente* | *Il programma visualizza a schermo il modulo di registrazione di un utente, attendendo un corretto input per ogni campo richiesto. Ogni utente è identificato in maniera univoca dal suo nickname.* |
| RU01 | *Modifica dati di un utente già esistente* | *Il programma visualizza con il medesimo modulo di registrazione, l'interfaccia utente, mostrando inoltre l'attuale valore che possiede ogni campo. E' possibile modificarne il contenuto (tranne del nickname) oppure CTRL + Z se si vuol lasciare inalterato il campo.* |
| RE00 | *Aggiunta di un nuovo evento* | *Il programma visualizza a schermo il modulo di compilazione di un evento, attendendo un corretto input per ogni campo richiesto. In particolare, riguardo l'inserimento del locale in cui si terrà l'evento, visualizza la lista dei locali disponibili attendendo una scelta tra quelli proposti.* |
| RE01 | *Ricerca di un evento* | *Il programma ricerca un evento in base a un parametro indicato dall'utente, visualizzandone i dati.* |
| RE02 | *Modifica o annullamento di un evento* | *All'interno della gestione di un evento il programma propone le scelte di annullamento o modifica di un evento. Riguardo alla seconda, il programma visualizza il modulo di compilazione dell'evento mostrando, inoltre, l'attuale valore che possiede ogni campo. Per modificarne il contenuto l'utente deve inserire un nuovo valore, altrimenti CTRL + Z per lasciarlo inalterato.* |
| RE03 | *Visualizzazione spettatori prenotati a un evento* | *Il programma aspetta in input il parametro di ricerca dell'evento e successivamente visualizza la lista degli spettatori prenotati per lo stesso.* |
| RU02 | *Visualizzazione degli eventi a cui un utente ha partecipato* | *Il programma aspetta in input il nickname dell'utente interessato, successivamente visualizza una lista di tutti gli eventi a cui quest'ultimo ha partecipato, in ordine di data o costo.* |
| RU03 | *Vendita di un biglietto ad un utente* | *Il programma visualizza il modulo di vendita del biglietto e aspetta l'input dei vari campi, se tutto è corretto chiede la conferma di vendita del biglietto.* |
| RE05 | *Verifica della disponibilità di una poltrona di un evento* | *Il programma effettua una verifica sulla disponibilità di una poltrona indicata dall'utente. Questa funzionalità è correlata alla funzionalità RU03, infatti solo se la poltrona indicata è libera, il biglietto potrà essere venduto.* |
| RE06 | *Visualizzazione dell'incasso complessivo di un evento* | *Il programma visualizza l'incasso complessivo di un evento, dato dalla vendita dei biglietti.* |
| RU04 | *Ricerca di un utente* | *Il programma ricerca un utente ne file CSV apposito in base al nickname inserito in input.* |
| RL00 | *Aggiungi locale* | *Il programma visualizza il modulo di registrazione del locale, attendendo un corretto input per ogni campo.* |
| RL01 | *Modifica locale* | *Il programma visualizza il medesimo modulo di registrazione del locale, visualizzando il valore attuale per ogni campo. Rende possibile modificarne il contenuto inserendo un altro valore corretto oppure lasciare l’attuale inserendo CTRL + Z.* |
| RL02 | *Cancella locale* | *Il programma chiede il nome del locale interessato, ne visualizza le informazioni e chiede la conferma di eliminazione.* |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RM00*** | *Visualizzazione del menu di scelta.* | *Il programma visualizza a schermo il menù con tutte le possibili opzioni di navigazione e attende un input corretto da parte dell'utente.* |
| ***PRE-CONDIZIONI*** | *Diagnostica superata con successo.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il software visualizza con successo la lista delle scelte e dà la possibilità all’utente di scegliere una delle opzioni disponibili.* | *La diagnostica ha riscontrato degli errori e propone all’utente una serie di scelte per risolverli*. *Una volta risolti verrà riproposto il caso d’uso.* |
| ***EVENTO INNESCANTE*** | *Avvio del programma* |  |
| ***SCENARIO DI BASE*** | *Viene mostrato il menù a scelta e si aspetta un input da parte dell’utente* |  |
| ***SCENARIO ALTERNATIVO*** | *L'input inserito dall'utente non è corretto: non rientra tra quelli previsti dal menu. Il programma richiede il suo inserimento finché non è corretto.* |  |

## **Casi d’uso**

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RU00*** | *Registrazione di un nuovo utente.* | *Il programma visualizza a schermo il modulo di registrazione di un utente, attendendo un corretto input per ogni campo richiesto. Ogni utente è identificato in maniera univoca dal suo nickname.* |
| ***PRE-CONDIZIONI*** | * *Per ogni campo richiesto, l'input deve essere corretto, deve rispettare le sue regole;* * *il nickname scelto non deve essere stato già utilizzato.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il software inserisce correttamente il nuovo utente nel file CSV apposito.* | *Il sistema non inserisce l'utente nel file CSV apposito e richiede se riprovare o annullare l’inserimento.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Registra nuovo utente".* |  |
| ***SCENARIO DI BASE*** | *L’utente viene registrato correttamente ed è abilitato per la prenotazione di eventi.* |  |
| ***SCENARIO ALTERNATIVO*** | *Almeno un input non rispetta le regole di almeno un campo. L'azione verrà ripetuta finché l'input non risulta corretto oppure si annulla l’inserimento.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RU01*** | *Modifica dati di un utente già esistente.* | *Il programma visualizza con il medesimo modulo di registrazione, l'interfaccia utente, mostrando inoltre l'attuale valore che possiede ogni campo. È possibile modificarne il contenuto (tranne del nickname) oppure inserire il carattere di fine stringa se si vuol lasciare inalterato il campo.* |
| ***PRE-CONDIZIONI*** | * *"CODICE RICERCA UTENTE" andato a buon fine;* * *corretto caricamento in memoria della stringa, relativa al nickname scelto, del file CSV apposito;* * *utente già registrato nel file CSV apposito.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il programma procede con la modifica dei dati dell'utente e viene visualizzato il rispettivo messaggio di successo.* | *Il programma mostra un errore in quanto il caricamento della stringa, relativa al nickname scelto, del file CSV apposito non è andato a buon fine oppure il nickname inserito non è stato trovato.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Modifica utente".* |  |
| ***SCENARIO DI BASE*** | *I dati dell'utente vengono modificati e salvati localmente.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il nickname inserito restituisce "UTENTE NON TROVATO". L'operazione si conclude con un messaggio di avviso e il programma torna nell'interfaccia del menu principale.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE00*** | *Aggiunta di un nuovo evento* | *Il   programma visualizza a schermo il modulo di compilazione di un evento, attendendo un corretto input per ogni campo richiesto. In particolare, riguardo l'inserimento del locale in cui si terrà l'evento, visualizza la lista dei locali disponibili attendendo una scelta tra quelli proposti.* |
| ***PRE-CONDIZIONI*** | * *Per ogni campo richiesto, l'input deve essere corretto, deve rispettare le sue regole;* * *Il codice del locale richiesto per la registrazione deve esistere ed essere trovato nel file CSV dei locali.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il software inserisce correttamente il nuovo evento nel file CSV apposito.* | *Il sistema non inserisce l'evento nel file CSV apposito, in quanto ci sono stati errori logici oppure non è possibili attribuirli un locale.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Aggiungi nuovo evento".* |  |
| ***SCENARIO DI BASE*** | *L’'evento viene registrato correttamente.* |  |
| ***SCENARIO ALTERNATIVO*** | *All’inserimento di un input sbagliato verrà chiesto di reinserirlo o di annullare la registrazione. Qualora non ci fossero locali disponibili la registrazione di un nuovo evento risulta bloccata.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE01*** | *Ricerca di un evento* | *Il   programma ricerca un evento in base a un parametro indicato dall'utente, visualizzandone i dati.* |
| ***PRE-CONDIZIONI*** | *Il parametro di ricerca inserito deve essere valido.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il programma visualizza tutti gli eventi che rispettano il parametro di ricerca e propone all'utente la scelta di un unico evento tra quelli trovati.*  *[!] Se l'evento è unico quest'ultima fase verrà saltata automaticamente.* | *Il programma visualizza un warning di fallimento nella ricerca dell'evento.* |
| ***EVENTO INNESCANTE*** | * *Selezione della voce del menù: "Ricerca un evento";* * *chiamata da altre funzioni.* |  |
| ***SCENARIO DI BASE*** | *Il software visualizza le informazioni dell'evento con la possibilità di modificarlo o annullarlo oppure ritorna l’istanza.* |  |
| ***SCENARIO ALTERNATIVO*** | *Se sono stati trovati più eventi il programma chiede all'utente di selezionarne uno per poi procedere con lo scenario di base. Oppure se non è stato trovato nemmeno un evento viene visualizzato un warning.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RU04*** | *Ricerca di un utente* | *Il programma ricerca un utente ne file CSV apposito in base al nickname inserito in input.* |
| ***PRE-CONDIZIONI*** | *Il nickname deve essere valido.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | La funzione restituisce l’indirizzo della locazione di memoria in cui si trova l’utente ricercato. | *La funzione restituisce NULL se l’array di utenti è vuoto oppure l’utente non è stato trovato* |
| ***EVENTO INNESCANTE*** | * *Chiamata da altre funzioni;* * *Selezione dell’opzione “Ricerca utente” dal menù di navigazione principale* |  |
| ***SCENARIO DI BASE*** | L’utente viene trovato e possono essere svolte le operazioni riguardanti lo stesso, come la visualizzazione dei suoi dati |  |
| ***SCENARIO ALTERNATIVO*** | Viene mostrato un messaggio di avviso. |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE02*** | *Modifica o annullamento di un evento* | *All'interno   della gestione di un evento il programma propone le scelte di annullamento o   modifica di un evento. Riguardo alla seconda, il programma visualizza il   modulo di compilazione dell'evento mostrando, inoltre, l'attuale valore che   possiede ogni campo. Per modificarne il contenuto l'utente deve inserire un   nuovo valore, altrimenti CTRL + Z per lasciarlo inalterato.* |
| ***PRE-CONDIZIONI*** | * *RE01 deve aver ottenuto un’istanza;* * *deve essere scelta 1 delle 2 opzioni;* * *l'evento deve essere un evento futuro.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *L'evento viene modificato o annullato e viene visualizzato il rispettivo messaggio di successo.* | *Nel caso di modifica, uno dei valori inseriti non rispetta le regole di almeno un campo e l'evento non viene aggiornato. Oppure si è tentato di agire su un evento già concluso.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Gestisci evento".* |  |
| ***SCENARIO DI BASE*** | *I dati dell'evento vengono modificati e salvati localmente, oppure l'evento viene annullato, rimborsando gli utenti che hanno acquistato il relativo biglietto.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il risultato non è quello atteso, viene visualizzato un messaggio di errore.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE03*** | *Visualizzazione spettatori prenotati a un evento.* | *Il programma aspetta in input il parametro di ricerca dell'evento e successivamente visualizza la lista degli spettatori prenotati per lo stesso.* |
| ***PRE-CONDIZIONI*** | *RE01 ha trovato un’istanza* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il software visualizza a schermo la lista di tutti i nickname degli spettatori prenotati per l'evento cercato.* | *Il programma visualizza un warning di fallimento nella ricerca dell'evento.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Visualizza spettatori di un evento".* |  |
| ***SCENARIO DI BASE*** | *Il programma funziona correttamente e visualizza la lista degli spettatori dell'evento cercato.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il risultato non è quello atteso, viene visualizzato un messaggio di errore.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RU02*** | *Visualizzazione degli eventi a cui un utente ha partecipato.* | *Il programma aspetta in input il nickname dell'utente interessato, successivamente visualizza una lista di tutti gli eventi a cui quest'ultimo ha partecipato, in ordine di data o costo.* |
| ***PRE-CONDIZIONI*** | * *RU04 andato a buon fine;* * *"CODICE RICERCA BIGLIETTI" andato a buon fine.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il programma visualizza la lista degli eventi (senza ridondanze) ordinata per data o costo, oppure visualizza un warning se l'utente non ha mai acquistato un biglietto di un evento, in passato.* | *l programma mostra un errore in quanto il nickname inserito non è stato trovato.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Visualizza eventi a cui un utente ha partecipato".* |  |
| ***SCENARIO DI BASE*** | *Il risultato è quello atteso, il programma visualizza la lista degli eventi, a cui l'utente ha partecipato, ordinata per costo o data.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il programma visualizza un messaggio di warning, in quanto l'utente non ha mai acquistato un biglietto, per un evento, in passato.* |  |

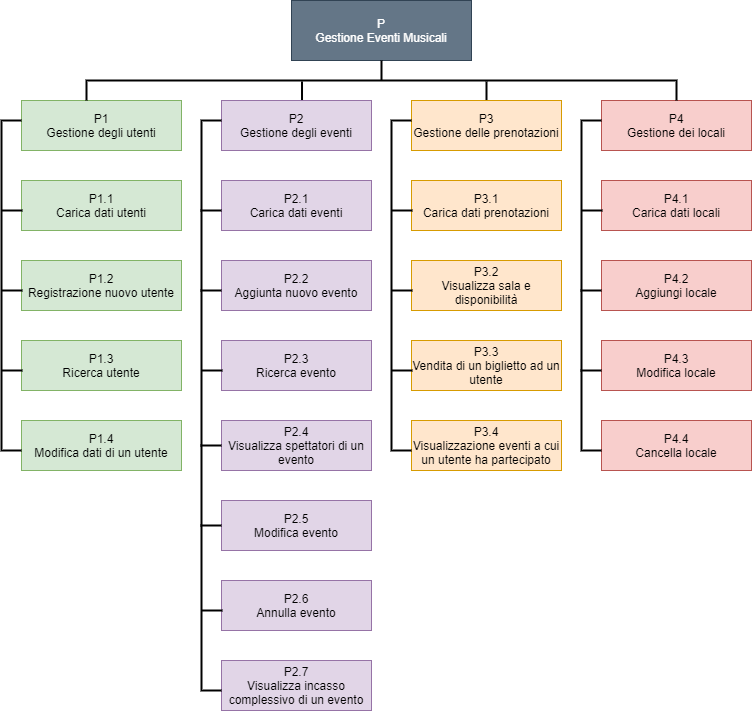
|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RU03*** | *Vendita di un biglietto ad un utente* | *Il   programma visualizza il modulo di vendita del biglietto e aspetta l'input dei vari campi, se tutto è corretto chiede la conferma di vendita del biglietto.* |
| ***PRE-CONDIZIONI*** | * RU04 andato a buon fine; * RE01 andato a buon fine; * l'evento deve essere futuro; * RE05 andato a buon fine. |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il sistema riconosce correttamente l'utente, l'evento e ha verificato che la poltrona scelta è disponibile; ne consegue che avviene con successo la vendita del biglietto.* | *Il programma riscontra un problema nel riconoscimento dell'utente o dell'evento, visualizza un messaggio di errore e riporta al menù principale.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Vendi biglietto".* |  |
| ***SCENARIO DI BASE*** | *Il risultato è quello atteso, il programma riesce a vendere correttamente il biglietto all'utente, prenotando la poltrona interessata.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il risultato non è quello atteso: il programma avvisa che la poltrona scelta è già stata prenotata oppure avvisa se il locale è totalmente prenotato; per entrambi i casi fornisce la possibilità di ritornare al menù principale.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE05*** | *Verifica della disponibilità di poltrone di un evento* | *Il   programma mostra una simulazione della sala con le poltrone ordinate per file e numero, evidenziando in verde le poltrone libere e in rosso le poltrone già prenotate. Questa funzionalità può essere correlata al caso d'uso RU03.* |
| ***PRE-CONDIZIONI*** | * *RE01 andato a buon fine;* * *l'evento deve essere futuro.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il programma visualizza una simulazione della sala con le poltrone ordinate per file e numero, evidenziando in verde le poltrone libere e in rosso le poltrone già prenotate.* | *Il programma visualizza un messaggio di errore e riporta l'utente alla schermata di menù principale.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Vendi biglietto" oppure “Visualizza disponibilità”.* |  |
| ***SCENARIO DI BASE*** | *Il programma visualizza correttamente la simulazione della sala e permette di proseguire con la vendita del biglietto.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il programma avvisa che l'evento è un evento passato oppure è sold-out.* |  |

|  |  |  |
| --- | --- | --- |
|  | ***DESCRIZIONE*** |  |
| ***CODICE: RE06*** | *Visualizzazione dell'incasso complessivo di un evento* | *Il programma visualizza l'incasso complessivo di un evento, dato dalla vendita dei biglietti.* |
| ***PRE-CONDIZIONI*** | *RE01 andato a buon fine.* |  |
| ***POST-CONDIZIONI*** | ***SUCCESSO*** | ***FALLIMENTO*** |
|  | *Il programma visualizza l'incasso complessivo dell'evento, indicando il numero complessivo delle prenotazioni.* | *Il programma visualizza un messaggio di errore, in quanto non è stato possibile trovare l'evento.* |
| ***EVENTO INNESCANTE*** | *Selezione della voce del menù: "Visualizza incasso evento".* |  |
| ***SCENARIO DI BASE*** | *Il risultato è quello atteso e vengono visualizzati l'incasso complessivo e il numero delle prenotazioni.* |  |
| ***SCENARIO ALTERNATIVO*** | *Il risultato non è quello atteso, non è stata ancora prenotata nessuna poltrona oppure l'evento inserito non è quello atteso.* |  |

# **PROGETTAZIONE**

*Il seguente schema rappresenta le 4 aree principali di gestione del software.*

**

## **Progettazione dei tipi di dati e strutture**

Sia per la struct utente, che per la struct locale, evento e prenotazione, è stata creata un ulteriore struct di riferimento col nome al plurale (esempio: utenti, eventi, prenotazioni, locali) che contiene un puntatore ad un array dinamico con campo di struct al singolare, un intero che contiene la dimensione dell’array e per le struct che necessitavano di un index AUTO\_INCREMENT un puntatore all’intero di riferimento dell’indice.

|  |  |  |  |
| --- | --- | --- | --- |
| **NOME** | ***TIPOLOGIA*** | **DESCRIZIONE** | **NOME ATTRIBUTO/TIPO/CAMPO** |
| ***UTENTE*** | *STRUCT* | *Tipo di dato definito utilizzato per descrivere gli attributi degli utenti del sistema nonché degli settatori* | *Nickname: char nickname[16]* |
| *E-mail: char email[64]* |
| *Cognome: char cognome[20]* |
| *Nome: char nome[20]* |
| *Genere: int genere* |
| *Data di nascita: data dataNascita* |
| *N. Telefono: char telefono* |
| *Genere musicale: int genereMusicale* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME ATTRIBUTO/TIPO/CAMPO*** |
| ***EVENTO*** | *STRUCT* | *Tipo di dato definito utilizzato per descrivere gli attributi degli eventi* | *Codice univoco: int codice* |
| *Data: data* |
| *Orario inizio: orario oraInizio* |
| *Orario fine: orario oraFine* |
| *Nome ospite: char ospite[50]* |
| *Codice locale: int locale* |
| *Costo prima fila: float costo\_prima\_fila* |
| *Costo ultima fila: float costo\_ultima\_fila* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME ATTRIBUTO/TIPO/CAMPO*** |
| ***PRENOTAZIONE*** | *STRUCT* | *Tipo di dato definito utilizzato per descrivere gli attributi delle prenotazioni* | *Numero prenotazione: int numero* |
| *Codice evento: int evento* |
|
| *Nickname utente: char nickname[16]* |
| *Fila: int fila* |
| *Numero poltrona: int n\_poltrona* |
| *Costo: float importo* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME ATTRIBUTO/TIPO/CAMPO*** |
| ***LOCALE*** | *STRUCT* | *Tipo di dato definito utilizzato per descrivere gli attributi dei locali* | *Codice locale: int codice* |
| *Nome locale: char nome[20]* |
| *Numero file: int file* |
| *Numero poltrone per fila: int posti* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME COSTANTE/VALORE*** |
| ***genere\_musicale*** | *ENUM* | *Tipo di dato definito per descrivere i generi musicali* | *BLUES=0* |
| *CLASSICA=1* |
| *HEAVY\_METAL=2* |
| *HIP-POP=3* |
| *JAZZ=4* |
| *POP=5* |
| *PUNK=6* |
| *ROCK=7* |
| *ALTRO=8* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME COSTANTE/VALORE*** |
| ***genere*** | *ENUM* | *Tipo di dato definito per descrivere il genere dell'utente* | *M=0* |
| *F=1* |
| *ALTRO=2* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***FORMATTAZIONE*** |
| ***eventi.csv*** | *File* | *File contenente i dati degli eventi* | *CODICE, DATA, ORARIO\_INIZIO, ORARIO\_FINE, CODICE\_LOCALE, PRIMA\_FILA, ULTIMA\_FILA* |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***FORMATTAZIONE*** |
| ***locali.csv*** | *File* | *File contenente i dati dei locali* | *CODICE, NOME, N\_FILE, N\_POSTI* |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***FORMATTAZIONE*** |
| ***prenotazioni.csv*** | *File* | *File contenente i dati delle prenotazioni* | *NUMERO, CODICE\_EVENTO, NICKNAME, FILA, POLTRONA, IMPORTO* |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***FORMATTAZIONE*** |
| ***utenti.csv*** | *File* | *File contenente i dati degli utenti* | *NICKNAME, E-MAIL, COGNOME, NOME, GENERE, DATA\_NASCITA, N\_CELLULARE, GENERE\_PREFERITO* |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***FORMATTAZIONE*** |
| ***indexs.dat*** | *File* | *File contenente gli indici delle principali entità* | *Codice AUTO\_INCREMENT eventi|locali|prenotazioni* |
|
|

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME ATTRIBUTO/TIPO/CAMPO*** |
| ***colori*** | *ENUM* | *Tipo di dato definito per descrivere i colori di visualizzazione di messaggi a schermo* | *ROSSO* |
| *GIALLO* |
| *VERDE* |
| *CELESTE* |
| *VIOLA* |
| *DEFAULT* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***NOME*** | ***TIPOLOGIA*** | ***DESCRIZIONE*** | ***NOME ATTRIBUTO/TIPO/CAMPO*** |
| ***TypeMerge*** | *ENUM* | *Tipo di dato definito per descrivere che struttura di dati si deve ordinare* | *\_utenti* |
| *\_eventi* |
| *\_locali* |
| *\_prenotazioni* |

## **Progettazione delle librerie e funzioni**

*Sono state progettate le seguenti librerie:*

* *utenti*
* *eventi*
* *prenotazioni*
* *locali*
* *utils*

*I metodi principali nella libreria utenti sono:*

* *int caricaUtenti()*
* *int salvaUtenti()*
* *int aggiungiUtente()*
* *int modificaUtente()*
* *void visualizzaUtente()*
* *utente\* ricercaUtente()*
* *int esisteUtente()*

*I metodi principali nella libreria eventi sono:*

* *int caricaEventi();*
* *int salvaEventi();*
* *int aggiungiEvento()*
* *evento\* cercaEventoByCodice()*
* *evento\* cercaEventoByData()*
* *evento\* cercaEventoByString()*
* *int modificaEvento()*
* *int cancellaEvento()*
* *void mostraSpettatori()*
* *float mostraIncasso()*

*I metodi principali nella libreria prenotazioni sono:*

* *void visualizzaDisponibilita(evento\*)*
* *void vendiBiglietto()*
* *void mostraEventiUtente(utente\*)*

*I metodi principali nella libreria locali sono:*

* *int caricaLocali()*
* *int salvaLocali()*
* *int aggiungiLocale()*
* *int modificaLocale()*
* *vois visualizzaLocale()*
* *int eliminaLocale()*
* *locale\* ricercaLocale()*

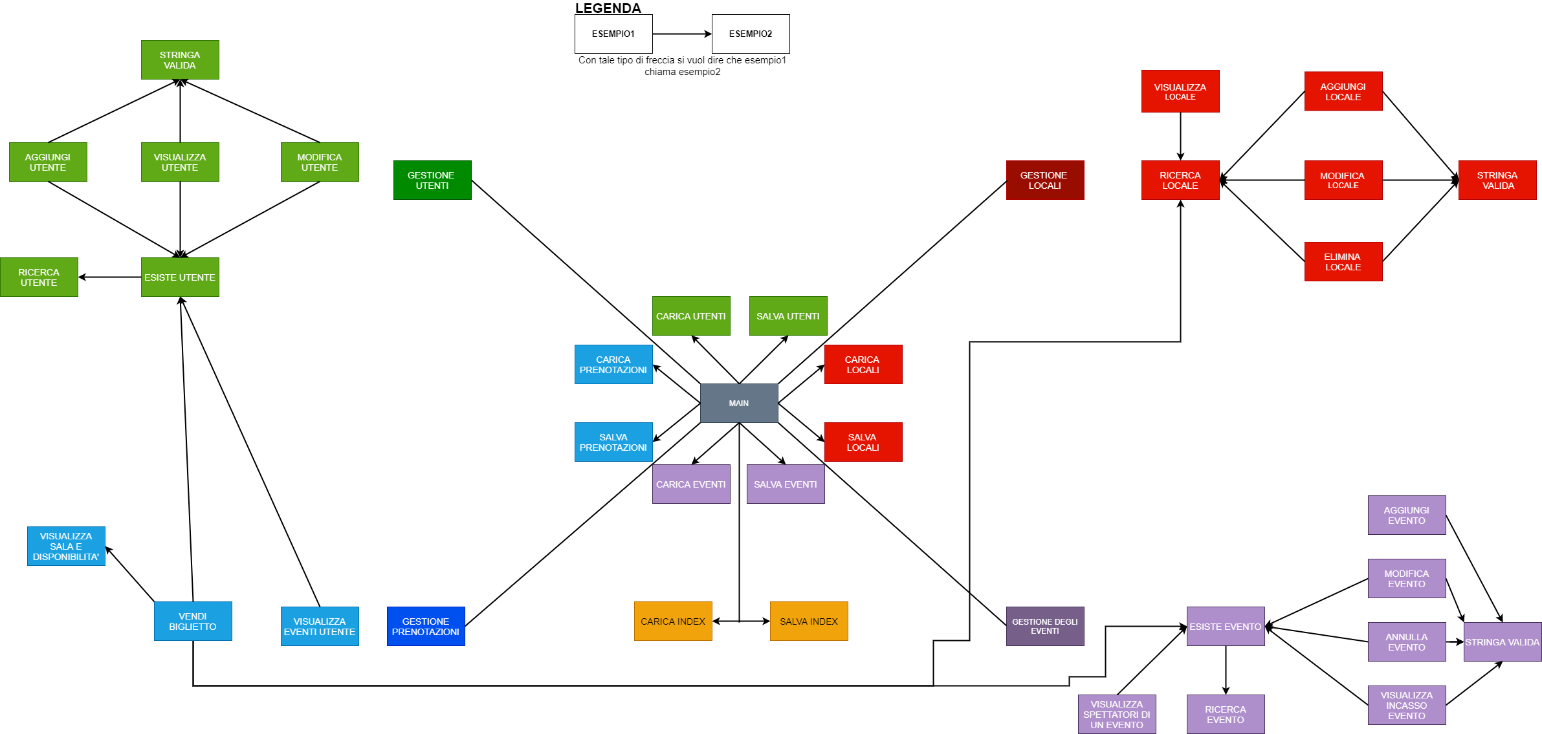
*I metodi principali nella libreria utils sono:*

* *void caricaIndex()*
* *void salvaIndex()*
* *contaLinee()*
* *stringaValida()*
* *myParser()*
* *myLower()*
* *setColor()*
* *mergeSort()*

*Le librerie sono state ideate utilizzando un criterio di separazione per competenze in: gestione degli utenti in utenti.h, gestione eventi in eventi.h, gestione delle prenotazioni in prenotazioni.h, gestione dei locali in locali.h e gestione dei files, strutture e gestione generale in utils.h.*

*[!] Ulteriori informazioni riguardanti le librerie sono specificate con la documentazione Doxygen.*

## **Dipendenza tra funzioni**

*Per miglior visibilità consultare l’immagine allegata nella cartella della documentazione.*

## **Pseudocodice**

### **Gestione Generale**

#### Carica Files

##### Carica File Utenti

* Viene aperto il file relativo agli utenti in modalità lettura;
* Se il file non viene aperto viene visualizzato un messaggio di errore altrimenti il file viene aperto e vengono contate le righe. Il numero di righe viene salvato in una variabile che fungerà da contatore di numero di elementi del vettore dinamico relativo agli utenti;
* Il vettore dinamico, il cui indirizzo viene passato tramite parametro viene inizializzato con una calloc (in modo da azzerare anche il contenuto delle celle di memoria) (struct \* numero righe);
* Viene letta ogni riga del file interessato e il buffer che la contiene viene parserizzato (il carattere delimitatore è ‘;’). Il parser avviene con una funzione custom di strtok in modo da poter restituire null se il parser trova doppio delimitatore che simboleggia campo vuoto.
* I campi che vengono individuati saranno inseriti all’interno di una struct che si troverà nell’indice che sarà indicato dal numero della riga del buffer. (Esempio: riga 5 i campi verranno inseriti nella struct che si trova in array[4].
* La funzione restituisce 0 se l’operazione è terminata con successo, -1 se ci sono stati problemi di apertura del file.

##### Carica File Locali

* Viene aperto il file relativo ai locali;
* Viene eseguito il medesimo algoritmo degli utenti modellato per i campi dei Locali.

##### Carica File Eventi

* Viene aperto il file relativo agli eventi;
* Viene eseguito il medesimo algoritmo degli utenti modellato per i campi degli Eventi.

##### Carica File Prenotazioni

* Viene aperto il file relativo alle prenotazioni;
* Viene eseguito il medesimo algoritmo degli utenti modellato per i campi delle Prenotazioni

#### Salva Files

##### Salva File Utenti

* Viene eseguito un controllo sul vettore dinamico, se questo risulterà non vuoto (e quindi è presente almeno 1 struct di informazioni) il programma prosegue, altrimenti ritorna -1.
* Viene aperto un nuovo file relativo agli utenti in modalità scrittura (con consecutiva cancellazione dei dati; Se non è possibile aprire il file la funzione ritorna valore -1.
* Viene effettuato un ciclo che verrà eseguito n volte dove n avrà il valore della variabile che registrava il numero di struct presenti del vettore dinamico.
* Una volta che la funzione avrà scritto tutte le righe su file verrà azzerata la variabile che conservava il numero di righe e il vettore dinamico sarà deallocato con la funzione free(); Se tutto questo va a successo la funzione ritorna 0

##### Salva File Locali

* Viene eseguito un controllo sul vettore dinamico, se questo risulterà non vuoto (e quindi è presente almeno 1 struct di informazioni) il programma prosegue, altrimenti ritorna -1.
* Viene eseguito lo stesso algoritmo relativo al salvataggio degli utenti modellato per salvare i campi presenti nelle struct relative ai Locali.

##### Salva File Eventi

* Viene eseguito un controllo sul vettore dinamico, se questo risulterà non vuoto (e quindi è presente almeno 1 struct di informazioni) il programma prosegue, altrimenti ritorna -1.
* Viene eseguito lo stesso algoritmo relativo al salvataggio degli utenti modellato per salvare i campi presenti nelle struct relative agli Eventi.

##### Salva File Prenotazioni

* Viene eseguito un controllo sul vettore dinamico, se questo risulterà non vuoto (e quindi è presente almeno 1 struct di informazioni) il programma prosegue, altrimenti ritorna -1.
* Viene eseguito lo stesso algoritmo relativo al salvataggio degli utenti modellato per salvare i campi presenti nelle struct relative alle Prenotazioni.

#### Conta Righe

* Viene aperto il file a cui punta il parametro inserito e per ogni riga che viene lette viene incrementata una variabile contatore.
* La funzione resetta il puntatore all’inizio del file e ritorna il numero delle righe.

### **Gestione Utenti**

#### Aggiungi Utente

* Dichiarazione di una struct utente temporanea
* Visualizzazione titolo del modulo di registrazione
* Vengono richiesti in input i campi di registrazione e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati. (Esempio: il nickname non deve già esistere)
* Qualora il tipo di input fosse errato o viene inserito un campo già presente in memoria viene visualizzato un messaggio di errore ed in seguito ci sarà la possibilità di riprovare o di uscire dalla funzione
* Se i dati inseriti risultano corretti, i campi della struct temporanea saranno copiati in una nuova cella di memoria del vettore dinamico di riferimento attraverso un realloc della memoria di dimensione ++n dove n rappresenta il valore contenuto nella variabile che rappresenta il numero delle righe del vettore.
* Viene deallocata la struct temporanea.

#### Ricerca Utente

* Viene passato come parametro la chiave di ricerca (nickname) univoco per ogni utente.
* Viene effettuata una ricerca lineare in tutto il vettore dinamico di riferimento e qualora venisse individuata una corrispondenza viene restituito il puntatore alla struct all’interno del vettore dinamico (Per intenderci l’indirizzo di array[i] dove i rappresenta l’offset dell’utente individuato).
* Qualora la chiave non dovesse essere individuata viene restituita una struct nulla.

#### Modifica Utente

* Viene richiesto in input il nickname dell’utente da modificare
* Viene eseguito l’algoritmo di ricerca passando per parametro la chiave
* Se la ricerca ritorna l’indirizzo della struct di riferimento, viene visualizzato il titolo del modulo
* Viene creata una struct temporanea dove viene già inserito il nickname dell’utente
* Vengono richiesti in input i campi di registrazione assieme ad una visualizzazione dei dati attuali e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati. [!] Non è possibile modificare il nickname e se non si vuole modificare un campo è sufficiente premere il tasto invio
* Se i dati inseriti risultano corretti, i campi della struct nel vettore saranno sostituiti dalla struct temporanea che successivamente verrà deallocata.

### **Gestione Locali**

#### Aggiungi Locale

* Dichiarazione di una struct locale temporanea
* Viene letto l’indice AUTO\_INCREMENT relativo ai Locali
* Visualizzazione titolo del modulo di inserimento
* Vengono richiesti in input i campi e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati.
* Qualora il tipo di input fosse errato ci sarà la possibilità di riprovare o di uscire dalla funzione
* Il campo codice della struct locale conterrà il valore presente nell’index dei valori AUTO\_INCREMENT relativo ai Locali.
* Se i dati inseriti risultano corretti, i campi della struct temporanea saranno copiati in una nuova cella di memoria del vettore dinamico di riferimento attraverso un realloc della memoria di dimensione ++n dove n rappresenta il valore contenuto nella variabile che rappresenta il numero delle righe del vettore.
* Viene deallocata la struct temporanea.

#### Ricerca Locale

* Viene passato come parametro il presunto codice del Locale da ricercare.
* Viene effettuata una ricerca lineare in tutto il vettore dinamico di riferimento e qualora venisse individuata una corrispondenza viene restituito il puntatore alla struct all’interno del vettore dinamico (Per intenderci l’indirizzo di array[i] dove i rappresenta l’offset del locale individuato).
* Qualora la chiave non dovesse essere individuata viene restituita una struct nulla.

#### Modifica Locale

* Viene richiesto in input il codice del locale da modificare
* Viene eseguito l’algoritmo di ricerca passando per parametro la chiave
* Se la ricerca ritorna l’indirizzo della struct di riferimento, viene visualizzato il titolo del modulo
* Viene creata una struct temporanea dove viene già inserito il codice del locale
* Vengono richiesti in input i campi di registrazione assieme ad una visualizzazione dei dati attuali e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati. [!] Non è possibile modificare il codice e se non si vuole modificare un campo è sufficiente premere il tasto invio
* Se i dati inseriti risultano corretti, i campi della struct nel vettore saranno sostituiti dalla struct temporanea che successivamente verrà deallocata.

#### Cancella Locale

* Viene richiesto in input il codice del locale da cancellare
* Viene eseguito l’algoritmo di ricerca passando per parametro la chiave
* Se la ricerca ritorna l’indirizzo della struct di riferimento, viene richiesta una conferma per l’eliminazione, se confermata la struct di riferimento diventerà null e verrà eseguita una realloc del vettore dinamico che sarà riallocato a –n posizioni dove n rappresenta il numero attuale delle righe.
* Affinchè non ci siano perdite di dati, prima del realloc viene effettuato un ordinamento per struct NON-NULLE per preservare l’integrità di struct valide in coda al vettore.
* Verrà effettuata una diagnostica di integrità dei dati che eliminerà tutte le righe degli altri vettori in cui era presente il codice della struct eliminata effettuando relativi realloc e salvataggio di dati.

### **Gestione Eventi**

#### Aggiungi Evento

* Dichiarazione di una struct evento temporanea
* Viene letto l’indice AUTO\_INCREMENT relativo agli Eventi
* Visualizzazione titolo del modulo di inserimento
* Vengono richiesti in input i campi e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati.
* Durante l’inserimento dei campi, nel momento in cui bisognerà selezionale un locale verrà mostrata una lista di tutti i locali disponibili con codice annesso e si attenderà l’input del locale.
* Il codice inserito verrà passato per parametro alla funzione ricerca Locale, che, se ritornerà l’indirizzo di una struct nel vettore verrà valutato come codice valido, se ritornerà NULL verrà valutato come errato.
* Qualora il tipo di input fosse errato ci sarà la possibilità di riprovare o di uscire dalla funzione
* Il campo codice della struct evento conterrà il valore presente nell’index dei valori AUTO\_INCREMENT relativo agli Eventi.
* Se i dati inseriti risultano corretti, i campi della struct temporanea saranno copiati in una nuova cella di memoria del vettore dinamico di riferimento attraverso un realloc della memoria di dimensione ++n dove n rappresenta il valore contenuto nella variabile che rappresenta il numero delle righe del vettore.
* Viene deallocata la struct temporanea.

#### Ricerca Evento

* Viene passato come parametro da ricercare.
* Viene effettuata una ricerca lineare in tutto il vettore dinamico di riferimento e qualora venisse individuata una corrispondenza viene restituito il puntatore alla struct all’interno del vettore dinamico (Per intenderci l’indirizzo di array[i] dove i rappresenta l’offset del locale individuato).
* Qualora la chiave non dovesse essere individuata viene restituita una struct nulla.

#### Visualizza Spettatori Evento

* Viene richiesto di operare su un Evento, e perciò verrà chiesto all’utente di selezionare un parametro per la ricerca.
* Una volta effettuata la ricerca, se la funzione restituirà un puntatore ad una struct allora verrà chiamato il vettore relativo alle prenotazioni in cui avverrà una ricerca per trovare tute le corrispondenze relative al Codice dell’evento. Per ogni biglietto trovato relativo a quell’evento, viene ricercato l’utente di riferimento (eseguendo una ricerca dell’utente attraverso nickname) e verranno stampati tutti in ordine di Nickname senza ripetizioni causate da acquisti di più biglietti.

#### Modifica Evento

* Viene richiesto in input il codice dell’evento da modificare
* Viene eseguito l’algoritmo di ricerca passando per parametro la chiave
* Se la ricerca ritorna l’indirizzo della struct di riferimento, viene visualizzato il titolo del modulo
* Viene creata una struct temporanea dove viene già inserito il codice dell’evento
* Vengono richiesti in input i campi di registrazione assieme ad una visualizzazione dei dati attuali e contemporaneamente avviene un controllo di sicurezza e di conformità dei dati. [!] Non è possibile modificare il codice e se non si vuole modificare un campo è sufficiente premere il tasto invio
* Durante l’inserimento dei campi, nel momento in cui bisognerà selezionale un locale verrà mostrata una lista di tutti i locali disponibili con codice annesso e si attenderà l’input del locale.
* Il codice inserito verrà passato per parametro alla funzione ricerca Locale, che, se ritornerà l’indirizzo di una struct nel vettore verrà valutato come codice valido, se ritornerà NULL verrà valutato come errato.
* Qualora il tipo di input fosse errato ci sarà la possibilità di riprovare o di uscire dalla funzione
* Se i dati inseriti risultano corretti, i campi della struct nel vettore saranno sostituiti dalla struct temporanea che successivamente verrà deallocata.

#### Annulla Evento

* Viene richiesto in input il codice dell’evento da cancellare
* Viene eseguito l’algoritmo di ricerca passando per parametro la chiave
* Se la ricerca ritorna l’indirizzo della struct di riferimento, viene richiesta una conferma per l’eliminazione, se confermata la struct di riferimento diventerà null e verrà eseguita una realloc del vettore dinamico che sarà riallocato a –n posizioni dove n rappresenta il numero attuale delle righe.
* Affinchè non ci siano perdite di dati, prima del realloc viene effettuato un ordinamento per struct NON-NULLE per preservare l’integrità di struct valide in coda al vettore.
* Verrà effettuata una diagnostica di integrità dei dati che eliminerà tutte le righe degli altri vettori in cui era presente il codice della struct eliminata effettuando relativi realloc e salvataggio di dati.

#### Visualizza Incasso

* Viene richiesto un parametro che identifica l’evento, verrà effettuata una ricerca e stampati possibili risultati multipli.
* Una volta selezionato l’evento interessato, verrà creata una variabile accumulatrice che andrà a sommare tutti gli importi dei biglietti nella quale compare il codice dell’evento ricercato.
* Viene visualizzato sullo schermo l’importo totale.

### **Gestione Prenotazioni**

#### Visualizza Sala e Disponibilità

* Viene richiesto un parametro che identifica l’evento, verrà effettuata una ricerca e stampati possibili risultati multipli.
* Una volta selezionato l’evento interessato, viene visualizzata una simulazione dalla sala con numero di file e poltrone. I pixel colorati di verde indicano le poltrone libere, quelle di rosso le poltrone occupate. Se le poltrone “disponibili” sono di colore giallo significa che l’evento è un evento non più prenotabile.

#### Vendita di un biglietto

* Viene eseguito l’algoritmo di Visualizzazione della sala e controllo disponibilità
* Se l’evento è ancora prenotabile viene chiesto all’utente di indicare la poltrona da prenotare
* Viene calcolato il costo della poltrona (funzione matematica che definisce un costo intermedio in base alla posizione rispetto i costi della prima e ultima fila)
* Viene richiesta una conferma e successivamente il biglietto viene creato (viene generata una struct prenotazione con i dati relativi e viene inserita dinamicamente nel vettore)

#### Visualizza Eventi a cui un Utente ha partecipato

* Viene richiesto in input un parametro per ricercare un utente
* Se l’algoritmo di ricerca utenti va a buon fine (il ritorno della funzione è non nullo) vengono ricercate tutte le corrispondenze in cui è presente il nickname dell’utente trovato
* Da ogni corrispondenza viene prelevato il codice relativo all’evento e ne vengono presi i relativi dati (nome data etc) e vengono visualizzati sullo schermo.

# **TESTING**

*Nella seguente tabella sono registrati i casi di testing principali, ignorando i testing banali (anche quelli ripetitivi e/o simili) e riportando solo quelli particolari.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CODICE REQUISITO** | **CODICE TEST** | **NOME** | **DESCRIZIONE TEST** | **INPUT** | **RISULTATO ATTESO** | **RISULTATO OTTENUTO** |
| *RM00* | *TM00* | *Visualizzazione menù* | *Inserimento scelta non valida* | *Test* | *Opzione indicata non valida* | *Opzione indicata non valida* |
| *RM00* | *TM01* | *Visualizzazione menù* | *Inserimento scelta valida + non valida* | *1 e* | *Il programma ignora l’input non valido(e) ed avvia la registrazione dell’utente (1)* | *Il programma ignora l’input non valido(e) ed avvia la registrazione dell’utente (1)* |
| *RU00* | *TU00* | *Registrazione nuovo utente – Nickname* | *Inserimento nickname già esistete* | *Reav* | *L’utente è già presente in memoria* | *L’utente è già presente in memoria* |
| *RU00* | *TU01* | *Registrazione nuovo utente – Nickname* | *Inserimento nickname non valido* | *The Simo06* | *Puoi inserire solo lettere* | *Puoi inserire solo lettere* |
| *RU00* | *TU02* | *Registrazione nuovo utente – Nickname* | *Inserimento nickname non valido* | *Nickname troppo lungo o corto* | *Il nickname deve avere minimo 4 caratteri e massimo 15* | *Il nickname deve avere minimo 4 caratteri e massimo 15* |
| *RU00* | *TU03* | *Registrazione nuovo utente – E-mail* | *Inserimento e-mail non valida* | *E-mail troppo corta o troppo lunga* | *L’e-mail deve avere minimo 12 caratteri e massimo 63* | *L’e-mail deve avere minimo 12 caratteri e massimo 63* |
| *RU00* | *TU04* | *Registrazione nuovo utente – E-mail* | *Inserimento e-mail non valida* | *E-mail non rispetta il formato imposto* | *Puoi inserire solo lettere, numeri! Formato: xxxx@xxxx.xxxx* | *Puoi inserire solo lettere, numeri! Formato: xxxx@xxxx.xxxx* |
| *RU00* | *TU05* | *Registrazione nuovo utente - Cognome/Nome* | *Inserimento cognome/nome non valido* | *Digirolamo 98/ Simon e98* | *Puoi inserire solo lettere e spazi* | *Puoi inserire solo lettere e spazi* |
| *RU00* | *TU06* | *Registrazione nuovo utente - Genere* | *Inserimento genere non valido* | *Test* | *Puoi inserire solo un numero tra quelli indicati* | *Puoi inserire solo un numero tra quelli indicati* |
| *RU00* | *TU07* | *Registrazione nuovo utente - Genere* | *Inserimento genere valido + non valido* | *2 43* | *Il programma ignora l’input non valido (43) e non segnala nessun errore* | *Il programma ignora l’input non valido (43) e non segnala nessun errore* |
| *RU00* | *TU08* | *Registrazione nuovo utente - Data di nascita - giorno* | *Inserimento giorno non valido* | *Test* | *Giorno indicato non valido* | *Giorno indicato non valido* |
| *RU00* | *TU09* | *Registrazione nuovo utente - Data di nascita - giorno* | *Inserimento giorno valido + non valido* | *31 e* | *Il programma ignora l’input non valido (e) e non segnala nessun errore* | *Il programma ignora l’input non valido (e) e non segnala nessun errore* |
| *RU00* | *TU10* | *Registrazione nuovo utente - Data di nascita – giorno + mese* | *Inserimento mese non valido in base al giorno inserito* | *Giorno: 31*  *Mese: 2* | *Il giorno indicato non è compreso nel mese scelto* | *Il giorno indicato non è compreso nel mese scelto* |
| *RU00* | *TU11* | *Registrazione nuovo utente - Data di nascita – anno* | *Inserimento anno non valido* | *206* | *Anno indicato non valido* | *Anno indicato non valido* |
| *RU00* | *TU12* | *Registrazione nuovo utente – Num. telefono* | *Inserimento numero non valido* | *Numero troppo lungo o troppo corto* | *Il numero di telefono deve avere minimo 10 cifre e massimo 19* | *Il numero di telefono deve avere minimo 10 cifre e massimo 19* |
| *RU00* | *TU13* | *Registrazione nuovo utente – Num. telefono* | *Inserimento numero non valido* | *328876fd 323* | *Puoi inserire solo numeri* | *Puoi inserire solo numeri* |
| *RU00* | *TU14* | *Registrazione nuovo utente – Genere musicale* | *Inserimento genere musicale non valido* | *Test* | *Opzione indicata non valida* | *Opzione indicata non valida* |
| *RU00* | *TU15* | *Registrazione nuovo utente – Genere musicale* | *Inserimento genere musicale valido + non valido* | *2 p* | *Il programma ignora l’input non valido (p) e non segnala nessun errore* | *Il programma ignora l’input non valido (p) e non segnala nessun errore* |
| RU04 | TU16 | Visualizza utente | Inserimento nickname non esistente | Prova | L’utente ‘Prova’ non esiste! | L’utente ‘Prova’ non esiste! |
| RE00 | TE00 | Registrazione nuovo evento  -Orario | Inserimento orario non valido | Ora inizio: 10:10  Ora fine 10:10 | Hai inserito una durata dell’evento non valida | Hai inserito una durata dell’evento non valida |
| RE00 | TE01 | Registrazione nuovo evento  -Orario | Inserimento orario non valido | Prova | L’ora indicata non è valida | L’ora indicata non è valida |
| RE00 | TE02 | Registrazione nuovo evento  -Locale | Inserimento locale non esistente | Prova | Non è stato trovato alcun locale col nome ‘Prova’! | Non è stato trovato alcun locale col nome ‘Prova’! |
| RE00 | TE03 | Registrazione nuovo evento  -Costo prima e ultima fila | Inserimento costo ultima fila maggiore del costo prima fila | Costo prima fila: 10  Costo ultima fila: 12 | Il costo dell’ultima fila non può essere maggiore di quello della prima fila | Il costo dell’ultima fila non può essere maggiore di quello della prima fila |
| RE02 | TE04 | Elimina evento | Inserimento nome evento non esistente | Prova123 | L’evento ricercato non esiste! | L’evento ricercato non esiste! |
| RU03 | TB00 | Vendi biglietto  -inserimento evento | Inserimento evento sold out | SunSetDisco | Hai indicato un evento sold out! Ritornerai al menù principale | Hai indicato un evento sold out! Ritornerai al menù principale |
| RU03 | TB01 | Vendi biglietto  -inserimento evento | Inserimento evento passato | Vip Music | Hai selezionato un evento passato. | Hai selezionato un evento passato. |
| RU03 | TB02 | Vendi biglietto  -inserimento nickname | Inserimento nickname non esistente nel file CSV | TESTNOME | L’utente ‘TESTNOME’ non è stato trovato. | L’utente ‘TESTNOME’ non è stato trovato. |
| RU03 | TB03 | Vendi biglietto  -inserimento fila e posto | Inserimento posto già occupato | FILA: 11  POSTO: 5 | Il posto selezionato non è disponibile! | Il posto selezionato non è disponibile! |
| RL01 | TL00 | Modifica posti Locale con conseguenze sui biglietti prenotati in quei posti | Eliminazione ultimi posti delle file, con due biglietti prenotati | POSTI: 11 🡪 POSTI:10 | Segnalazione da parte della diagnostica. Rimborso dei due biglietti coinvolti | Segnalazione da parte della diagnostica. Rimborso dei due biglietti coinvolti |

## **SUDDIVISIONE DELLA RESPONSABILITA’**

|  |  |  |
| --- | --- | --- |
| **NOME FUNZIONI** |  |  |
| **UTILS** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| caricaIndexs |  | **x** |
| salvaIndexs |  | **X** |
| contaLinee | **X** | **X** |
| stringaValida | **X** | **X** |
| myParser |  | **X** |
| myLower | **X** | **X** |
| setColor | **X** | **X** |
| inizializzaStringa | **X** | **X** |
| contaCifre | **X** |  |
| differenzaOrari | **X** |  |
| InserisciData | **X** |  |
| mergeSort | **X** |  |
| **UTENTI** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| caricaUtenti |  | **X** |
| salvaUtenti |  | **X** |
| aggiungiUtente | **X** | **x** |
| visualizzaUtete | **X** | **X** |
| visualizzaUtenti | **X** | **X** |
| modificaUtente | **X** | **X** |
| ricercaUtente | **X** | **X** |
| esisteUtente | **X** | **X** |
| mergeUtenti | **X** |  |
| **PRENOTAZIONI** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| caricaPrenotazioni |  | **X** |
| salvaPrenotazioni |  | **X** |
| visualizzaBiglietto | **X** |  |
| visualizzaDisponibilitaSala | **X** |  |
| vendiBiglietto | **X** |  |
| rimborsaBiglietto |  | **X** |
| differenzaDate | **X** |  |
| ricercaPrenotazione | **X** |  |
| ricercaPrenotazioneByNumero | **X** | **X** |
| visualizzaSpettatoriEvento | **X** | **X** |
| visualizzaEventiUtente | **X** | **X** |
| visualizzaIncassoEvento | **X** | **X** |
| ordinaEventi | **X** | **X** |
| **LOCALI** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| caricaLocali |  | **X** |
| salvaLocali |  | **X** |
| visualizzaLocale | **X** | **X** |
| visualizzaLocali | **X** | **X** |
| aggiungiLocale | **X** | **X** |
| modificaLocale | **X** | **X** |
| eliminaLocale | **X** | **X** |
| ricercaLocale | **X** | **X** |
| ricercaLocaleByCodice | **X** | **X** |
| mergeLocali | **X** |  |
| **EVENTI** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| caricaEventi |  | **X** |
| salvaEventi |  | **X** |
| aggiungiEvento | **X** | **X** |
| modificaEvento | **X** | **X** |
| eliminaEvento | **X** | **X** |
| visualizzaEvento | **X** | **X** |
| modificaEvento | **X** | **X** |
| eliminaEvento | **X** | **X** |
| visualizzaEvento | **X** | **X** |
| visualizzaEventi | **X** | **X** |
| ricercaEventoByCodice | **X** | **X** |
| ricercaEventoByData | **X** | **X** |
| ricercaEventoByString | **X** |  |
| ricercaEvento | **X** |  |
| **MAIN** | **FRANCESCO DIGIROLAMO** | **VINCENZO BIALLO** |
| eseguiDiagnostica |  | **X** |
| visualizzaMenu | **X** | **X** |

## **CONSIDERAZIONI FINALI**

Il software necessità della presenza di tutti i file ad esso legati. Nel caso in cui ne mancasse 1, grazie alla diagnostica implementata il programma continuerà a funzionare creando il file mancante ma vuoto.

Il programma presenta una struttura stabile, sono stati inseriti molteplici controlli sintattici e logici su ogni inserimento di input, che hanno ottenuto risultati eccellenti in vari test, effettuati anche da un gruppo di tester. L’unico controllo che non è al 100% efficiente è quello effettuato sull’e-mail, nel programma non sono state inserite le espressioni regolari con la libreria Regex, questo controllo garantisce una correttezza sintattica dell’e-mail al 95%.

Per rendere il programma visivamente più gradevole è stata implementata una funzione per colorare l’output del pannello grafico.

Alla consegna il programma comprende tutte le funzionalità richieste dalla traccia, più altre funzionalità inserite dagli autori in modo da rendere il programma più soddisfacente da utilizzare. E’ completo al 100% e utilizzabile immediatamente per l’uso per il quale è stato creato.