FUSU ELENA CASIANA

14.01.2020  Grupa 412

Biblioteca online

# INFORMAȚII GENERALE

*Biblioteca online* este o aplicație web dezvoltată cu scopul de a facilita managementul cărților dintr-o bibliotecă, precum și procesul de împrumut/restituire cărți. Prin intermediul aplicației se poate păstra o evidență clară a stocului de cărți al bibliotecii, iar cititorii pot consulta cu ușurință dacă o carte există sau nu în biblitotecă.

# TEHNOLOGII FOLOSITE

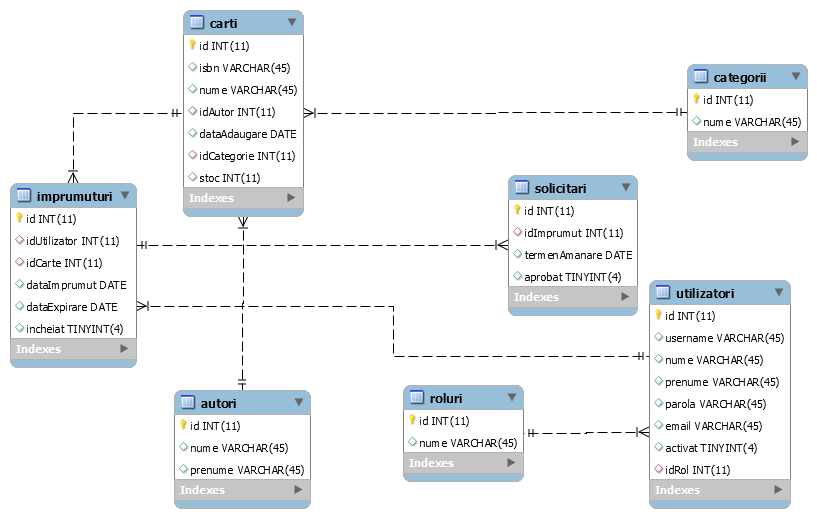
Aplicația prezentată este dezvoltată în limbajul Java v11, folosind framework-ul Spring Boot v2.4.1. Baza de date folosită este MySql. Funcționalitățile aplicației vor fi prezentate folosind utilitarul Postman.

# ENTITĂȚI

Entitățile pe care se bazează funcționalitatea aplicației sunt:

* Carte
* Categorie
* Solicitare
* Împrumut
* Rol
* Autor
* Utilizator

Fiecărei entități din baza de date, îi corespund clase în cadrul aplicației. Fiecare entitate folosită are definite următoarele tipuri de clase: repository, controller, service, model, dto și mapper. Pentru maparea automată între entitățile interne aplicației și entitățile externe de tip DTO (Data Transfer Objects) am folosit MapStruct v1.3.1. De exemplu, pentru Carte avem: Carte (modelul), CarteDto, CarteController, CarteMapper, CarteRepository și CarteService.



# FUNCȚIONALITĂȚI IMPLEMENTATE

### Carte REST endpoints:

1. Filtrarea listei de cărți după numele autorului: [http://localhost:8080/carti/getbyauthor/{id\_autor}](http://localhost:8080/carti/getbyauthor/%7bid_autor%7d)
2. Obținerea unei liste ce cuprinde toate cărțile folosind request-ul <http://localhost:8080/carti/>
3. Filtrarea listei de cărți după numele cărții: [http://localhost:8080/carti/getbyname/{id\_autor}](http://localhost:8080/carti/getbyname/%7bid_autor%7d)
4. Obținerea unei cărți în funcție de id: [http://localhost:8080/carti/{id\_carte}](http://localhost:8080/carti/%7bid_carte%7d). Dacă nu există, se returnează un mesaj de eroare de forma "Carte - acest obiect nu a fost găsit".
5. Adăugare carte, ce primește în request body un JSON și returnează un JSON cu obiectul nou creat: <http://localhost:8080/carti/create>.

PK este generată automat la nivelul bazei de date. Validările realizate la adăugare carte sunt descrise în secțiunea următoare (metoda *validateRequest(Carte request)* din CarteValidatorService).

1. Modificare detalii carte, ce primește în request body un JSON și returnează numărul obiectelor modificate: <http://localhost:8080/carti/update> .

Id-ul cărții pentru care se dorește modificare este transmis prin JSON. Sunt realizate validări (metoda *validateUpdateRequest(Carte request)* din Carte ValidatorService).

1. Ștergerea unei cărți din gestiune: [http://localhost:8080/carti/delete/{id\_carte}](http://localhost:8080/carti/delete/%7bid_carte%7d)

Atât în cazul realizării cu succes a acțiunii, cât și în cazul în care nu se găsește o carte cu acest id, este returnat un mesaj sugestiv.

! Înainte de ștergere se interoghează BD pentru a vedea dacă există un împrumut pentru această carte (metoda *validateDelete(int id\_carte)* din CarteValidatorService)

### Autor REST endpoints:

1. Obținerea unei liste ce cuprinde toți autorii: <http://localhost:8080/autori>
2. Obținerea unui autor după id: [http://localhost:8080/autori/{id\_autor}](http://localhost:8080/autori/%7bid_autor%7d)
3. Introducerea unui nou autor în BD: <http://localhost:8080/autori/create> . Primește în request body un JSON și returnează un JSON cu obiectul nou creat. PK este generat la nivelul BD. Sunt realizate validări (metoda *validateRequest(Autor request)* din AutorValidatorService).
4. Modificarea datelor unui autor: <http://localhost:8080/autori/update> . ID-ul autorului pentru care se dorește modificarea se specifică în JSON. Sunt realizate validări.
5. Ștergere autor: [http://localhost:8080/autori/delete/{id\_autor}](http://localhost:8080/autori/delete/%7bid_autor%7d) . Request-ul returnează mesaje sugestive în cazul în care acțiunea s-a realizat cu succes sau nu.

! Înainte de ștergere se verifică dacă există o carte cu acest autor (metoda *validateDelete(int id\_autor)* din AutorValidatorService)

### Categorie REST endpoints:

1. Obținerea unei liste ce cuprinde toate categoriile: <http://localhost:8080/categorii>
2. Obținerea unei categorii după id: [http://localhost:8080/categorii/{id\_categorie}](http://localhost:8080/categorii/%7bid_categorie%7d) . Dacă nu există, se returnează un mesaj de eroare de forma "Categorie - acest obiect nu a fost găsit".
3. Introducerea unei noi categorii: <http://localhost:8080/categorii/create>. Primește în request body un JSON și returnează un JSON cu obiectul nou creat. PK este generat la nivelul BD. Sunt realizate validări (metoda *validateRequest(Categorie request)* din CategorieValidatorService).
4. Modificarea unei categorii: <http://localhost:8080/categorii/update>. ID-ul categoriei pentru care se dorește modificarea se specifică în JSON. Sunt realizate validări.
5. Ștergerea unei categorii: [http://localhost:8080/categorii/delete/{id\_categorie}](http://localhost:8080/categorii/delete/%7bid_categorie%7d).

! Se verifică mai întâi dacă există o carte din acea categorie.

### Împrumut REST endpoints:

1. Obținerea unei liste cu toate împrumuturile: <http://localhost:8080/imprumuturi>
2. Obținerea unei liste cu toate împrumuturile neîncheiate: <http://localhost:8080/imprumuturi/getImprumuturiNeincheiate>
3. Obținerea unui împrumut în funcție de id: [http://localhost:8080/imprumuturi/{id\_imprumut}](http://localhost:8080/imprumuturi/%7bid_imprumut%7d)
4. Înregistrarea în BD a unui nou împrumut: <http://localhost:8080/imprumuturi/create>.

! Pentru că ne dorim ca această acțiune să fie făcută doar de bibliotecari am simulat acest comportament prin adăugarea proprietății idUtilizatorAutentificat în ImprumutDto.

La adăugare împrumut se verifică dacă utilizatorul există și dacă are rolul de bibliotecar (admin). Alte validări realizate la acest nivel sunt descrise în secțiunea următoare.

La împrumut numărul cărților cu id-ul din împrumut request scade. Metoda *create* din ImprumutService este o tranzacție.

1. Închiderea unui împrumut: <http://localhost:8080/imprumuturi/update>. Prin JSON se specifică id-ul împrumutului.
2. Ștergere împrumut: [http://localhost:8080/imprumuturi/delete/{id\_imprumut}/{id\_utilizatorAutentificat}](http://localhost:8080/imprumuturi/delete/%7bid_imprumut%7d/%7bid_utilizatorAutentificat%7d)

Acțiunea de ștergere este tot o tranzacție.

! Dacă împrumutul în momentul ștergerii nu era încheiat, stocul cărții crește.

### Utilizator REST endpoints:

1. Obținerea unei liste ce cuprinde toți utilizatorii: <http://localhost:8080/utilizatori>
2. Obținerea unui utilizator în funcție de id: http://localhost:8080/utilizatori/{id\_utilizator} . Dacă nu există, se returnează un mesaj de eroare de forma "Utilizator - acest obiect nu a fost găsit".
3. Introducerea unui nou utilizator în BD: <http://localhost:8080/utilizatori/create>. Informațiile utilizatorului se trimit printr-un JSON. Se realizează validări prin metoda *validateRequest(Utilizator request)* din UtilizatorValidatorService. PK este generat la nivelul BD. Câmpul „activat” se completează automat cu valoarea „false”.
4. Modificare detalii utilizator: <http://localhost:8080/utilizatori/update>. ID-ul utilizatorului pentru care se dorește modificarea se specifică în JSON. Sunt realizate validări.
5. Activare cont utilizator: [http://localhost:8080/utilizatori/activate/{id\_utilizator}](http://localhost:8080/utilizatori/activate/%7bid_utilizator%7d). Dacă era deja activat contul, se returnează un mesaj sugestiv.
6. Ștergere cont: [http://localhost:8080/utilizatori/delete/{id\_utilizator}](http://localhost:8080/utilizatori/delete/%7bid_utilizator%7d)

! Înainte de ștergere se verifică dacă utilizatorul are împrumuturi active.

### Solicitare REST endpoints:

1. Vizualizarea tuturor cererilor de amânare: <http://localhost:8080/solicitari/>
2. Crearea unei noi cereri de amânare: <http://localhost:8080/solicitari/create/>.

Data de amânare este trimisă printr-un JSON. Se verifică dacă împrumutul pentru care s-a solicitat amânarea nu a fost deja încheiat.

1. Aprobarea unei cereri de amânare: [http://localhost:8080/solicitari/aproba/{id}/{idUtilizatorAutentificat}](http://localhost:8080/solicitari/aproba/%7bid%7d/%7bidUtilizatorAutentificat%7d).

ID-ul utilizatorului pentru care se dorește modificarea se specifică în JSON.

### Rol REST endpoints:

1. Obținerea unei liste ce conține toate rolurile:

[http://localhost:8080/roluri/{ idUtilizatorAutentificat}](http://localhost:8080/roluri/%7b%20idUtilizatorAutentificat%7d)

1. Afișarea rolului în funcție de id:

[http://localhost:8080/roluri/{id\_rol}/{ idUtilizatorAutentificat}](http://localhost:8080/roluri/%7bid_rol%7d/%7b%20idUtilizatorAutentificat%7d)

1. Crearea unui nou rol:

[http://localhost:8080/roluri/create/{idUtilizatorAutentificat}](http://localhost:8080/roluri/create/%7bidUtilizatorAutentificat%7d)

1. Update rol:

[http://localhost:8080/roluri/update/{idUtilizatorAutentificat}](http://localhost:8080/roluri/update/%7bidUtilizatorAutentificat%7d)

1. Ștergere rol:

[http://localhost:8080/roluri/delete//{id\_rol}/{ idUtilizatorAutentificat}](http://localhost:8080/roluri/delete//%7bid_rol%7d/%7b%20idUtilizatorAutentificat%7d)

# VALIDARE

Pentru validare au fost create clase dedicate:

* AuthValidatorService (verifică dacă un utilizator este autorizat pentru realizarea unei anumite acțiuni în funcție de rolul acestuia)
* AutorValidatorService (numele și prenumele să conțină doar litere, numele autorului să fie unic la nivelul bazei de date, la ștergere Autor să nu fie cărți cu acel autor)
* CarteValidatorService (numele cărți să conțină doar litere sau cifre, data adăugării să nu fie din trecut, stocul să nu fie mai mic decât 0, autorul și categorie să existe în BD, la ștergere carte să nu existe împrumuturi cu acea carte)
* CategorieValidatorService (numele categoriei să conțină doar litere sau cifre, numele categoriei să fie unic la nivelul BD, în cazul ștergerii categoriei să nu existe cărți cu acea categorie)
* ImprumutValidatorService (se verifică dacă împrumutul a fost deja încheiat, dacă utilizatorul și cartea pentru care se face împrumutul există, data de expirare să nu fie mai mică decât data împrumutului, data de împrumut să nu fie din viitor)
* UtilizatorValidatorService (numele și prenumele să conțină doar litere, email-ul să fie valid, rolul utilizatorului să existe în BD, verificare daca este activat, verificare dacă are împrumuturi active înainte de ștergerea contului)

# ERROR HANDLING

Pentru tratarea erorilor folosim următoarele clase:

* BadRequestException
* EntityNotFoundException
* NotAuthorizedException
* PropertyNotGoodException

# TESTARE

Pentru testarea aplicației am folosit testare unitară. Prin intermediul acestor teste ne asigurăm că orice modificare adusă ulterior aplicației, nu va strica vechea funcționalitate. REST endpoint-urile sunt testate pentru toate modelele. Service-urile sunt testate pentru: AutorService, CarteService, CategorieService, UtilizatorService și AutorValidatorService, RolService.