GESTIONE CINEMA MULTISALA

Il cliente ci ha commissionato la realizzazione di un portale web custom per la gestione dei cinema multisala su diverse location di prossima apertura nel territorio italiano.

Ogni location avrà almeno 3 sale con i seguenti orari di apertura:

* lun-ven: 12:00 - 23:00
* sab-dom: 11:00 - 1:00

Il portale web dovrà permettere la gestione di 3 tipologie di utenze:

* ADMIN -> amministratore della location
* EMPLOYEE -> personale del cinema con diverse qualifiche (esempio: responsabile sala, bigliettaio, venditore popcorn, barista)
* CUSTOMER -> cliente

Tb:

UserTypes

* userTypeId
* userType

Di seguito le attività che possono svolgere le varie utenze previa autenticazione:

ADMIN:

* [auth] Deve creare/modificare le varie qualifiche (responsabile sala, bigliettaio) per gli EMPLOYEE (CRUD) 🡪tb: JobEmployeeQualification
  + Le qualifiche devono essere univoche (non ci possono essere 2 qualifiche “Responsabile di sala” per es.) (fatto sulla “ShortDescr”) (OK)
  + Non può essere né eliminata né modificata una qualifica se ha anche solo un EMPLOYEE associato (fatto) (OK)
  + NON può essere associata una qualifica ad un ADMIN o CUSTOMER
    - Questa mi viene gratis perché ho una tabella dedicata per gli emloyee (OK)
* [auth] Deve censire nuovi EMPLOYEE (CRUD) (fatto) tb: UserEmployee
  + E poter assegnare loro una qualifica
    - La qualifica di un EMPLOYEE può essere cambiata:
      * Per i Responsabili di Sala, quando questi non sono assegnati a nessuna sala (fatto)(OK)
      * Per i Bigliettai, se ce ne sono almeno 2 attivi (in modo da rimanere “coperti”) (fatto, se c’è il valore la colonna “minimumRequired” controllo che ci siano almeno quelli richiesti) (OK)
    - Per l’eliminazione di un EMPLOYEE valgono gli stessi vincoli citati qui sopra per il cambio qualifica (fatto) (OK)
* [auth] Creare/gestire le sale (numero posti, numero posti vip (che avranno un biglietto più costoso, in percentuale rispetto ai posti standard) Tb: CinemaRoom
  + Una sala NON può essere eliminata se c’è un film programmato in quella sala (fatto) (OK)
    - Se viene eliminata, il Responsabile di Sala NON viene eliminato ma rimane “libero” (OK)
  + I posti, posti vip ecc, possono essere modificati solo quando non ci sono in programma film approvati in quella sala (fatto)(OK)
    - Il numero e il nome (facoltativo) devono essere unici (fatto, unique su roomName)
    - Il numero può essere cambiato ma non può essere < 1 (fatto, ho usato un constraint che controlla il valore di maxVipSeat e maxStdSeat)
  + Per creare una sala serve un Responsabile di Sala da associare (uno solo per sala) (fatto)(OK)
    - Non può esserci lo stesso per più sale (fatto)
* [auth] Deve inserire nuovi film (data orario, in quale sala viene proiettato, durata, genere, trama, cover\_img, attori principali (nome e cognome e basta direi), regista, anno di produzione) (fatto) (ok)
  + Un film non può essere proiettato in più di due sale contemporaneamente (in due potenzialmente sì) (fatto) (OK)
  + La data di inizio film non può essere minore di oggi, l’orario deve essere maggiore di adesso
    - Attenzione agli orari del cinema (vedi in cima al word) (fatto) (OK)
  + Tra un film e l’altro, nella stessa sala, ci deve essere un distacco di 10m (per simulare la pulizia della sala o comunque il tempo di far uscire le persone e far entrare le altre) (fatto) (OK)
  + Può essere modificato finché non viene approvato dall’ADMIN (questo punto non ha senso, in quanto è l’admin a fare le update, fatto) (OK l’ho implementato cmq)
  + Non ci possono essere due film nella stessa sala allo stesso orario! (fatto) (OK)
* [auth] Deve approvare i film inseriti in modo che risultano programmati (fatto)(OK)
  + Può approvarli solo se rispecchia i requisiti scritti qui sopra (fatto) (OK, nel mio caso non ha senso ci sono già i controlli vari)
* [auth] Può visualizzare lo stato delle sale in diretta: il film in una certa sala è in corso oppure non è ancora iniziato/è finito quello in corso => tramite SignalR ho la situazione in diretta (la sala diventa verde quando il film non è ancora iniziato e rossa quando il film in quella sala è in corso) (OK, avviene la notifica quando lo stato di un film cambia)
* [auth] Visualizzare le recensioni fatte dai CUSTOMER (non in diretta, semplicemente un’API che tira su la lista delle recensioni fatte su un film, la lista deve essere paginata e con filtri (numero stelle e/o criterio) (fatto: filtro solo su nome del file completo (no like))
* FACOLTATIVO: [auth] Proporre dei report sulle vendite dei biglietti dei vari film fornendo delle statistiche del tipo: range di orari col maggior flusso di clienti, generi preferiti, biglietti venduti/cancellati etc…)

**EMPLOYEE:**

Responsabile di Sala:

* [auth] Deve visualizzare in diretta (SignalR) lo stato dei posti della sala in cui si trova (es. 100 posti totali, di cui 20 vip -> 50 verdi: liberi, 50 rossi: prenotati)
  + Non può vedere lo stato delle altre sale (fatto)

**Bigliettaio:**

* [Deve essere notificato (SignalR) quando un ADMIN approva un film e per questo devono essere creati i biglietti] (fatto)
* [auth] Deve generare i biglietti di un film che deve ancora essere proiettato
  + Non è possibile generare biglietti per film già in corso o terminati (fatto)
  + Il numero dei biglietti viene calcolato automaticamente in base ai posti della sala (attenzione ai posti vip) (fatto)
  + Il prezzo viene inserito dal bigliettaio; per i posti vip il prezzo sarà maggiore di tot % (il bigliettaio inserisce una percentuale e viene calcolato il prezzo VIP?) (fatto)

CUSTOMER:

* [no-auth] Può visionare la lista (paginata) dei vari film schedulati/approvati (con filtri: giorno, genere) (fatto)
* [no-auth] Può visualizzare il dettaglio di un film
  + Si vedono le info del film e della sala in cui viene proiettato
  + Si vedono anche i posti occupati/disponibili (fatto)
* [auth] Può acquistare un biglietto per un certo film che viene trasmesso in una sala ad un orario prefissato
  + Nel caso in cui il film sia 18+, verificare la maggiore età dell’utente e nel caso bloccarlo
  + Se il film è già iniziato non si possono acquistare più i biglietti (anche se ancora disponibili)
  + Può prenotare anche per altre persone (max 4) (fatto)
* [auth] Può cancellare la prenotazione che ha fatto per un film che deve ancora iniziare (fatto)
* [auth] Può recensire il film appena visto mettendo un voto da 1 a 5, i criteri in base ai quali è stata fatta la recensione (es. trama del film, attori, ambientazione, ecc.) e un eventuale commento sul film (fatto)
* [auth] Nella sua area personale può vedere la lista dei film che ha visto (con dettaglio film) e in che data (fatto)

WORKER

* Worker che annulla i film approvati se non vengono generati i biglietti prima dell’inizio del film. Viene tolta la spunta dell'approvazione e l'ADMIN se lo ritroverà da approvare dopo averlo schedulato nuovamente (fatto)
* Worker che ogni tot minuti (es. 5m) “libera” il Responsabile di Sala quando una sala non ha più film programmati per quel giorno (fatto)

NOTA: ovviamente non si faranno film che durano ore ecc., useremo i minuti e i secondi

Risultato atteso:

* ~~Analisi e stima dei task~~
* Disegno architetturale
* Eventuali pattern scelti con motivazione
* ~~Collezione Postman con le API prodotte~~
* Solution con tutto il codice sorgente comprensivo della parte di SignalR (verrà fornito un client per la comunicazione col backend)

Requisiti:

* Database relazionale: SQL Server (o eventualmente a vostra scelta)
* Accesso al dato tramite ORM ed applicazione di eventuali pattern
* Gestione di un layer con le logiche di business organizzate secondo i pattern che meglio si adeguano
* Esposizione dei dati tramite API con relativa autenticazione/autorizzazione e permessi (esempio: se un CUSTOMER chiama le API a cui non dovrebbe accedere deve avere un 401)
* Aggiornamento dello stato delle sale e dei posti delle varie sale (liberi/occupati) tramite SignalR

Requisiti tecnici:

* .NET 6
* EF o altro ORM
* DI (DbContext e altri servizi)
* Middleware (ad esempio: commit/rollback con EF)
* Hosted/Background Services schedulabile ogni tot minuti
* Utilizzo di patterns

NOTA:

* tutto quello che non è espressamente richiesto in questo documento può essere gestito liberamente da voi giustificando le scelte con delle valide motivazioni.

Consegna:

* L’elaborato finale deve essere una Solution Visual Studio 2022
* È necessario il collegamento ad un DataBase reale attraverso l’ORM che reputate più adatto ed applicando i pattern che meglio agevolano la scrittura/lettura/estendibilità del codice.
* Per la consegna è necessario inviare una mail ai docenti con:
  + ❑ Link al repository GIT con condivisione dell’accesso
  + ❑ File ZIP allegato alla mail (senza pacchetti NuGet e/o node\_modules) ATTENZIONE: le mail con allegati troppo pesanti non vengono recapitate
  + ❑ Link a file ZIP salvato sul Cloud con eventuali credenziali (esempio: <https://wetransfer.com/>)
  + È gradito un breve README per il setup della Solution
  + Non sono accettate immagini Docker (se volete potete fare anche quelle, ma non solo quelle)

La data ultima per sottoporre l’elaborato è:

domenica 21 maggio alle ore 23.59 (fa fede la data di ricezione mail)