Università degli Studi di Salerno
Corso di Ingegneria del Software

CineNow
Database Design Document
Versione 1.1

Data: 06/01/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Emanuele Iovane	0512120565
Armando Vigliotti	0512117739
Antonio Caiazzo	0512117751

Scritto da:	Caiazzo Antonio, Iovane Emanuele, Vigliotti Armando

Revision History

Data	Versione	Descrizione	Autore
22/11/2024	0.1	Creazione del Database Design Document	Tutto il team
24/11/2024	1.0	Revisione finale del documento e primo rilascio	Tutto il team
06/01/2025	1.1	Aggiunta sezione script, e revisione documento	Tutto il team

Indice

1. INTRODUZIONE

Questo documento descrive la progettazione dettagliata del database per il sistema CineNow. Include il modello concettuale e lo schema relazionale per supportare la gestione di tutte le funzionalità chiave del sistema, come la gestione degli utenti, la prenotazione dei biglietti, la gestione dei film e delle sale, e il monitoraggio della disponibilità dei posti. Per gestire la persistenza dei dati, si è scelto di utilizzare un database relazionale, gestendo tramite il DBMS MySQL. La scelta di un database relazionale per questo sistema, è stata definita perché il suo modello tabellare strutturato è ideale per gestire dati, organizzati in modo chiaro e definito. La capacità di supportare le proprietà ACID garantisce operazioni sicure e consistenti, fondamentali per la gestione affidabile delle transazioni. Inoltre, i vincoli di integrità, attraverso l'uso di chiavi primarie ed esterne, permettono di mantenere un alto livello di qualità e coerenza dei dati, riducendo la necessità di controlli complessi a livello applicativo. Inoltre l'utilizzo di query SQL consente interrogazioni avanzate e aggregazioni complesse, rendendo questo approccio estremamente versatile. È stato scelto MySQL come DBMS perché unisce affidabilità e facilità di utilizzo, nonché alta compatibilità con molteplici piattaforme e linguaggi, tra cui JAVA.

Dal class diagram del sistema presente nel RAD_CineNow.pdf , le modifiche apportate al modello concettuale del database sono le seguenti:

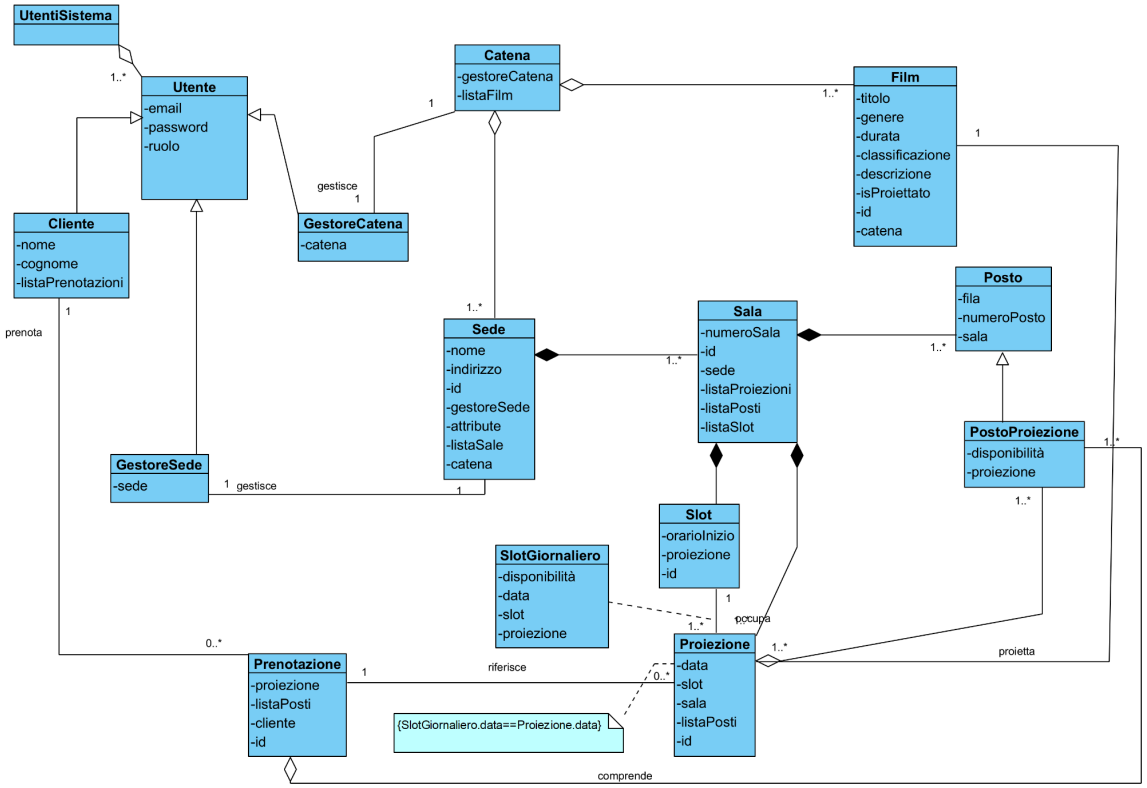
- La classe associativa SlotGiornaliero è stata accorpata nella relazione tra l'entità Proiezione e l'entità SlotBase.
- L'entità "*POSTO_PROIEZIONE*" mappa l'attributo postiProiezione della classe Proiezione. Questa entità tiene traccia dello stato dei posti associati a una proiezione, indicando se un posto è prenotato (true) o libero (false). Questa scelta è stata fatta per normalizzare i dati e semplificare la gestione dello stato dei posti per ogni proiezione.
- Non è stata creata un'entità per la classe Catena, in quanto la classe Catena rappresenta l'intero sistema, e quindi non necessaria ai fini della gestione dei dati.
- Le locandine dei film, verranno memorizzate nel database come stringhe, indicative del path dell'immagine, che invece verrà direttamente caricata nella cartella "images" dell'applicazione, sul server.

1.1. Struttura del documento

Questo documento è suddiviso in diverse parti per fornire una visione chiara e dettagliata della progettazione del database per CineNow:

- La **prima parte** introduce il contesto e gli obiettivi del database, spiegando le motivazioni delle scelte progettuali.
- La **seconda parte** descrive il **modello concettuale** del DB.
- La **terza parte** illustra lo **schema relazionale**, fornendo una mappatura chiara tra il modello concettuale e la struttura tabellare del database.
- La **quarta parte** dettaglia la **struttura tabellare**, includendo i tipi di dati, i vincoli di integrità e le chiavi primarie ed esterne per ogni tabella.

2. MODELLO CONCETTUALE DEL DATABASE



Dato il class diagram per il sistema CineNow già definito nel documento RAD_CineNow.pdf. Andando a specificare gli attributi utili per la memorizzazione ed esplicitato gli attributi derivati dalle relazioni, abbiamo ottenuto il Class Diagram mostrato sopra. Nell'analisi del diagramma concettuale del sistema, sono state prese le seguenti decisioni:

- la classe associativa SlotGiornaliero nella relazione tra proiezione e Slot è stata mantenuta, utilizzando una relazione tra Slot e Proiezione, con chiavi esterne verso entrambe le entità, ed eliminando l'attributo implicito disponibilità. LA data è stata mantenuta esclusivamente nella proiezione.
- è stata definita l'entità PostoProiezione, che mappa la relazione tra la Proiezione e Posto. PostoProiezione tiene traccia dei posti della proiezione e del loro stato (true se prenotato, false se libero). Si è scelto per chiarezza di definire una nuova entità anziché mantenere solo la relazione, per semplificare le richieste al database.
- non viene memorizzata la Catena, essendo unica e rappresentativa del sistema.
- Si è definito il ***Posto*** come un'entità debole. La chiave primaria è composta dalla chiave primaria di Sala e dagli attributi fila (rappresenta il numero della fila) e numero (rappresenta il numero del posto).
- è stato aggiunto alla sala l'attributo "*capienza*", definito dal numero di posti di una sala.

- lo Slot modella gli slot orari disponibili per effettuare le proiezioni. Essi saranno fissi per il nostro sistema, e andranno dalle 18:00 alle 22:00, ogni 30 minuti.
- la classe UtentiSistema, non viene memorizzata, essendo la definizione della tabella utenti.

Il database è stato progettato per minimizzare la ridondanza dei dati e ottimizzare le operazioni di interrogazione, ad esempio separando gli attributi dei posti (fila, numero) dagli stati (prenotato/libero).

3. SCHEMA RELAZIONALE DEL DATABASE

Entità/Relazione	**Attributi**	**Chiave primaria**	**Chiave esterna**
UTENTE	email, password, ruolo	email	
CLIENTE	UTENTE.email, nome, cognome	UTENTE.email	UTENTE.email
GEST_SEDE	UTENTE.email, SEDE.id	UTENTE.email	UTENTE.email, SEDE.id
SEDE	id, nome, via, cap, città	id	
SALA	id, SEDE.id, numeroSala, capienza	id	SEDE.id
POSTO	SALA.id, fila, numero	SALA.id, fila, numero	SALA.id
PROIEZIONE	id, FILM.id, SALA.id, data	id	FILM.id, SALA.id
POSTO_PROIEZIONE	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id, stato	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id
FILM	id, titolo, genere, classificazione, durata, descrizione, locandina, isproiettato	id	id
SLOT	id, oraInizio	id	
PRENOTAZIONE	id, CLIENTE.email, PROIEZIONE.id	id	CLIENTE.email, PROIEZIONE.id
Occupa {PRENOTAZIONE - POSTO_PROIEZIONE}	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id, PRENOTAZIONE.id	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id, PRENOTAZIONE.id	SALA.id, POSTO.fila, POSTO.numero, PROIEZIONE.id, PRENOTAZIONE.id
Proiezione_Slot	PROIEZIONE.id, SLOT.id	PROIEZIONE.id, SLOT.id	PROIEZIONE.id, SLOT.id

NOTE TABELLA: *ENTITÀ.attributo*, indica l'attributo dell'entità posseditrice. Le relazioni seguono il seguente formato: *Relazione{ ENTITÀCOINVOLTA1 - ENTITÀCOINVOLTA2 }*

4. STRUTTURA TABELLARE DEL DATABASE

Si indica con **PK la chiave primaria**, con **FK la chiave esterna**,con **PKC una chiave** candidata che fa parte della chiave **primaria composta.** Si indica con il formato ENTITA.attributo, il riferimento all'attributo della specifica ENTITÀ. Si indica con DEFAULT(valore), il valore di default dell'attributo.

UTENTE		
Attributo	**Tipo**	**Vincoli**
email	string	PK
password	string	not null
ruolo	string	not null

CLIENTE		
Attributo	**Tipo**	**Vincoli**
email	string	PK & FK(UTENTE.email)
nome	string	not null
cognome	sting	not null

GEST_SEDE		
Attributo	**Tipo**	**Vincoli**
email	string	PK & FK(UTENTE.email)
id_sede	integer	FK(SEDE.id)

SEDE		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
nome	string	not null

via	string	not null
città	string	not null
cap	string(5 caratteri)	not null

SALA		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
id_sede	integer	FK(SEDE.id)
numero	integer	not null
capienza	integer	not null

POSTO		
Attributo	**Tipo**	**Vincoli**
id_sala	integer	PKC & FK(SALA.id)
numero	integer	PKC
fila	string	PKC

FILM		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
titolo	string	not null
genere	string	not null
classificazione	string	not null
durata	integer	not null

locandina	string	not null
descrizione	string	not null
isProiettato	boolean	DEFAULT(false)

SLOT		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
ora_inizio	time	not null

PROIEZIONE		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
data	date	not null
id_film	integer	FK(FILM.id)
id_sala	integer	FK(SALA.id)

POSTO_PROIEZIONE		
Attributo	**Tipo**	**Vincoli**
id_sala	integer	PKC & FK(SALA.id)
fila	string	PKC & FK(POSTO.fila)
numero_posto	integer	PKC & FK(POSTO.numero)
id_proiezione	integer	PKC & FK(PROIEZIONE.id)
stato	boolean	DEFAULT(true)

PRENOTAZIONE		
Attributo	**Tipo**	**Vincoli**
id	integer	PK
email_cliente	string	FK(CLIENTE.email)
id_proiezione	integer	FK(PROIEZIONE.id)

Occupa		
Attributo	**Tipo**	**Vincoli**
id_sala	integer	PKC & FK(SALA.id)
fila	string	PKC & FK(POSTO.fila)
numero_posto	integer	PKC & FK(POSTO.numero)
id_proiezione	integer	PKC & FK(PROIEZIONE.id)
id_prenotazione	integer	PKC & FK(PRENOTAZIONE.id)

PPROIEZIONE_SLOT		
Attributo	**Tipo**	**Vincoli**
id_proiezione	integer	PKC & FK(PROIEZIONE.id)
id_slot	integer	PKC & FK(SLOT.id)

5. SCRIPT SQL - CREAZIONE DB

```
CREATE DATABASE CineNow;
USE CineNow;

CREATE TABLE utente (
   email VARCHAR(255) PRIMARY KEY,
   password VARCHAR(255) NOT NULL,
   ruolo VARCHAR(50) NOT NULL
);

CREATE TABLE cliente (
   email VARCHAR(255),
   nome VARCHAR(255) NOT NULL,
   cognome VARCHAR(255) NOT NULL,
   PRIMARY KEY (email),
   FOREIGN KEY (email) REFERENCES utente(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE sede (
   id INT PRIMARY KEY AUTO_INCREMENT,
   nome VARCHAR(255) NOT NULL,
   via VARCHAR(255) NOT NULL,
   città VARCHAR(255) NOT NULL,
   cap CHAR(5) NOT NULL
);

CREATE TABLE gest_sede (
   email VARCHAR(255),
   id_sede INT,
   PRIMARY KEY (email, id_sede),
   FOREIGN KEY (email) REFERENCES utente(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (id_sede) REFERENCES sede(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```
CREATE TABLE sala (
   id INT PRIMARY KEY AUTO_INCREMENT,
   id_sede INT,
   numero INT NOT NULL,
   capienza INT NOT NULL,
   FOREIGN KEY (id_sede) REFERENCES sede(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE posto (
   id_sala INT,
   fila CHAR(1),
   numero INT,
   PRIMARY KEY (id_sala, fila, numero),
   FOREIGN KEY (id_sala) REFERENCES sala(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE film (
   id INT PRIMARY KEY AUTO_INCREMENT,
   titolo VARCHAR(255) NOT NULL,
   genere VARCHAR(100) NOT NULL,
   classificazione VARCHAR(50) NOT NULL,
   durata INT NOT NULL,
   locandina MEDIUMBLOB,
   descrizione TEXT NOT NULL,
   is_proiettato BOOLEAN DEFAULT FALSE
);

CREATE TABLE slot (
   id INT PRIMARY KEY AUTO_INCREMENT,
   ora_inizio TIME NOT NULL
);

CREATE TABLE proiezione (
   id INT PRIMARY KEY AUTO_INCREMENT,
   data DATE NOT NULL,
   id_film INT,
   id_sala INT,
   FOREIGN KEY (id_film) REFERENCES film(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
```

```
    FOREIGN KEY (id_sala) REFERENCES sala(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE proiezione_slot (
    id_proiezione INT,
    id_slot INT,
    PRIMARY KEY (id_proiezione, id_slot),
    FOREIGN KEY (id_proiezione) REFERENCES proiezione(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (id_slot) REFERENCES slot(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE posto_proiezione (
    id_sala INT,
    fila CHAR(1),
    numero INT,
    id_proiezione INT,
    stato BOOLEAN DEFAULT TRUE,
    PRIMARY KEY (id_sala, fila, numero, id_proiezione),
    FOREIGN KEY (id_sala, fila, numero) REFERENCES posto(id_sala, fila, numero)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (id_proiezione) REFERENCES proiezione(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE prenotazione (
    id INT PRIMARY KEY AUTO_INCREMENT,
    email_cliente VARCHAR(255),
    id_proiezione INT,
    FOREIGN KEY (email_cliente) REFERENCES cliente(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (id_proiezione) REFERENCES proiezione(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```
CREATE TABLE occupa (
    id_sala INT,
    fila CHAR(1),
    numero INT,
    id_proiezione INT,
    id_prenotazione INT,
    PRIMARY KEY (id_sala, fila, numero, id_proiezione, id_prenotazione),
    FOREIGN KEY (id_sala, fila, numero) REFERENCES posto(id_sala, fila, numero)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (id_proiezione) REFERENCES proiezione(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (id_prenotazione) REFERENCES prenotazione(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```