**Relazione tecnica Web Application Videonoleggio (Rent Net)**

**Studente:** Gherardi Gianmattia  
**Matricola:** STA07049/L8  
**Progetto:** Videonoleggio (Rent Net)  
**Repository:** https://github.com/ggherardi/videonoleggio

1. **Tecnologie utilizzate**

Per la realizzazione del progetto sono state utilizzate le seguenti tecnologie:

* PHP 7
* DB MySQL (InnoDB)
* JavaScript ES 6
* jQuery 3.2.1
* BootStrap 4
* CSS3

Il gestionale è stato realizzato secondo il paradigma SPA (Single Page Application): non appoggiandosi ad alcun framework JavaScript esterno, è stato realizzato un meccanismo di gestione delle view ad hoc, utilizzando JavaScript e come, uniche librerie esterne, jQuery e BootStrap 4. L’accesso al DB è garantito tramite dei servizi realizzati in PHP 7, sotto forma di classi, che comunicano con altrettante classi JavaScript che fanno da proxy ed effettano le chiamate REST.  
Il server utilizzato è un Raspberry 3 modello B+, sul quale risiedono il DB e il file system dal quale attinge il web server.

1. **Comunicazione con il database**

Alcune operazioni svolte dagli utenti implicheranno la persistenza di dati sul database MySQL. Questi dati verranno comunicati tramite delle chiamate REST a dei servizi PHP, che a loro volta andranno ad interrogare il database per restituire infine al client una response in formato JSON.

Da notare che non tutte le richieste API potranno essere effettuate se l’utente non ha effettuato l’accesso al sito: alcuni endpoint necessitano, infatti, dei Token di sicurezza staccati durante l’autenticazione.

1. **Autenticazione e sicurezza**

All’accesso degli utenti, se l’autenticazione andrà a buon fine verrà creato un cookie http-only per mantenere la sessione di quell’utente attiva: questo cookie ha durata di 12 ore e conterrà i dati del contesto attuale più un token, criptati. La creazione del Token avviene utilizzando due diverse chiavi di cifratura e la stringa JSON contenente i dati. In questa maniera viene generato un Token che sarà indissolubilmente legato a queste informazioni, così da impedire tentativi di manomissione dei cookie e quindi delle chiamate ai servizi: infatti, alcuni endpoint (in particolare quelli che andranno a scrivere sul DB) necessitano della validazione del Token che verrà passato assieme al cookie criptato durante la chiamata; se la decriptazione non andasse a buon fine, la chiamata al servizio termina e viene restituito al client un messaggio di errore di autenticazione. Inoltre, per garantire una maggiore sicurezza, determinati endpoint verificheranno che la Delega dell’utente che ha effettuato la chiamata al sia sufficiente per poter consumare il servizio stesso. In caso contrario, verrà restituito un altro messaggio di accesso negato e verrà fatto un redirect su una pagina di errore.

1. **Navigazione del sito**

Essendo una SPA (Single Page Application), esisterà una singola pagina (index.html). Tramite il menù di navigazione un Controller si occuperà di richiamare la view corrispondente alla voce selezionata, e caricare HTML e JavaScript relativi, sostituendo l’HTML del container principale.

1. **Sistema di logging**

E’ stato creato un sistema di logging che permette di tracciare le operazioni eseguite dai servizi PHP. Verrà creato ogni giorno (eventualmente da suddividere ulteriormente) un file di log che scriverà le operazioni eseguite ed eventuali eccezioni. Ogni singola chiamata al servizio verrà dotata di un ID di correlazione (CorrelationID) che permetterà di tracciare in sequenza tutte le operazioni effettuate dal servizio stesso sul log, fino al termine dell’esecuzione.

1. **Analisi funzionale**

Al fine di realizzare un applicativo conforme alla esigenze del cliente, è stato necessario rivedere con il cliente stesso alcune delle richieste avanzate e dei dettagli relativi al funzionamento aziendale, così da poter risolvere ambiguità che avrebbero impedito una corretta redazione dello schema concettuale.

* “*La società gestisce 8 punti vendita in varie località; in ogni punto vendita lavorano da 1 a 6 addetti più il responsabile del negozio.*”

**Domanda**: ogni punto vendita è dotato di un proprio database, o si desidera centralizzare tutta la persistenza all’interno di un database centralizzato?

**Risposta**: desideriamo mantenere tutte le informazioni all’interno di un unico database.

* “*La società ha 6 fornitori che dopo 90 giorni di affitto ritirano la merce*”.

**Domanda**: è necessario gestire e predisporre un sistema di alert per la scadenza dei 90 giorni assieme a un sistema di reportistica tabellare? Bisogna inoltre gestire in qualche modo la mancata riconsegna dopo i 90 giorni?

**Risposta**: non è necessario predisporre un sistema di avvisi, una tabella in cui sono presenti i giorni mancanti alla riconsegna è sufficiente. Non serve la gestione della mancata riconsegna.

* “*Ogni settimana avviene uno scarico dei nuovi video e la restituzione dei video scaduti*”.

**Domanda**: chi si occuperà dello scarico fisico e virtuale dei video? Come avviene la procedura di scarico/carico?

**Risposta**: principalmente saranno gli addetti ad occuparsi dello scarico/carico dei video, sia fisici che virtuali. Ogni settimana ci sarà un trasportatore che porterà i nuovi video e caricherà quelli vecchi.

**Domanda**: è dunque necessario predisporre un componente di gestione degli ordini che si interfacci con i sistemi dei fornitori?

**Risposta**: no. Utilizziamo un applicativo per gli ordini che è il medesimo per tutti i fornitori. E’ sufficiente avere un report di tutti i video attualmente in magazzino.

* “*I punti vendita noleggiano i video a giorni interi e le tariffe sono decrescenti all’aumentare dei giorni di noleggio*”.

**Domanda**: le tariffe sono sempre le stesse o vanno gestite tramite un pannello di amministrazione?

**Risposta**: attualmente sono fisse e decise dal proprietario, ma in futuro potrebbe essere necessarie gestirle tramite un pannello amministrativo.

* “*Il responsabile del punto vendita può applicare gli sconti ad alcuni clienti fidelizzati, ma la politica degli sconti deve essere codificata*”.

**Domanda**: la funzione di creazione e modifica dei record relativi ai clienti sarà disponibile a tutti i tipi di utenza, eccetto la scelta della fidelizzazione?

**Risposta**: esatto, tutti i dipendenti potranno creare utenze, ma unicamente i responsabili o il proprietario potranno scegliere il tipo di fidelizzazione da applicare ai clienti.

* “*Ogni cliente dovrà firmare una liberatoria per la legge sulla privacy e si dovrà tenere traccia delle coordinate del documento cartaceo corrispondente*”

**Domanda**: può essere utile un meccanismo di upload del file cartaceo, così da averlo a disposizione per la stampa in qualsivoglia momento senza dover necessariamente andare a cercare il cartaceo in archivio?

**Risposta**: sì, può esserci molto utile.

* “*Prenotazione dei titoli in uscita da parte dei clienti*”.

**Domanda**: è necessario gestire le prenotazioni in modo tale che interagiscono con il conteggio dei noleggi effettuabili, o è sufficiente garantire un sistema di report che mostri quanti film sono stati prenotati?

**Risposta**: è sufficiente una reportistica.

* “*I clienti dovranno essere rintracciabili e deve essere tenuta traccia del cliente e dei video che ha in noleggio*”.

**Domanda**: la visualizzazione dei video in noleggio (attivi e passati) dovrà essere condivisa da tutti i punti vendita, o dovranno essere visualizzati i noleggi relativi al singolo punto vendita?

**Risposta**: ogni punto vendita dovrà visualizzare i propri noleggi attivi e passati relativi a quel cliente.

* I dipendenti dovranno noleggiare e prenotare i film richiesti dai clienti.

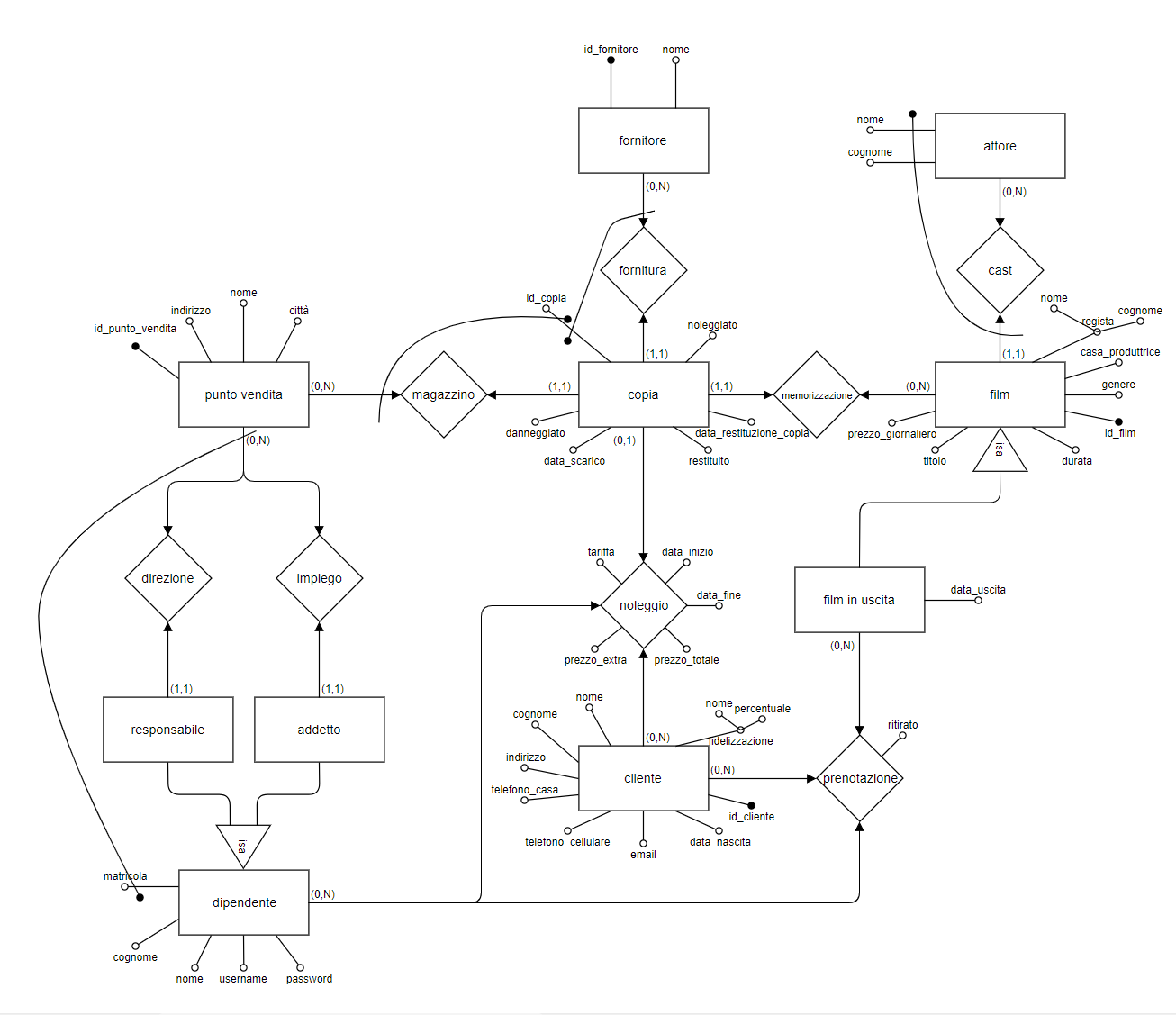
**Domanda**: come avviene il noleggio? Il cliente porta le copie in cassa o esse vengono recuperate dall’addetto?

**Risposta**: ogni cliente avrà una propria tessera che dovrà presentare all’addetto in cassa. Una volta scelti i film, il dipendente dovrà selezionare i film da noleggiare inserendo la matricola del cliente. Le copie verranno prese automaticamente dal magazzino tramite un macchinario apposito, che recupererà i film non danneggiati e con la data di restituzione al fornitore il più lontana possibile.

**Domanda**: per la gestione dell’eventualità che più clienti possano voler noleggiare l’ultima copia di un film nello stesso momento pensavamo di introdurre un meccanismo di prenotazione delle copie quando l’addetto procederà a premere il pulsante “Noleggia”: le copie verranno temporaneamente considerate come già noleggiate e quindi non disponibili alle query che si occuperanno di recuperare le copie noleggiabili. Predisporremo un evento che girerà secondo una determinata schedulazione: questo evento richiamerà una stored procedure che verificherà la data di “prenotazione” del noleggio, e se sarà superiore rispetto al range temporale prefissato, la libererà rendendola nuovamente disponibile. La stessa stored procedure verrà chiamata ogniqualvolta verrà fatto l’accesso alla pagina relativa ai noleggi. Può andare bene?

**Risposta**: certamente.

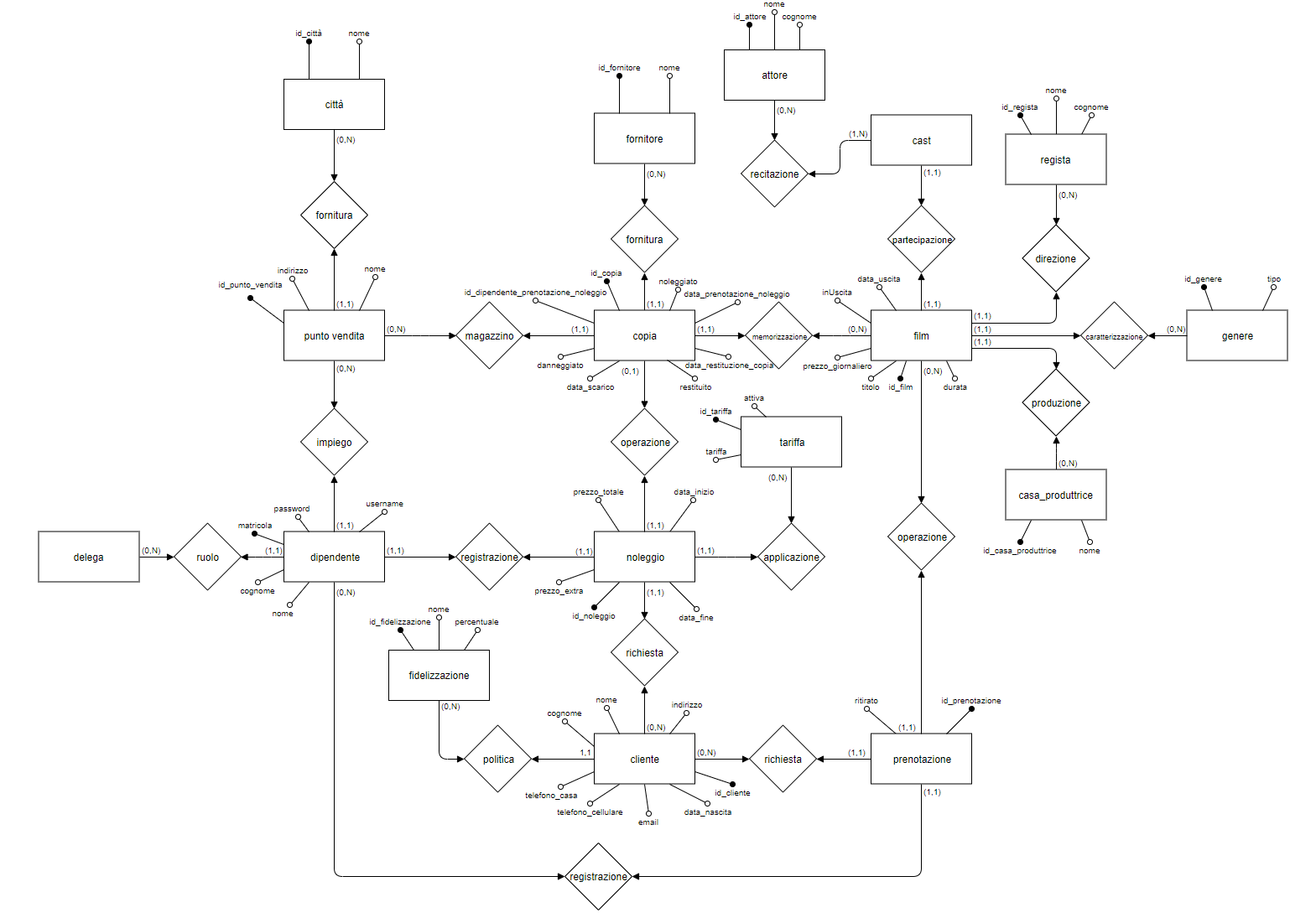
1. **Schema concettuale E-R**



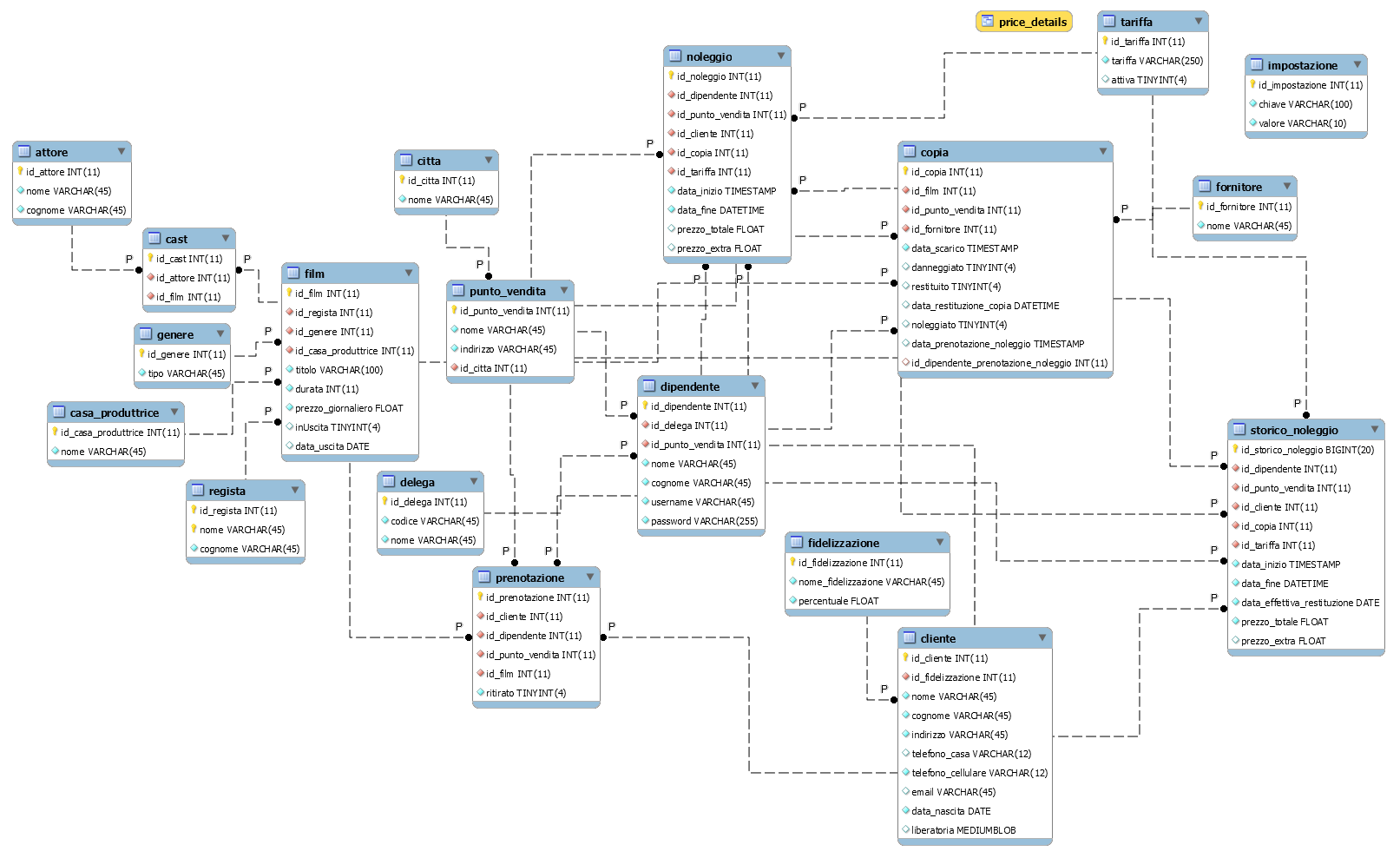
1. **Ristrutturazione schema E-R**

Al fine di poter tradurre lo schema concettuale in uno schema logico efficiente, è stato necessario effettuare una ristrutturazione dello schema E-R:

* Rimozione della gerarchia ISA “Responsabile”/”Addetto“ -> “Dipendente”, lasciando quindi come unica entità “Dipendente”. Per mantenere una corretta suddivisione dei ruoli viene introdotta l’entità “Delega”, con due attributi: “codice” e “nome”. Grazie all’introduzione di questa entità verrà garantito un buon livello di scalabilità sui ruoli dei dipendenti, laddove si riscontri la necessità di introdurne di nuovi.
* Rimozione della gerarchia ISA “Film in uscita” -> “Film”, lasciando come unica entità “Film”. Per mantenere la distinzione tra i film in uscita e quelli già usciti viene creato l’attributo booleano “inUscita”, mentre l’attibuto “data\_uscita” viene accorpato all’entità “Film”.
* Eliminazione dell’identificatore esterno “Punto vendita(id\_punto\_vendita)”/Dipendente(matricola)” e trasformazione dell’attributo “matricola” in chiave primaria dell’entità “Dipendente”. Questo perché il cliente ha richiesto di avere un database condiviso, ergo la matricola sarà sufficiente a garantire l’univocità delle istanze di “Dipendente”.
* Eliminazione degli identificatori esterni “Fornitore(id\_fornitore)”/Copia(id\_copia)” e “Punto vendita(id\_punto\_vendita)/Copia(id\_copia)” e conseguente trasformazione dell’attributo “id\_copia” in chiave primaria dell’entità “Copia”.
* Esplosione della relazione ternaria “Noleggio” con la trasformazione della relazione in entità per ottenere un diagramma più leggibile. Stessa trasformazione da relazione a entità per quanto riguarda “Prenotazione”.
* Creazione dell’entità “Fidelizzazione” per rimuovere l’attributo multiplo “fidelizzazione” dell’entità “Cliente”. La partecipazione per quanto riguarda “Cliente” alla nuova relazione “Politica” è obbligatoria, in quanto ogni cliente dovrà avere un certo tipo di sottoscrizione.
* Trasformazione degli attributi “genere”, “casa\_produttrice” e “regista” dell’entità “Film” in tre entità distinte, così da garantire eterogeneità ed evitare anomalie durante le operazioni di update. La nuova entità “Regista”, in particolare, avrà come attributi “id\_regista”, “nome” e “cognome”.
* Creazione dell’entità “Cast” con conseguente eliminazione dell’identificatore esterno “Film(id\_film)/Attore(nome, cognome)” e aggiunta della chiave “id\_attore” all’entità “Attore”, in modo tale da poter attingere da un bacino chiuso e facilmente modificabile di attori ed evitare anomalie di aggiornamento.
* Trasformazione dell’attributo “tariffa” dell’entità “Noleggio” in una nuova entità “Tariffa”. In questa maniera si potrà garantire una facile implementazione di una funzionalità di “cambio tariffa”: i proprietari dell’azienda potranno cambiare le tariffe (cioè le scontistiche da applicare ai noleggi su più giorni consecutivi) tramite un pannello amministrativo, creandone di nuove o attivando tariffe già create in passato. La creazione dell’entità “Tariffa” eviterà, tra le altre cose, anomalie e incongruenze sulle istanze dei noleggi già esistenti (se un cliente noleggia una copia di un film con una determinata tariffa, essa deve rimanere tale e quale anche in caso di cambiamento delle tariffe da parte dei proprietari). Questa entità possiederà gli attributi “id\_tariffa”, “tariffa” (contenente un JSON che rappresenta un oggetto di tipo dictionary avente come chiave il numero di giorni e come valore la percentuale di scontistica applicata a quel numero di giorni) e “attiva” (che identificherà quale tariffa è attiva in quel momento).
* Rimozione dell’attributo “città” dall’entità “Punto vendita” e conseguente creazione dell’entità “Città”. Questo viene fatto sempre per garantire eterogeneità di dati e per evitare anomalie di aggiornamento.
* Introduzione degli attributi “data\_prenotazione\_noleggio” e “id\_dipendente\_prenotazione\_noleggio” all’entità “Copia”. Questo viene fatto per garantire il corretto funzionamento della funzionalità di prenotazione dei noleggi.

Si ottiene, dunque, il seguente schema E-R ristrutturato:

1. **Schema logico**

A questo punto

1. **Configurazioni server**

Per l’ambiente di sviluppo è stato utilizzato IIS, mentre per quello di produzione, essendo hostato su un Raspberry, è stato utilizzato Apache. Non è necessario applicare configurazioni particolari.