DOCUMENTAZIONE

FUNZIONALITA'

Le funzionalità messe a disposizione dall'applicazione si suddividono in tre principali gruppi:

1) Autenticazione/registrazione
2) Ricerca film e acquisto biglietti
3) Area gestori

L'accesso a queste funzionalità non è libero ma è regolato da una politica di controllo degli accessi. Questo controllo degli accessi fa riferimento a tre categorie di ruoli per gli utenti: utenti anonimi (non autenticati) ; clienti ; gestori. La politica di controllo degli accessi ha le seguenti linee guida:

- Gli utenti anonimi possono autenticarsi/registrarsi e possono ricercare film (ma non possono acquistare biglietti)
- I clienti possono ricercare film e acquistare biglietti
- I gestori possono accedere all'area gestori

1) Autenticazione/registrazione

Dalla home page un utente anonimo può effettuare il login (autenticazione), tramite cui accede a un profilo già creato: in questo modo si autentica come cliente o come gestore. Per autenticarsi l'utente deve inserire l'email del profilo con cui vuole autenticarsi e la password relativa: se entrambi questi dati sono corretti e corrispondono allora l'autenticazione è completata correttamente. A seconda che il profilo con

cui l’utente anonimo accede è un profilo cliente o un profilo gestore l’utente anonimo si autentica come cliente o come gestore.

In alternativa alla login l’utente anonimo può creare un nuovo profilo (registrazione). Per creare un nuovo profilo l’utente anonimo deve inserire tutti i dati necessari: email, nome utente, password, provincia, anno nascita, sesso. Se tutti i dati sono corretti è creato un nuovo profilo cliente con questi dati (se per esempio l’email scelta è già usata l’operazione è fatta fallire). E’ bene dunque precisare che può essere creato solo un cliente tramite questa procedura. La creazione di nuovi gestori può essere effettuata solo dall’area gestori. (Si veda sezione sull’area gestore)

Sia quando l’operazione di login che quando l’operazione di registrazione sono effettuate correttamente l’utente da anonimo diventa autenticato: ora la home page è la sua area riservata. Quest’area riservata è diversa a seconda che l’utente sia un cliente o un gestore.

-L’area riservata del cliente, oltre al link per effettuare il logout, mostra tutti i posti di proiezioni future acquistati da quel cliente. Questi sono ordinati per data della proiezione e per posto. (Ordine lessicografico rispetto a questi due attributi)

- L’area riservata del gestore, oltre al link per effettuare il logout, mostra tutte le funzionalità che saranno descritte nella sezione sull’area gestori.

2)Ricerca film e acquisto posto

Alla pagina principale per la ricerca di film si può accedere in due modi:

-Dalla home page dell’utente anonimo

-Dalla home page del cliente

La pagina principale della ricerca mostra innanzitutto i film più gettonati del momento. Ovvero mostra i film in programmazione (ovvero film che hanno proiezioni future) ordinati in modo decrescente rispetto al numero

complessivo di biglietti venduti tramite queste proiezioni. Sono mostrati al massimo i primi 5 film più gettonati. (Se nessun film nel database ha proiezioni future o ha biglietti venduti allora non è mostrato alcun film). Di questi film è mostrato solo il titolo. Cliccando su uno di questi film vengono mostrate le proiezioni future del film in questione, ordinate per data.

Successivamente la pagina principale offre altri 4 modi diversi di ricerca di film: mostra tutti i film in programmazione; ricerca per giorno; ricerca per titolo; ricerca per genere.

-Nella pagina dove sono mostrati tutti i film in programmazione (o meglio, i loro titoli) è possibile cliccare su ognuno di questi film per visualizzare tutte le sue proiezioni future, ordinate per data.

-Nella pagina dove si ricerca per giorno l'utente può specificare un giorno e sono mostrate tutte le proiezioni in quel giorno, se ce ne sono. La selezione di una data già passata dà risultato vuoto.

-Nella pagina dove si ricerca per titolo l'utente inserisce una stringa di testo e sono mostrati tutti i film(o meglio, i loro titoli) con titolo contenti questa stringa. In altre parole sono mostrati i film per cui la stringa digitata è sottostringa del loro titolo, case unsensitive. La ricerca può ovviamente portare a risultato vuoto. Cliccando su un film sono mostrate tutte le sue proiezioni future, ordinate per data. (Sempre che quel film abbia proiezioni future)

-Nella pagina dove si ricerca per genere sono mostrati tutti i generi attualmente presenti nel database (C'è almeno un film con tale genere). Cliccando su un genere sono mostrati tutti i film(o meglio, i loro titoli) con quel genere. A questo punto cliccando su un film sono mostrate tutte le sue proiezioni future, ordinate per data. (Sempre che quel film abbia proiezioni future)

Da tutti questi 5 possibili modi di ricercare film si arriva ad uno stesso punto: una pagina dove sono mostrate delle proiezioni future. Per ogni proiezione è mostrato il titolo e la durata del film relativo, l'orario, la sala e il prezzo di un posto. Ora cliccando su una di queste proiezioni si procede alla visualizzazione dei posti liberi (se ce ne sono) e al possibile acquisto di un posto. Attenzione però: se l'utente che ha cliccato su una di queste proiezioni è anonimo (non è un cliente) allora tale utente è invitato ad autenticarsi prima di poter procedere alla visualizzazione dei posti liberi.

Dalla pagina di visualizzazione dei posti liberi sono mostrate anche tutte le informazioni citate sopra sulla proiezione in questione (titolo,durata,orario,sala,prezzo). A questo punto il cliente può comprare dei posti. E' possibile acquistare solo un posto alla volta. Ogni volta che è acquistato un posto è confermato o meno il successo dell'operazione. Il cliente può ora tornare alla pagina precedente per acquistare altri posti per la stessa proiezione. Oppure il cliente può anche tornare alla sua home page, se vuole verificare che il posto appena acquistato è effettivamente tra la lista dei posti da lui presi.

Tutte le pagine per la ricerca di film e per l'acquisto di posti non possono essere accedute dai gestori. Inoltre le pagine per l'acquisto dei posti non possono essere accedute dagli utenti anonimi.

3)Area gestori

Come descritto sopra, dopo avere effettuato l'accesso con le credenziali di un gestore precedentemente creato si può accedere a tutte le funzionalità del gestore.

Le funzionalità del gestore si suddividono in 3 sezioni:

-Amministrazione del DB principale nella quale si basa la webapp;

-Visualizzazione delle statistiche relative alle proiezioni che sono presenti nel DB;

-Aggiunta a sua volta di un altro account gestore, che avrà le sue stesse autorizzazioni.

Amministrazione del Database

Breve sintesi

Il gestore può, una volta entrato nella prima sezione, effettuare operazioni aggiungendo film, aggiungendo o gestendo sale (viene permessa di impostare la disponibilità delle sale, dato che una volta create sono impostate “non disponibili” di default). Inoltre il gestore può agire sulle proiezioni aggiungendone di nuove oppure eliminando quelle che sono presenti nelle sale disponibili.

Aggiungi film: viene inserito nel db un film dato: titolo, anno, regista, generi e la sua durata. All’aggiunta del film corrisponde anche l’aggiunta nel DB dei relativi generi associati al film, aggiunti nella tabella “GeneriFilm”, memorizzandone il genere e il film alla quale è associato.

Aggiungi Sala: viene permessa l’aggiunta di una sala del cinema specificandone il numero di posti e di file. Importante ricordare che non ci devono essere posti in eccesso in sala (quindi il resto della divisione tra i posti e il numero di file dev’essere 0) e che quando una sala è creata è impostata non disponibile di default.

Gestisci sale: viene mostrata una tabella con tutte le sale presenti nel cinema, con a fianco la voce “Disponibile” che indica se la sala è disponibile oppure se non lo è. Nella colonna “Imposta disponibilità”, invece, si può scegliere se rendere la sala disponibile oppure no. Se la spunta sarà blu allora la sala sarà disponibile altrimenti verrà impostata non disponibile. Tutte le relative proiezioni e biglietti delle sale non disponibili verranno eliminate.

Aggiungi proiezione: viene aggiunta una proiezione inserendo la sala, il film da proiettare, la relativa data e il costo del biglietto per la proiezione. La proiezione può essere inserita nel rispetto del vincolo nella quale

nessuna proiezione può essere messa a meno di 3 ore dall'inizio di un'altra proiezione nella stessa sala.

Elimina proiezione futura: vengono mostrate tutte le proiezioni future delle sale che sono disponibili, con la relativa data, prezzo e titolo del film.

Statistiche dei vari film

Breve sintesi

In questa sezione il gestore ha la possibilità di raccogliere i dati relativi alle proiezioni e i biglietti venduti dei vari film effettuando diverse modalità di ricerca, per effettuare un'analisi statistica quando necessaria.

Le statistiche effettuate possono essere di 3 tipi:

-per genere;

-per titolo;

-per provincia.

Statistiche per genere

Vengono visualizzati in una selectbox tutti i generi dei film che sono stati memorizzati all'interno del database e si ha la possibilità, selezionandone uno, di vedere tutti i film relativi a quel genere, il numero di proiezioni associate a quei film, il numero di biglietti venduti e il guadagno totale da quei biglietti.

Statistiche per titolo

Il risultato ottenuto è il medesimo rispetto a quello dato dalle statistiche per genere solo che in questo caso possiamo inserire il titolo o parte del titolo di un film.

Statistiche per provincia

In questa sezione invece possiamo andare a visualizzare i biglietti acquistati e di conseguenza il guadagno totale di tutti gli utenti che vengono dalla provincia che selezioneremo dalla selectbox.

Aggiunta di un nuovo gestore

In questa sezione possiamo aggiungere un nuovo gestore inserendo email, nome utente, password, provincia, sesso e l'anno assunzione. Come nella registrazione di un utente questa operazione va a buon fine solamente nel caso in cui non ci sia già un utente registrato (anche non gestore) con la stessa email inserita, dato che dev'essere univoca.

Tutte queste pagine per la gestione del cinema non possono essere accedute nè dagli utenti nè dai clienti .

PROGETTAZIONE CONCETTUALE E LOGICA

Progettazione concettuale

Nel modello concettuale sono presenti le seguenti collezioni

1)Film

La classe Film rappresenta appunto i film presenti nel cinema. Ha i seguenti attributi:

- titolo : String <<NOT NULL>> Titolo del film

- anno : Integer <<NOT NULL>> Anno in cui il film è stato prodotto
- regista : String <<NOT NULL>> Regista che ha prodotto il film
- minuti : Integer <<NOT NULL>> Durata del film in minuti
- generi : SEQ String

2)Proiezioni

La classe Proiezioni rappresenta tutte le proiezioni presenti nel cinema, sia passate che future. Ha i seguenti attributi:

- orario : Datetime <<NOT NULL>> Giorno e ora in cui è proiettata
- prezzo : Float <<NOT NULL>> Prezzo di un posto

3)Sale

La classe Sale rappresenta tutte le sale presenti nel cinema, sia attualmente disponibili che attualmente non disponibili. (Con sala disponibile si intende sala che può essere attualmente usata per mostrare film. Una sala non disponibile è, per esempio, una sala che sta subendo lavori di manutenzione). Ha i seguenti attributi:

- idSala : Integer <<PRIMARY KEY>> E' il numero identificativo della sala
- numPosti : Integer <<NOT NULL>> Numero di posti della sala

- numFile : Integer <<NOT NULL>> Numero delle file della sala
- disponibile : Boolean <<NOT NULL>> Indica se tale sala è disponibile o no

4)Biglietti

La classe Biglietti rappresenta tutti i biglietti acquistati nel cinema, sia passati (relativi a proiezioni passate) che future. Ogni biglietto consiste in un solo posto acquistato. Ha dunque un solo attributo

- posto : Integer <<NOT NULL>>

5)Utenti

La classe Utenti rappresenta tutti gli utenti del cinema. Ha i seguenti attributi:

- email : String <<PRIMARY KEY>> email dell'utente
- nomeUtente : String <<NOT NULL>> nome utente
- pwd : String <<NOT NULL>> password
- annoNascita : String <<NOT NULL>> anno di nascita
- sesso : (M,F) <<NOT NULL>> sesso
- provincia : String <<NOT NULL>> provincia

La classe Utenti ha due sottoclassi, che rappresentano le due tipologie di utenti del cinema: i clienti e i gestori.

- La classe Gestori rappresenta appunto gli utenti gestori. Questa classe presenta un unico attributo specifico

  annoAssunzione : String <<NOT NULL>>

- La classe Clienti rappresenta appunto gli utenti clienti. Questa classe non ha attributi specifici

Le classi Gestori e Clienti sono una partizione della superclasse Utenti (sono classi disgiunte e la loro unione copre tutta la classe Utenti)

Tra queste classi ci sono varie associazioni

1) Associazione Film-Proiezioni

Questa associazione, che chiamiamo “haProiettato”, è multivalore e parziale da Film a Proiezioni ed è univoca e totale da Proiezioni a Film.

Ciò significa che ogni film può avere associate più proiezioni o anche nessuna e che ogni proiezione ha associato un unico film.
Ovviamente un film può avere associate sia proiezioni passate che future

2) Associazione Proiezioni-Sale

Questa associazione, che chiamiamo “èCollocata”, è univoca e totale da Proiezioni a Sale ed è multivalore e parziale da Sale a Proiezioni.

Ciò significa che ogni proiezione ha associata un’unica sala e che ogni sala può avere associate più proiezioni o anche nessuna.
Come detto prima, una proiezione futura può avere associata solo una sala disponibile. Mentre una proiezione passata può avere associata anche una sala non disponibile.

3) Associazione Proiezioni-Biglietti

Questa associazione, che chiamiamo “ha”, è multivalore e parziale da Proiezioni a Biglietti ed è univoca e totale da Biglietti a Proiezioni.

Ciò significa che ogni proiezione può avere associati più biglietti o anche nessuno e che ogni biglietto ha associata un’unica proiezione.
Sia le proiezioni passate che le proiezioni future possono avere posti associati. Ovviamente le proiezioni passate non potranno avere nuovi posti acquistati.

4) Associazione Clienti-Biglietti

Questa associazione, che chiamiamo “haComprato”, è multivalore e parziale da Clienti a Biglietti ed è univoca e totale da Biglietti a Clienti.

Ciò significa che ogni cliente può avere associati più biglietti o anche nessuno e che ogni biglietto ha associata un unico cliente.

Per concludere l’illustrazione del modello concettuale mostriamo tre importanti invarianti di questo schema:

- Invariante nella classe Proiezioni.
  Le proiezioni future ,ovvero con orario maggiore dell’istante attuale, devono avere sala disponibile.
  Nel codice del progetto ci assicuriamo che ciò sia sempre rispettato (quando si crea una proiezione si controlla che la sala relativa sia disponibile e quando si mette una sala non disponibile si eliminano le proiezioni future)

- Invariante nella classe Proiezioni.
  Non possono esserci due biglietti di una stessa proiezione con uguale posto.
  Tale invariante è fatto rispettare tramite i vincoli di primary key che saranno messi sulla tabella Biglietti nel modello relazionale

- Invariante nella classe Biglietti.
  Ogni biglietto ha posto maggiore o uguale a 0 e minore dell'attributo numPosti della sala della proiezione relativa al biglietto
  Nel codice del progetto ci assicuriamo che ciò sia sempre rispettato (ogni volta che si acquista un posto si controlla che ciò sia rispettato)

Progettazione logica

Ogni classe prima descritta diventa una tabella. Vengono ora mostrate le modifiche/aggiunte

1)Nella tabella Film è stato aggiunto un attributo chiave primaria

idFilm : Integer <<PRIMARY KEY>>

Sono inoltre aggiunti dei vincoli di check su due attributi

anno CHECK(anno>=1970)
minuti CHECK(minuti>=1)

E' in più tolto l'attributo generi (vedi prossimo punto)

2)E' aggiunta la tabella GeneriFilm per rappresentare l'attributo multivalore "generi" della classe Film. Questa nuova tabella ha due attributi

- genere : String

- film : Integer <<FOREIGNKEY(Film)>>
  Tale chiave esterna è definita ON DELETE CASCADE e ON UPDATE CASCADE (Eventuali modifiche fatte alla tabella Film sono riportate alle ennuple di GeneriFilm che puntano a tali

ennuple modificate. In realtà nella nostra webapp non diamo la possibilià di eliminare o modifcare film, abbiamo messo questi vincoli per completezza)

Gli attributi genere e film insieme sono la primary key della tabella GeneriFilm

3) Nella tabella Proiezioni è stato aggiunto un attributo chiave primaria

idProiezione : Integer <<PRIMARY KEY>>

Sono inoltre stati aggiunti i seguenti attributi come chiavi esterne

film : Integer <<FOREIGNKEY(Film)>> <<NOT NULL>>
Non è specificato nulla sull'ON DELETE o sull'ON CASCADE per questa chiave esterna. Dunque eventuali modifiche sulle tabelle puntate sono semplicemente rifiutate
sala : Integer <<FOREIGNKEY(Sale)>> <<NOT NULL>>
Non è specificato nulla sull'ON DELETE o sull'ON CASCADE per questa chiave esterna. Dunque eventuali modifiche sulle tabelle puntate sono semplicemente rifiutate

Sono inoltre aggiunti dei vincoli di check su due attributi

orario	CHECK(orario>=1970-01-01)
prezzo	CHECK(prezzo>=0.0)

4) Nella tabella Sale è stato aggiunto un attributo chiave primaria

idSala : Integer <<PRIMARY KEY>>

Sono inoltre aggiunti dei vincoli di check. Uno sull'attributo numPosti, uno sull'attributo numFile e uno riguardante entrambi questi attributi (check su tuple)

numPosti	CHECK(numPosti>=10)
numFile	CHECK(numFile>=1)

CHECK(numPosti%numFile=0)

5) Nella tabella Biglietti sono stati aggiunti i seguenti attributi come chiavi esterne

proiezione : Integer <<FOREIGNKEY(Proiezioni)>>
Tale chiave esterna è definita ON DELETE CASCADE e ON UPDATE CASCADE (Eventuali modifiche fatte alla tabella puntata sono riportate alle ennuple di GeneriFilm che puntano a tali ennuple modificate)
cliente : String <<FOREIGNKEY(Utenti)>> <<NOT NULL>>
Tale chiave esterna è definita ON DELETE CASCADE e ON UPDATE CASCADE (Eventuali modifiche fatte alla tabella puntata sono riportate alle ennuple di GeneriFilm che puntano a tali ennuple modificate)

Gli attributi posto e proiezione insieme sono la primary key della tabella Biglietti (Ciò di fatto fa rispettare l'invariante che una proiezione non può avere due biglietti con stesso posto)

E' stato inoltre aggiunto il seguente vincolo di check

posto CHECK(posto>=0)

6)La gerarchia di classi Utenti-Gestori-Clienti è stata realizzata tramite tabella unica. Unica tabella Utenti, con attributo discriminatore "gestore" .

gestore : Boolean <<NOT NULL>>

Vale True se l'utente è un gestore, False altrimenti.

Essendo tutto in un'unica tabella, l'attributo annoAssunzione può valere NULL (se l'utente è un cliente tale attributo vale NULL)

Il DBMS utilizzato è postgreSQL, il quale supporta (con una sintassi simile a quella appena usata) tutte le caratteristiche specificate. Inoltre tutte le primary key intere sono di default soggette al comodo auto increment: se nell'inserire un'ennupla non viene specificato il valore per

l’attributo primary key, questo assume come valore il valore successivo all’ultimo valore preso.

QUERY INTERESSANTI

Parte login/registrazione/area riservata

-Query per trovare tutti i posti relativi a proiezioni future acquistati dal cliente in questione. Oltre ai posti, è selezionato anche l'orario,il titolo, la durata e la sala della proiezione relativa. Le ennuple sono ordinate per orario della proiezione e per posto. Tale query è effettuata nella funzione posti_cliente_query(email) (in input c’è la mail del cliente)

```
select([biglietti.c.posto, proiezioni.c.orario, film.c.titolo,
proiezioni.c.sala, film.c.minuti ]).where( and_(
biglietti.c.cliente==bindparam("email"),
proiezioni.c.idProiezione==biglietti.c.proiezione,
film.c.idFilm==proiezioni.c.film, proiezioni.c.orario>datetime.now()
)).order_by( proiezioni.c.orario, biglietti.c.posto)
```

```
SELECT Biglietti.posto, Proiezioni.orario, Proiezioni.sala,
Film.titolo, Film.minuti
FROM Film, Proiezioni, Biglietti
WHERE (Biglietti.cliente=:email AND Film.idFilm=Proiezioni.film
AND Proiezioni.idProiezione=Biglietti.proiezione  AND
Proiezioni.orario>NOW() )
ORDER BY  Proiezioni.orario, Biglietti.posto ;
```

Parte ricerca film e acquisto posti

-Query per trovare i film gettonati del momento, ovvero i film con proiezioni future ordinati in modo decrescente rispetto al numero complessivo di biglietti venduti tramite queste proiezioni. Tale query è effettuata nella funzione filmGettonati_query().

```
select([film]).where(and_( proiezioni.c.film==film.c.idFilm,
proiezioni.c.orario>datetime.now(),
biglietti.c.proiezione==proiezioni.c.idProiezione )).group_by(
film.c.idFilm ).order_by( desc( func.count() ))
```

```
SELECT Film.idFilm, Film.titolo, Film.anno, Film.regista,
Film.minuti
FROM Film, Proiezioni, Biglietti
WHERE (Film.idFilm=Proiezioni.film AND
Proiezioni.idProiezione=Biglietti.proiezione  AND
Proiezioni.orario>NOW() )
GROUP BY Film.idFilm
ORDER BY  COUNT(*)  DESC ;
```

-Query per trovare le proiezioni future di un dato film. Per correttezza si controlla anche che la sala di tali proiezioni sia disponibile. Inoltre le proiezioni sono ordinate per orario crescente. Questa query è usata nella funzione proiezioni_film_query(id_film).

```
select([proiezioni, film.c.titolo, film.c.minuti ]).where(and_(
proiezioni.c.film==film.c.idFilm, film.c.idFilm==bindparam('id'),
proiezioni.c.orario>datetime.now(),
sale.c.idSala==proiezioni.c.sala, sale.c.disponibile )).order_by(
proiezioni.c.orario )
```

```
SELECT Proiezioni.idProiezione, Proiezioni.orario,
Proiezioni.prezzo, Proiezioni.film, Proiezioni.sala, Film.titolo,
Film.minuti
FROM Film, Proiezioni, Sale
```

WHERE (Film.idFilm=Proiezioni.film AND Film.idFilm=:id AND Proiezioni.orario>NOW() AND Sale.idSala=Proiezioni.sala AND Sale.disponibile)
ORDER BY Proiezioni.orario ;

Parte analisi delle statistiche

-Query per trovare il guadagno dai biglietti venduti relativi a una determinata proiezione di un certo film, usata nella funzione statisticheTitolo_query, in database.py

select([func.sum(proiezioni.c.prezzo).label("guadagno")]).where(and_(film.c.idFilm==proiezioni.c.film,proiezioni.c.idProiezione==biglietti.c.proiezione,film.c.titolo.contains(bindparam('titolo'))))

SELECT SUM(Proiezioni.prezzo) AS guadagno

FROM Film, Proiezioni, Biglietti

WHERE Film.idFilm==Proiezioni.idFilm AND Proiezioni.idProiezione==Biglietti.proiezione AND

Film.titolo LIKE '%' 'titolo' '%'

-Query che conta i biglietti venduti a tutti gli utenti che risiedono in una certa provincia.

select([func.count().label("nbiglietti")]).where(and_(film.c.idFilm==proiezioni.c.film,proiezioni.c.idProiezione==biglietti.c.proiezione,biglietti.c.cliente==utenti.c.email,utenti.c.provincia==bindparam('provincia')))

SELECT COUNT(*) AS biglietti

FROM Film, Proiezioni, Biglietti, Utenti

WHERE Film.idFilm==Proiezioni.film AND Proiezioni.idProiezioni==Biglietti.proiezione AND Biglietti.cliente==Utenti.email AND Utenti.provincia=='provincia'

Parte gestione delle proiezioni

-Query per determinare se esiste una proiezione di un film nell'orario in cui voglio mettere il mio film. Un film può essere messo al minimo alla distanza di 3 ore da uno iniziato. Usata nell'aggiunta delle proiezioni in "Aggiungi Proiezione".

select([film.c.titolo]).where(and_(film.c.idFilm==proiezioni.c.film,proiezioni.c.sala==sale.c.idSala,sale.c.idSala==bindparam('sala'),proiezioni.c.orario<orario,orario<(proiezioni.c.orario+timedelta(hours=3))))

SELECT Film.titolo

FROM Film, Sale, Proiezioni

WHERE Film.idFilm==Proiezioni.film AND Proiezioni.sala==Sale.idSala AND Sale.idSala=='sala' AND Proiezioni.orario<orario AND orario<(Proiezioni.orario+3)

-Query per eliminare tutte le proiezioni (e di conseguenza i relativi biglietti) alle sale che sono state impostate non disponibili nella sezioni "Gestisci Sale".

proiezioni.delete().where(and_(proiezioni.c.sala==sale.c.idSala,sale.c.disponibile==True,sale.c.idSala==bindparam('sala'),proiezioni.c.orario>datetime.now()))

DELETE FROM Proiezioni

WHERE Proiezioni.sala==Sala.idSala AND Sala.disponibile==True AND Sala.idSala==’sala' AND Proiezioni.orario>NOW()

DESCRIZIONE DELLE PRINCIPALI SCELTE EFFETTUATE

Breve sintesi

Nel progetto sono state utilizzate transazioni per garantire l’integrità e la correttezza in qualsiasi momento del nostro database. Inoltre sono stati inseriti dei CHECK su tuple e sulle tabelle per il controllo di vincoli imposti in precedenza. Abbiamo anche utilizzato dei vincoli di foreign key per effettuare i collegamenti tra le varie tabelle. Abbiamo inserito anche delle politiche di reazione su queste foreign key per gestire al meglio gli update e le eliminazioni che vengono effettuate dalla webapp.

Vincoli

Check: nella tabella “Film” sono stati utilizzati per assicurarci che l’anno del film sia maggiore di 1970 (non potranno essere memorizzati quindi film più vecchi di 1970) e un film dovrà avere la durata al massimo di 1 minuto (la soglia è impostata così bassa perché è permessa anche l’aggiunta di cortometraggi). Nelle sale i posti devono necessariamente essere maggiori di 9, le file maggiori o uguali a 1 e il resto della divisione tra il numero dei posti e le file dev’essere 0 (come già detto sopra). Nelle proiezioni l’ora di proiezione deve essere maggiore dell’ 1 gennaio del 1970 e il prezzo del biglietto dev’essere maggiore uguale di 0. Nel biglietto invece il posto dev’essere maggiore uguale a 0.

Vincoli di foreign key: consultare il modello logico relazionale descritto precedentemente.

Politiche di reazione sulle foreign key: abbiamo specificato delle politiche di reazione per alcune chiavi esterne, per permettere l’automazione nella

gestione degli update e delle eliminazione dei record relativi ad alcune tabelle.

Attributo “film” di “GeneriFilm” è foreign key della primary key “idFilm” di “Film”.

Attributo “proiezione” di “Biglietti” è foreign key della primary key “idProiezione” di “Proiezioni”.

Attributo “cliente” di “Biglietti” è foreign key della primary key “email” di “Utenti”.

In tutte e 3 le chiavi esterne sono state messe le politiche ONDELETE CASCADE e ONUPDATE CASCADE.

In questo modo, ogni volta che noi eliminiamo/modifichiamo un record appartenente alla tabella in cui è definita la primary key associata alla foreign key verrà propagata l’eliminazione/modifica anche nel record della tabella dove è definita la foreign key. Questo insieme di politiche di reazione ci permette di mantenere l’integrità quando effettuiamo le operazioni di eliminazione di proiezioni (nella quale verranno eliminati anche tutti i biglietti associati) oppure quando effettuiamo la modifica tramite “Gestione Sale” al campo “disponibile”. In quel caso verranno eliminate tutte le proiezioni future e i relativi biglietti associate alle sale ora non più disponibili.

Transazioni

Usiamo le transazioni per permettere che più operazioni svolte dagli utenti della webapp in modo concorrente non danneggino l’integrità della base di dati.Usiamo le transazioni in tutti quei contesti in cui ci sono più operazioni sul database svolte in successione (in cui c’è almeno una scrittura) e che necessitano di essere eseguite come un’unica operazione atomica e serializzabile.

In particolare, sono usate nell’aggiunta di un cliente e di un gestore. Questo perchè l’aggiunta di un cliente/gestore effettua due operazioni in

successione: lettura sul DB per vedere se c'è un altro utente con stessa mail ; scrittura sul DB per riportare la creazione del nuovo utente. Se non usassimo le transazioni avremmo potenzialmente un problema nel momento in cui effettuassimo nella base di dati l'inserimento di un utente con la stessa mail nello stesso istante. (Uno dei due account di fatto non viene creato).

Inoltre anche l'acquisto di un biglietto è stato trattato come una transazione, dal momento che anche questa azione consiste in una lettura e una scrittura nel DB in successione. Con la lettura si vede se il posto per la proiezione è ancora libero/valido e poi si scrive sul DB riportando la creazione di un nuovo biglietto. Ciò significa che se due utenti comprassero lo stesso biglietto ciò darebbe problemi. La transazione è importante che inizi dalla fase di lettura nel database, poiché se effettuassimo la transazione solamente nel momento in cui l'utente scrive potrebbe essere già troppo tardi (avremmo il fenomeno dei lost update) dato che potremmo perdere un acquisto di un biglietto.

In tutte le operazioni di amministrazione del gestore sono state usate le transazioni poiché vengono effettuate in tutti i casi operazioni di lettura e successivamente scrittura nel database.

Aggiungi film, Aggiungi sala, Gestisci sala, Aggiungi proiezione ed elimina proiezione futura sono operazioni che vengono eseguite tutte con delle transazioni.

Sicurezza dall'esterno

Per risolvere un potenziale attacco di SQL injection abbiamo utilizzato nelle query degli appositi parametri che come abbiamo visto sopra passiamo a bindparam(): vengono inseriti dei placeholder nella query al posto di inserire direttamente l'input nello statement. Mediante l'uso del bindparam ogni comando SQL iniettato verrà trattato come non valido e non verrà accettato.

INFORMAZIONI AGGIUNTIVE

Per concludere riportiamo brevemente alcune informazioni utili per apprezzare il progetto

-Tutte le query le abbiamo scritte usando l'Expression Language di sqlalchemy. Inoltre abbiamo sempre pssato i parametri alle query in maniera opportuna e disciplinata, usando bindparam(): in questo modo preveniamo per costruzione l'sql injection.

-Abbiamo implementato il controllo degli accessi e la gestione dei ruoli della web app basandoci molto su quanto ci è offerto da flask login. Abbiamo definito la classe User sottoclasse di UserMixin e abbiamo definito il metodo "isGestore(self)" che ritorna True se l'utente è un cliente, False altrimenti. In ogni route si controlla se il current_user (altra funzionalità offerta da flask login, che ritorna l'oggetto User che rappresenta l'utente che sta interagendo con la web app in questo momento) possa accedere appunto a tale route. Per capire se l'utente attuale può accedere alla route bastano al massimo due ingredienti: basta capire se è autenticato o no, tramite current_user.is_authenticated (campo dell'oggetto) ; se è autenticato, capire se è cliente o gestore, tramite current_user.isGestore().

-Il codice del progetto si basa principalmente su due file python: webapp.py, dove sono definite tutte le route e tutto il codice per la web app ; database.py, dove sono definite le tabelle e dove sono definite tutte le funzioni necessarie per interagire con il database. Tutto il codice necessario per interagire con il database è in questo file, incapsulato in opportune funzioni. Disaccoppiamento codice webapp – codice interazione DB. In questo modo la leggibilità e la gestione del codice è migliore

-Le password degli utenti non le abbiamo salvate in chiaro, ma le abbiamo criptate tramite l’algoritmo di hash SHA-256.

-Abbiamo definito delle nostre eccezioni, in modo da avere una cattura e una gestione degli errori più fine e specifica. In particolare le due eccezioni ResultException e EmptyResultException sono lanciate dalle funzioni per l’interazione con il DB definite in database.py , e sono due in modo da distinguere due tipologie possibili di errori.
Inoltre il controllo e la gestione degli errori è tale che la web app è robusta e riesce a gestire anche casi particolari. Ad esempio casi in cui l’utente digita gli url e gli input da passare alle route tramite address bar.