Project Work

Gestione magazzino negozio libri

Database

Autore

Id	Nome	Cognome	NazioneResidenza
1	John Ronald	Tolkien	Inghilterra
2	Agatha	Christie	Inghilterra

LogWebOperation

Id	dataOra	URLRequest
1	2020-01-01 09:10:00	/user/
2	2020-01-01 09:11:12	/user/pBaudo

(1,n)

(1,1)

Libro

Isbn	Titolo	Descrizione	categoria	prezzo	nCopie	idAutore
123445432	Lo Hobbit	Epico	Fantasy	54	40	1
398498221	Dieci piccoli indiani	Emozionante	Giallo	12	120	2

Categoria

categoria
Fantasy
Giallo

(0,n)

(0,2)

(1,1)

(1,1)

Magazzino

Id	quantita	statoStock	isbn
1	100	richiesto	123445432
2	50	disponibile	123445432

richiesto → processato → disponibile

DAO

CategoriaDAO

insert(String categoria): void
delete(String categoria): void

LogDAO

Insert(LogElement l): void
select (LocalDateTime start):List<LogElement>

AutoreDAO

insert(Autore a): void
selectById (int id): Autore
selectByIsbn(int isbn): Autore
update(Autore a): void
delete(int id): void

LibroDAO

insert(Libro l): void
update(Libro l):void
delete(int isbn): void
selectForCategory(String genere): List<Libro>
selectForAuthor(int idAutore): List<Libro>
selectForIsbn(int isbn): Libro
selectAllPage (int offset, int page):List<Libro>
selectNBooksForCountry():Map<String, Integer>

Service

LibroService

registraNuovoLibro(Libro l): void
modificaDatiLibro(Libro l):void
eliminaLibro(int isbn): void
selezionaLibriPerAutore (int idAutore): List<Libro>
leggiDatiLibro(int id): Libro
selezionaTuttiLibri(int offset, int page):List<Libro>
statistichePerNazione():Map<String, Integer>
approvvigionaLibro(int isbn, int nCopie): void
cambiaStatoRichiesta (int isbn, String stato):void
leggiInfoStock(int isbn): List<StockElement>
chiudiRichiesta(int isbn): void

Libro

isbn: int
titolo: String
descrizione: String
categoria: String
prezzo: double
nCopie: int
autore: Autore
categoria: Categoria
richieste: List<StockElement>

StockElement

id: int
quantita: int
statoStock: String

registraNuovoLibro --> registra un libro nel sistema e richiede un approvvigionamento di 50 copie. Se il libro è già presente solleva un errore. Un Libro ha sempre un autore e una categoria che devono essere già registrati sul DB

l --> il libro che si vuole registrare

modificaDatiLibro --> modifica (tutti) i dati del libro. Se la modifica comporta che il numero di copie scende sotto la soglia dei 10 libri, allora va fatto un approvvigionamento automatico di 50 copie

l --> il libro con un nuovi dati

eliminaLibro --> elimina un singolo libro dal sistema. Il libro è eliminabile se non vi sono richieste di approvvigionamento nello stato processato. In tal caso si solleva una eccezione. Se il libro è eliminabile, vanno eliminate anche (tutte) le richieste di approvvigionamento

Isbn --> il codice del libro che si vuole eliminare

- **selezionaLibriPerAutore -->** recupera dal sistema tutti i libri di una certa categoria

  idAutore--> l'id dell'autore relativo ai libri che si vogliono recuperare

- **leggiDatiLibro -->** recupera dal sistema i dati di un singolo libro. Se il libro non è presente ritorna null
  - isbn --> il codice del libro che si vuole leggere

- **selezionaTuttiLibri -->** recupera dal sistema tutti i libri ordinati per titolo. La richiesta è paginata pertanto la lista risultato conterrà solo i libri della pagina richiesta
  - offset --> numero di righe risultato per la pagina
  - page = numero della pagina

  Ad esempio, se vogliamo i 10 libri della quinta pagina (page=5, offset = 10), dobbiamo restituire i libri dal 41-esimo al 50-esimo

- **eliminaLibro -->** elimina un singolo libro dal sistema (ma non l'autore)
  - Isbn --> il codice del libro che si vuole eliminare

- **statistichePerNazione -->** per ogni nazione, recupera il numero di libri in vendita. Per nazione si intende la nazione di appartenenza dell'autore

- **approvvigionaLibro -->** consente fare una richiesta di approvvigionamento rispetto ad un certo libro. Parametri:
  - isbn --> codice univoco del libro soggetto ad approvvigionamento
  - nCopie --> numero di copie richieste

- Se esiste già una richiesta in corso (statoStock = richiesto), aumentare se necessario la quantità attuale di copie richieste per arrivare al numero desiderato
  - Ad esempio se si vogliono 100 copie di un libro ma esiste già una richiesta per 70, modificare tale chiesta in modo che il numero di copie richieste sia 100
- Se esiste già una richiesta in corso (statoStock = processato), non sarà possibile fare ulteriori richieste di approvvigionamento ne modificare la richiesta in corso
- Se esiste già una richiesta in corso (statoStock = disponibile), non sarà possibile modificare la richiesta in corso ma sarà possibile fare una nuova richiesta di approvvigionamento della quantità residua
  - Ad esempio se si vogliono 100 copie di un libro ma esiste già una richiesta per 70, creare una nuova richiesta per 30 copie

- **cambiaStatoRichiesta -->** cambia lo stato di approvvigionamento di uno StockElement
  - isbn --> codice univoco del libro soggetto ad approvvigionamento
  - stato --> il nuovo stato che deve assumere la richiesta

  Il passaggio di stato può avvenire soltanto da «richiesto» a «processato» oppure da «processato» a «disponibile»

  Se si passa di stato da «processato» a «disponibile» e esiste già una riga con stato «disponibile», accorpare le due righe sommando la quantità
  - Ad esempio se esiste una richiesta da 30 copie nello stato «processato» che deve passare nello stato «disponibile» ed esiste già anche un seconda richiesta nello stato «disponibile» da 100 copie, accorpare le due righe in una sola con numero di copie 130

- **leggiStockInfo -->** recupera le informazioni di riordino relative ad un libro. Il metodo deve tornare una lista con al massimo 2 elementi di tipo StockElement
  - isbn --> codice univoco del libro soggetto ad approvvigionamento

- **chiudiRichiesta -->** cancella la richiesta di approvvigionamento per un libro e aggiorna il numero di libri disponibili. Una richiesta si può chiudere solo se si trova nello stato «disponibile».

  Se non è possibile chiudere la richiesta sollevare una exception
  - isbn --> codice univoco del libro soggetto ad approvvigionamento

AutoreService

registraAutore(Autore a): void
modificaDatiAnagraficiAutore(Autore a):void
eliminaAutore(int id): void
selezionaAutorePerLibro(int isbn): Autore
leggiDatiAutore(int id): Autore
LeggiAutori(): List<Autore>

Autore

id: int
nome: String
cognome: String
nazioneResidenza: String

- **registraAutore -->** registra un nuovo Autore nel sistema. Se l' autore è già presente solleva un errore.
  - a --> l'autore che si vuole registrare

- **modificaDatiAnagraficiAutore -->** modifica (tutti) i dati dell'autore.
  - a --> l'autore che si vuole registrare

- **eliminaAutore -->** elimina un singolo autore dal sistema. L'autore è eliminabile se non vi sono libri da lui scritti nel sistema
  - Isbn --> il codice del libro che si vuole eliminare

- **selezionaAutorePerLibro -->** recupera dal sistema tutti l'autore di un certo libro
  - isbn--> il codice del libro di cui si vuole l'autore

- **leggiDatiAutore-->** recupera dal sistema i dati di un singolo Autore. Se l'autore non esiste ritorna null
  - id --> il codice dell'autore che si vuole leggere

- **LeggiAutori -->** recupera la lista di tutti gli autori

LogService

registraLogElement(LogElement le): void
statisticheNelPeriodo(LocalDateTime ldt):List<LogElement>

LogElement

id: int
dataOra: LocalDateTime
URLRequest: String

- **registraLogElement-->** registra un nuovo LogElement nel sistema.
  - le --> l'autore che si vuole registrare

- **statisticheNelPeriodo -->** ritorna una lista di tutti i LogElementi in base alla data di inizio e allo httsStatus.
  - ldt --> la data e ora di partenza da cui si vogliono i LogElement. Se è null, si devono considerare tutte le date

CateogiraService

registraCategoria(String categoria): void
eliminaCategoria(String categoria): void

Categoria

categoria: String

registraCategoria --> registra una nuova Categoria nel sistema. Se la categoria è già presente solleva un errore.

categoria --> il nome della categoria che si vuole registrare

eliminaCategoria --> elimina una singola categoria dal sistema. La categoria è eliminabile se non vi sono libri associati

categoria --> il nome della categoria che si vuole eliminare

Controller

Controller relativa ai servizi web per i libri

url	http Function	http Status success	http Status failure	Return type	Descrizione
/books	POST	CREATED	CONFLICT (chiave duplicata) BAD REQUEST (vincolo di integrità referenziale violato	ResponseEntity<Void>	Registra un nuovo libro sul sistema
/books/{isbn}	PUT	OK	NOT FOUND (libro non presente)	ResponseEntity<Libro>	Modifica tutti i dati di un libro
/books/{isbn}	DELETE	NOT CONTENT	NOT FOUND(libro non presente) CONFLICT (libro non cancellabile)	ResponseEntity<Libro>	Cancella un singolo libro
/books/{isbn}	GET	OK	NOT FOUND (libro non presente)	ResponseEntity<Libro>	Legge i dati di un singolo libro
/books/{country}/stat	GET	OK		ResponseEntity <MapLibroDTO>	Recupera il numero di libri per nazione
/books/{isbn}/author	GET	OK	NOT FOUND (libro non presente)	ResponseEntity<Autore>	Leggi l'autore di un certo libro

NOTA: quando viene invocato un qualsiasi servizio bisogna inserire una riga nella tabella LogWebOperation

url	http Function	http Status success	http Status failure	Return type	Descrizione
/books/{isbn}/warehouse	POST	OK	NOT FOUND (libro non disponibile)	ResponseEntity <Void>	Effettua una nuova richiesta di approvvigionamento per un libro
/books/{isbn}/warehouse	PUT	OK	NOT FOUND (libro non disponibile)	ResponseEntity <Void>	Fa evolvere una richiesta di approvvigionamento per un libro
/books/{isbn}/warehouse	DELETE	OK	NOT FOUND (libro non disponibile)	ResponseEntity <Void>	Chiude una richiesta di approvvigionamento per un libro
/books/{isbn}/warehouse	GET	OK	NOT FOUND (libro non disponibile)	ResponseEntity <ListStockElementDTO>	Legge i StockElements presenti per un certo libro

NOTA: quando viene invocato un qualsiasi servizio bisogna inserire una riga nella tabella LogWebOperation

Controller relativa ai servizi web per gli autori

url	http Function	http Status success	http Status failure	Return type	Descrizione
/authors	POST	CREATED	CONFLICT (chiave duplicata)	ResponseEntity<Void>	Registra un nuovo autore sul sistema
/authors/{id}	PUT	OK	NOT FOUND (autore non presente)	ResponseEntity<Autore>	Modifica tutti i dati di un autore
/authors/{id}	DELETE	NOT CONTENT	NOT FOUND (autore non presente)	ResponseEntity<Autore>	Cancella un singolo autore
/authors/{id}	GET	OK	NOT FOUND (autore non presente)	ResponseEntity<Autore>	Legge i dati di un singolo autore
/authors	GET	OK	NOT FOUND (autori non presenti)	ResponseEntity <ListAutoreDTO>	Leggi tutti gli autori
/authors/{idAutore}/libri	GET	OK	NOT FOUND (libri non presenti)	ResponseEntity <ListLibroDTO>	Leggi tutti i libri di un certo autore

NOTA: tutti i servizi devono scrivere nella tabella LogWebOperation e riportare nel DB l'esito della operazione

Controller relativa ai servizi web per il magazzino

url	http Function	http Status success	http Status failure	Return type	Descrizione
/warehouse/{status}	PUT	OK	NOT FOUND (autore non presente)	ResponseEntity<StockElement>	Modifica li dati di uno StockElement
/warehouse/{isbn}	GET	OK	NOT FOUND (autore non presente)	ResponseEntity<List<StockElement>>	Recupera gli StockElement di un libro

NOTA: quando viene invocato un qualsiasi servizio bisogna inserire una riga nella tabella LogWebOperation

Controller relativa al log

url	http Function	http Status success	http Status failure	Return type	Descrizione
/losa/stat/	GET	OK	NOT FOUND (log non presenti)	ResponseEntity <List<LogElement>	Recupera tutti i logElements