Source code

https://gitlab.cs.ut.ee/tenman/esi-project

Security aspects

REST API is secured by Spring Boot Security with JWT

Role	Description
LIBRARIAN	• Customers must be able to register at the librarian. For this they need to provide their name, address, email, phone number and ID. The system must check the ID. • Customers must be able to return books by coming in and handing the book to the librarian. If the deadline has passed then the customer has to pay for the fine. • If a book is returned, the system must check whether the deadline for the book has passed. • Also LIBRARIAN can do everything that CUSTOMER can do.
CUSTOMER	• Customers should be able to register to the system online and provide personal information such as described above. • On our online platform customers must be able to browse through the catalog containing all of our books. • On our online platform customers must be able to see the status of a book - whether it is available for rent or unavailable. • On our online platform customers must be able to reserve a book for themselves. This will change the status of a book to unavailable for a day. • Customers must be able to come into our library, pick out a book and rent it. To do this they must provide their account information. • If a book is returned, the system must check whether the deadline for the book has passed. • The online platform must show the history of rented books to the customer if they are logged in.

	● The online platform must show the current rented books and their deadlines to the customer. ● Customers must be able add a payment card in order to pay their fine online. ● The customer should not be allowed to book or rent a book if they have a fine on their account. This must be paid before a new book can be rented. ● The system allows the librarian to insert a book by providing a certain title, an author, a release date and a reference number. ● The system must allow the librarian to remove the fine from a customer's account if they pay by cash on site. ● The system must allow the librarian to change the status of a book. If a customer calls to reserve a book then a librarian can change an available book to unavailable. And also the other way if a customer wants to remove their booking. ● The system must allow search for books by title, author and filtering books by status, by language, by year, by review etc. ● The system must allow customers to reserve books for a day. This will change the book's status to unavailable for a day. If the person rents it out during the day then the status time will be extended. ● The system must allow a librarian to remove a book if it has gone missing or been destroyed etc.

Api Documentation

Overview

Api Documentation

Version information

Version : 1.0

License information

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0
Terms of service : urn:tos

URI scheme

Host : esitartulibrary.herokuapp.com
BasePath : /

Tags

- authentication-controller : Authentication Controller
- book-controller : Book Controller
- customer-controller : Customer Controller
- review-controller : Review Controller
- user-controller : User Controller

Paths

Inserts new book

```
POST /books
```

Parameters

Type	Name	Description	Schema
Body	**createBookRequest** *required*	createBookRequest	CreateBookRequest

Responses

HTTP Code	Description	Schema
200	OK	Book
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- book-controller

Retrieves all books

```
GET /books
```

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- book-controller

Updates the book

```
PUT /books
```

Parameters

Type	Name	Description	Schema
Body	**updateBookRequest** *required*	updateBookRequest	UpdateBookRequest

Responses

HTTP Code	Description	Schema
200	OK	Book
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- */*

Tags

- book-controller

Retrieves books by author

```
GET /books/authors/{author}
```

Parameters

Type	Name	Description	Schema
Path	**author** *required*	author	string

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves available books

```
GET /books/available
```

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves books by language

```
GET /books/languages/{lang}
```

Parameters

Type	Name	Description	Schema
Path	**lang** *required*	lang	string

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves books by status

```
GET /books/statuses/{status}
```

Parameters

Type	Name	Description	Schema
Path	**status** *required*	status	enum (AVAILABLE, BOOKED, RENTED)

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves books by title

```
GET /books/titles/{title}
```

Parameters

Type	Name	Description	Schema
Path	**title** *required*	title	string

Responses

HTTP Code	Description	Schema
200	OK	< Book > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves books by year

```
GET /books/years/{year}
```

Parameters

Type	Name	Description	Schema
Path	**year** *required*	year	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	< Book > array

HTTP Code	Description	Schema
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Retrieves the book by its id

```
GET /books/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	Book
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- book-controller

Delete the book by its id

```
DELETE /books/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	No Content
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- */*

Tags

- book-controller

Inserts new customer

```
POST /customers
```

Parameters

Type	Name	Description	Schema
Body	**customerDto** *required*	customerDto	CustomerDto

Responses

HTTP Code	Description	Schema
200	OK	CustomerDto
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- customer-controller

Retrieves all customers

```
GET /customers
```

Responses

HTTP Code	Description	Schema
200	OK	< Customer > array
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- customer-controller

Updates the customer by its id

```
PUT /customers
```

Parameters

Type	Name	Description	Schema
Body	**customerDto** *required*	customerDto	CustomerDto

Responses

HTTP Code	Description	Schema
200	OK	CustomerDto
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- */*

Tags

- customer-controller

Retrieves the customer by its id

```
GET /customers/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	Customer
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- customer-controller

Delete the customer by its id

```
DELETE /customers/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	No Content
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- */*

Tags

- customer-controller

Creates a new review for a book

```
POST /reviews
```

Parameters

Type	Name	Description	Schema
Body	**review** *required*	review	Review

Responses

HTTP Code	Description	Schema
200	OK	Review

HTTP Code	Description	Schema
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- review-controller

Updates review

```
PUT /reviews
```

Parameters

Type	Name	Description	Schema
Body	**review** *required*	review	Review

Responses

HTTP Code	Description	Schema
200	OK	Review
201	Created	No Content
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- review-controller

Delete the review by its id

```
DELETE /reviews/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	No Content
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- */*

Tags

- review-controller

Inserts new user

```
POST /users
```

Parameters

Type	Name	Description	Schema
Body	**user** *required*	user	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- application/json

Produces

- */*

Tags

- user-controller

Retrieves all users

```
GET /users
```

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- user-controller

Updates the user by its id

```
PUT /users
```

Parameters

Type	Name	Description	Schema
Body	**user** *required*	user	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content

HTTP Code	Description	Schema
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- user-controller

authorize

```
POST /users/authenticate
```

Parameters

Type	Name	Description	Schema
Body	**loginDto** *required*	loginDto	LoginDTO

Responses

HTTP Code	Description	Schema
200	OK	JWTToken
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- authentication-controller

getActualUser

```
GET /users/current
```

Responses

HTTP Code	Description	Schema
200	OK	User
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- user-controller

Retrieves the user by its id

```
GET /users/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	User
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- user-controller

Delete the user by its id

```
DELETE /users/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	No Content

HTTP Code	Description	Schema
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- */*

Tags

- user-controller

Definitions

Authority

Name	Schema
id *optional*	integer (int64)
role *optional*	enum (CUSTOMER, LIBRARIAN)

Book

Name	Description	Schema
author *optional*		string

Name	Description	Schema
categories *optional*		< enum (ACTION_AND_ADVENTURE, ALTERNATE_HISTORY, ANTHOLOGY, ART_ARCHITECTURE, AUTOBIOGRAPHY, BIOGRAPHY, BUSINESS_ECONOMICS, CHICK_LIT, CHILDREN_S, CLASSIC, COMIC_BOOK, COMING_OF_AGE, COOKBOOK, CRAFTS_HOBBIES, CRIME, DIARY, DICTIONARY, DRAMA, ENCYCLOPEDIA, FAIRYTALE, FANTASY, GRAPHIC_NOVEL, GUIDE, HEALTH_FITNESS, HISTORICAL_FICTION, HISTORY, HOME_AND_GARDEN, HORROR, HUMOR, JOURNAL, MATH, MEMOIR, MYSTERY, PARANORMAL_ROMANCE, PHILOSOPHY, PICTURE_BOOK, POETRY, POLITICAL_THRILLER, PRAYER, RELIGION_SPIRITUALITY_AND_NEW_AGE, REVIEW, ROMANCE, SATIRE, SCIENCE, SCIENCE_FICTION, SELF_HELP, SHORT_STORY, SPORTS_AND_LEISURE, SUSPENSE,

Name	Description	Schema
id *optional*		integer (int64)
language *optional*		string
releaseDate *optional*	**Example** : `"yyyy"`	string
status *optional*		enum (AVAILABLE, BOOKED, RENTED)
title *required*		string

BookRentingRequest

Name	Schema
book *optional*	Book
customer *optional*	Customer
fineAmount *optional*	number
id *optional*	integer (int64)
rentedAt *optional*	string (date-time)
rentedUntil *optional*	string (date-time)
status *optional*	enum (CANCELLED, DECLINED, EXPIRED, OPEN, RENTED)

CreateBookRequest

Name	Description	Schema
author *required*	**Length** : 2 - 2147483647	string

Name	Description	Schema
categories *optional*		< enum (ACTION_AND_ADVENTURE, ALTERNATE_HISTORY, ANTHOLOGY, ART_ARCHITECTURE, AUTOBIOGRAPHY, BIOGRAPHY, BUSINESS_ECONOMICS, CHICK_LIT, CHILDREN_S, CLASSIC, COMIC_BOOK, COMING_OF_AGE, COOKBOOK, CRAFTS_HOBBIES, CRIME, DIARY, DICTIONARY, DRAMA, ENCYCLOPEDIA, FAIRYTALE, FANTASY, GRAPHIC_NOVEL, GUIDE, HEALTH_FITNESS, HISTORICAL_FICTION, HISTORY, HOME_AND_GARDEN, HORROR, HUMOR, JOURNAL, MATH, MEMOIR, MYSTERY, PARANORMAL_ROMANCE, PHILOSOPHY, PICTURE_BOOK, POETRY, POLITICAL_THRILLER, PRAYER, RELIGION_SPIRITUALITY_AND_NEW_AGE, REVIEW, ROMANCE, SATIRE, SCIENCE, SCIENCE_FICTION, SELF_HELP, SHORT_STORY, SPORTS_AND_LEISURE, SUSPENSE,

Name	Description	Schema
language *required*		string
title *required*	**Length** : 2 - 2147483647	string
year *required*		integer (int32)

Customer

Name	Schema
bookRentingRequests *optional*	< BookRentingRequest > array
fineAmount *optional*	number
id *optional*	integer (int64)
idCode *required*	string
user *optional*	User

CustomerDto

Name	Description	Schema
activated *optional*		boolean
address *optional*		string
email *optional*		string

Name	Description	Schema
fineAmount *optional*		number
firstName *required*	**Length** : `2 - 50`	string
idCode *optional*		string
lastName *required*	**Length** : `2 - 50`	string
passWord *required*	**Length** : `4 - 50`	string
phoneNumber *optional*		string
roles *optional*		< enum (CUSTOMER, LIBRARIAN) > array
userId *optional*		integer (int64)
userName *required*	**Length** : `4 - 50`	string

JWTToken

Name	Schema
token *optional*	string

LoginDTO

Name	Description	Schema
password *required*	**Length** : `4 - 100`	string

Name	Description	Schema
rememberMe *optional*		boolean
username *required*	**Length** : `1 - 50`	string

Review

Name	Schema
book *optional*	Book
comment *optional*	string
id *optional*	integer (int64)
score *optional*	integer (int32)

UpdateBookRequest

Name	Description	Schema
author *required*	**Length** : `2 - 2147483647`	string

Name	Description	Schema
categories *optional*		< enum (ACTION_AND_ADVENTURE, ALTERNATE_HISTORY, ANTHOLOGY, ART_ARCHITECTURE, AUTOBIOGRAPHY, BIOGRAPHY, BUSINESS_ECONOMICS, CHICK_LIT, CHILDREN_S, CLASSIC, COMIC_BOOK, COMING_OF_AGE, COOKBOOK, CRAFTS_HOBBIES, CRIME, DIARY, DICTIONARY, DRAMA, ENCYCLOPEDIA, FAIRYTALE, FANTASY, GRAPHIC_NOVEL, GUIDE, HEALTH_FITNESS, HISTORICAL_FICTION, HISTORY, HOME_AND_GARDEN, HORROR, HUMOR, JOURNAL, MATH, MEMOIR, MYSTERY, PARANORMAL_ROMANCE, PHILOSOPHY, PICTURE_BOOK, POETRY, POLITICAL_THRILLER, PRAYER, RELIGION_SPIRITUALITY_AND_NEW_AGE, REVIEW, ROMANCE, SATIRE, SCIENCE, SCIENCE_FICTION, SELF_HELP, SHORT_STORY, SPORTS_AND_LEISURE, SUSPENSE,

Name	Description	Schema
id *required*		integer (int64)
language *required*		string
title *required*	**Length** : `2 - 2147483647`	string
year *required*		integer (int32)

User

Name	Description	Schema
activated *optional*		boolean
address *optional*		string
authorities *optional*		< Authority > array
email *optional*		string
firstName *required*	**Length** : `2 - 50`	string
id *optional*		integer (int64)
lastName *required*	**Length** : `2 - 50`	string
password *required*	**Length** : `4 - 100`	string
phoneNumber *optional*		string

Name	Description	Schema
type *optional*		enum (CUSTOMER, LIBRARIAN)
username *required*	**Length** : `4 - 50`	string