Library Management System API Documentation

1. Introduction

The Library Management System API allows users to manage books, authors and handle borrowing transactions.

Key Features:

- Books Management: CRUD operations for books, like filtering,
- Users Management: CRUD operations for users (authors, borrowers etc)

Base URL: http://localhost:8080/

Response Format: JSON

2. Book (BookController)

- POST /book/create

Description: adds a book to the database and returns a book response DTO.

Request(Body):

```
{   "title": "La belle et la bete",
    "isbn": "0306406152",
    "category": "FICTION"}
```

Response (200 OK):

```
{
    "id": 66,
    "title": "Error-correction coding for digital communications",
    "isbn": "0306406152",
    "category": "FICTION",
    "availability": true
}
```

Response (500 Internal Server Error- if one of the fields in the request body doesn't match any of the request fields):

```
{
    "timestamp": "2025-02-03T10:05:19.451+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/create"
}
```

Response (400 Bad Request- if the category is invalid):

```
{
    "timestamp": "2025-02-03T10:07:32.861+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/book/create"
}
```


- GET /book/view/{id}

Description: displays the details of the book with the given ID

Path parametres:

Id (integer) – the ID of the book to display

Example request: http://localhost:8080/book/view/7

Response (200 OK):


```
{
    "id": 7,
    "title": "Les malheurs de Sophie",
    "isbn": "987 239 324",
    "category": "FRENCH_LITERATURE",
    "availability": true
}
```


Response (500 internal server error- if the provided ID doesn't match any of the books present in the database):


```
{
    "timestamp": "2025-02-03T10:14:45.274+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/view/100"
}
```


- PATCH /book/set-book-unavailable/{id}

Description: used for testing purposes to set the availability of a book to false

Path parametres:

Id (Integer)- the ID of the book whose availability will be modified

Example request: http://localhost:8080/book/set-book-unavailable/7

Response (200 OK):


```
{
    "id": 7,
    "title": "Les malheurs de Sophie",
    "isbn": "987 239 324",
    "category": "FRENCH_LITERATURE",
    "availability": false
}
```


Response (500 internal server error- if the ID provided doesn't correspond to any book)


```
{
    "timestamp": "2025-02-03T10:19:58.747+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/set-book-unavailable/100"
}
```


- PATCH /book/set-book-available/{id}

Description: used for testing purposes to set the availability of a book to true

Path parametres:

Id (Integer)- the ID of the book whose availability will be modified

Example request: http://localhost:8080/book/set-book-available/7

Response (200 OK):


```
{
    "id": 7,
    "title": "Les malheurs de Sophie",
    "isbn": "987 239 324",
    "category": "FRENCH_LITERATURE",
    "availability": true
}
```

Response (500 internal server error- if the ID provided doesn't correspond to any book)


```
{
    "timestamp": "2025-02-03T10:22:39.882+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/set-book-available/100"
}
```


- GET /get-all-books

Description: displays all books in the database

Example request: http://localhost:8080/book/get-all-books

Response (200 OK):


```
[
    {
        "id": 8,
        "title": "L'ecole de la guerre",
        "isbn": "239 234 023",
        "category": "POLITICS",
        "availability": true
    },
    {
        "id": 9,
        "title": "Circuits electriques- EECE 210",
        "isbn": "349 230 938",
        "category": "ELECTRIC_CIRCUITS",
        "availability": true
    },
    {
        "id": 10,
        "title": "Electric circuits- EECE 290",
        "isbn": "345 945 932",
        "category": "ELECTRIC_CIRCUITS",
        "availability": false
    },

    {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": true
    },
```

```
    {
        "id": 7,
        "title": "Les malheurs de Sophie",
        "isbn": "987 239 324",
        "category": "FRENCH_LITERATURE",
        "availability": true
    }
]
```


- GET /book/find-by-id/{id}

Description: displays les details du book au ID correspondant

Path parametres:

Id (integer) – le id du book a retrieve

Example request: http://localhost:8080/book/view/7

Response (200 OK):


```
{
    "id": 7,
    "title": "Les malheurs de Sophie",
    "isbn": "987 239 324",
    "category": "FRENCH_LITERATURE",
    "availability": true
}
```


Response (500 internal server error- si le ID provided ne correspond a aucun des books):


```
{
    "timestamp": "2025-02-03T10:14:45.274+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/view/100"
}
```


- GET /book/find-by-title/{title}

Description: displays the details of the book with the corresponding title

Path parametres:

title (String) – the title of the book to find

Example request: http://localhost:8080/book/find-by-title/error

Response (200 OK):


```
[
    {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": true
    }
]
```


Response (500 internal server error- if the string provided n'est inclue dans aucun titre des books de la database):


```
{
    "timestamp": "2025-02-03T10:33:35.515+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/find-by-title/mathetiques"
}
```


- GET /book/find-by-category/{category}

Description: displays the details of the books in the given category

Path parametres:

category (Enum Category) – the category dont on veut display les books

Example request: http://localhost:8080/book/find-by-category/ELECTRIC_CIRCUITS

Response (200 OK):


```
[
    {
        "id": 9,
        "title": "Circuits electriques- EECE 210",
        "isbn": "349 230 938",
```

```
        "category": "ELECTRIC_CIRCUITS",
        "availability": true
    },
    {
        "id": 10,
        "title": "Electric circuits- EECE 290",
        "isbn": "345 945 932",
        "category": "ELECTRIC_CIRCUITS",
        "availability": false
    }
]
```


Response (500 internal server error- if the category isn't any of the enums):


```
{
    "timestamp": "2025-02-03T10:37:31.372+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/book/find-by-category/mathetiques"
}
```


- Delete /by-id/{id}

Description: deletes the book with the given ID from the database

Path parametres:
Id (Integer): the ID of the book that we want to delete
Example request: http://localhost:8080/book/by-id/7

Response (200 OK):
Book with id 7 deleted

Response (500 Internal Server Error if the book with the corresponding ID isn't found):


```
{
    "timestamp": "2025-02-03T10:43:04.687+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/book/by-id/7"
}
```


3. Author (AuthorController)

- Post /author/create

Description: creates an author object with the attributes given in the request body, adds it to the database, and returns an author response DTO.
Request(Body):

```
{
    "name": "Rafic",
    "biography": "22 yo inj fin student a l'epfl",
    "age": 22

}
```

Response (200 OK):

```
{
    "id": 71,
    "name": "Rafic",
    "biography": "22 yo inj fin student a l'epfl"
}
```

- GET /author/view/{id}

Description: displays les details du author au ID correspondant

Path parametres:

Id (integer) – le id du author a retrieve

Example request: http://localhost:8080/author/view/5

Response (200 OK):

```
{
    "id": 5,
    "name": "Mary Abou Rjeily",
    "biography": "electrical engineering a l'AUB, year 1. very determined and hard
working"
}
```

Response (500 internal server error- si le ID provided ne correspond a aucun des authors):

```
{
    "timestamp": "2025-02-03T10:55:21.327+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author/view/500"
}
```

- GET /get-all-authors

Description: displays all authors in the data base
Example request: http://localhost:8080/author/get-all-authors

Response (200 OK):


```
[
    {
        "id": 3,
        "name": "Ahmad Abdallah",
        "biography": "Ne le 21 decembre 1973. Etudes en telecommunication a BAU.
Currently CEO de Atom a Myanmar. Such an inspiring personne. "
    },
    {
        "id": 2,
        "name": "Rafic Abdallah",
        "biography": "master de financial engineering a l'EPFL"
    },
    {
        "id": 4,
        "name": "Rahil Chalak",
        "biography": "electrical engineering a l'AUB"
    },
    {
        "id": 5,
        "name": "Mary Abou Rjeily",
        "biography": "electrical engineering a l'AUB, year 1. very determined and hard
working"
    },
    {
        "id": 48,
        "name": "George C. Clark",
        "biography": null
    },
    {
        "id": 50,
        "name": "George C. Clark Jr.",
        "biography": null
    },
    {
        "id": 52,
        "name": "J. Bibb Cain",
        "biography": null
    },
    {
        "id": 71,
```

```
        "name": "Rafic",
        "biography": "22 yo inj fin student a l'epfl"
    },
]
```


- Patch /update/{id}

Description: updates the biography of the author with the given id
Path parametres:
Id (Integer): the ID of the author whose biography we want to update

Example Request: http://localhost:8080/author/update/48
Request Body:

```
ingenieur et poete
```

Response (200 OK):


```
{
    "id": 48,
    "name": "George C. Clark",
    "biography": "ingenieur et poete"
}
```


Response (500 Internal Server Error, if the id in the path doesn't correspond to any author):


```
{
    "timestamp": "2025-02-03T11:02:08.033+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author/update/480"
}
```


- Delete /by-id/{id}

Description: deletes the author with the corresponding id from the database
Path parametres:
Id (Integer): the ID of the author that we want to delete from the database
Example Request: http://localhost:8080/author/by-id/50

Response (200 OK):
Author with id 50 deleted

4. AuthorBook (AuthorBookController)

- POST /author-book/add-author-book

Description: creates an authorBook entity from a request DTO and returns the corresponding response DTO

Request Body:


```
{
    "authorId":4 ,
    "bookId": 8
}
```


Response (200 OK)


```
{
    "id": 73,
    "author": {
        "id": 4,
        "name": "Rahil Chalak",
        "biography": "electrical engineering a l'AUB"
    },
    "book": {
        "id": 8,
        "title": "L'ecole de la guerre",
        "isbn": "239 234 023",
        "category": "POLITICS",
        "availability": true
    }
}
```


- POST /author-book/ add-book-to-author/{authorId}/{bookId}

Description: creates an authorBook entity by adding a book to the list of the author's books

Path parametres:

authorId (long): the author's id

bookId (long): the book's id

Example request: http://localhost:8080/author-book/add-book-to-author/5/10

Response (200 OK):


```
{
    "id": 75,
    "author": {
        "id": 5,
        "name": "Mary Abou Rjeily",
        "biography": "electrical engineering a l'AUB, year 1. very determined and hard working"
    },
    "book": {
        "id": 10,
        "title": "Electric circuits- EECE 290",
        "isbn": "345 945 932",
```

```
        "category": "ELECTRIC_CIRCUITS",
        "availability": false
    }
}
```


Response (500 Internal Server Error- if one of the id's doesn't match to any author or book):


```
{
    "timestamp": "2025-02-03T12:51:17.772+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author-book/add-book-to-author/500/10"
}
```


- POST /author-book/ add-author-to-Book/{bookId}/{authorId}

Description: creates an authorBook entity by an author to the list of a certain book's authors

Path parametres:
authorId (long): the author's id
bookId (long): the book's id
Example request: http://localhost:8080/author-book/add-author-to-Book/10/2
Response (200 OK):


```
{
    "id": 76,
    "author": {
        "id": 2,
        "name": "Rafic Abdallah",
        "biography": "master de financial engineering a l'EPFL"
    },
    "book": {
        "id": 10,
        "title": "Electric circuits- EECE 290",
        "isbn": "345 945 932",
        "category": "ELECTRIC_CIRCUITS",
        "availability": false
    }
}
```


Response (500 Internal Server Error- if one of the id's doesn't match to any author or book):


```
{
    "timestamp": "2025-02-03T12:55:09.836+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author-book/add-author-to-Book/100/2"
}
```

- GET /get-all-books-written-by/{authorId}

Description: displays a list of all books written by an author, given the author's ID
Path parametres:
authorId (long): the ID of the author whose list of books we want to display
Example request: http://localhost:8080/author-book/get-all-books-written-by/3

Response (500 Internal Server Error):


```
{
    "timestamp": "2025-02-03T13:00:42.566+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author-book/get-all-books-written-by/3"
}
```


EntityNotFoundException: No books written by the author

Example request: http://localhost:8080/author-book/get-all-books-written-by/5
Response (200 OK):


```
[
    {
        "id": 10,
        "title": "Electric circuits- EECE 290",
        "isbn": "345 945 932",
        "category": "ELECTRIC_CIRCUITS",
        "availability": false
    }
]
```


- GET /author-book /get-all-authors-of/{bookId}

Description: returns all authors of a given book
Example request: http://localhost:8080/author-book/get-all-authors-of/66

Response(200 OK)


```
[
    {
        "id": 50,
        "name": "George C. Clark Jr.",
        "biography": null
    },
    {
        "id": 52,
        "name": "J. Bibb Cain",
        "biography": null
    },
    {
        "id": 48,
        "name": "George C. Clark",
```

```
        "biography": "ingenieur et poete"
    }
]
```


- GET /author-book/all

Description: displays all AuthorBook entities in the database
Response (200 OK):


```
[
    {
        "id": 67,
        "author": {
            "id": 48,
            "name": "George C. Clark",
            "biography": "ingenieur et poete"
        },
        "book": {
            "id": 66,
            "title": "Error-correction coding for digital communications",
            "isbn": "0306406152",
            "category": "FICTION",
            "availability": true
        }
    },
    {
        "id": 68,
        "author": {
            "id": 50,
            "name": "George C. Clark Jr.",
            "biography": null
        },
        "book": {
            "id": 66,
            "title": "Error-correction coding for digital communications",
            "isbn": "0306406152",
            "category": "FICTION",
            "availability": true
        }
    },
    {
        "id": 69,
        "author": {
            "id": 52,
            "name": "J. Bibb Cain",
            "biography": null
        },
        "book": {
```

```
            "id": 66,
            "title": "Error-correction coding for digital communications",
            "isbn": "0306406152",
            "category": "FICTION",
            "availability": true
        }
    },
    {
        "id": 73,
        "author": {
            "id": 4,
            "name": "Rahil Chalak",
            "biography": "electrical engineering a l'AUB"
        },
        "book": {
            "id": 8,
            "title": "L'ecole de la guerre",
            "isbn": "239 234 023",
            "category": "POLITICS",
            "availability": true
        }
    },
    {
        "id": 74,
        "author": {
            "id": 4,
            "name": "Rahil Chalak",
            "biography": "electrical engineering a l'AUB"
        },
        "book": {
            "id": 8,
            "title": "L'ecole de la guerre",
            "isbn": "239 234 023",
            "category": "POLITICS",
            "availability": true
        }
    },
    {
        "id": 75,
        "author": {
            "id": 5,
            "name": "Mary Abou Rjeily",
            "biography": "electrical engineering a l'AUB, year 1. very determined and
hard working"
        },
        "book": {
            "id": 10,
            "title": "Electric circuits- EECE 290",
```

```
            "isbn": "345 945 932",
            "category": "ELECTRIC_CIRCUITS",
            "availability": false
        }
    },
    {
        "id": 76,
        "author": {
            "id": 2,
            "name": "Rafic Abdallah",
            "biography": "master de financial engineering a l'EPFL"
        },
        "book": {
            "id": 10,
            "title": "Electric circuits- EECE 290",
            "isbn": "345 945 932",
            "category": "ELECTRIC_CIRCUITS",
            "availability": false
        }
    }
]
```


- DELETE /{authorBookId}

Description: deletes the authorBook entity with corresponding ID
Request example: http://localhost:8080/author-book/73
Response (200 OK):
AuthorBook with id: 73 deleted successfully

Response (500 Internal Server Error- if the ID isn't found in the database):


```
{
    "timestamp": "2025-02-03T13:09:25.228+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/author-book/73"
}
```


5. Borrowers (BorrowersController)

- POST /borrowers/create

Description: create a new borrowers entity and return a borrowers response DTO
Request Body:


```
{
    "name": "raf",
    "email": "abdallahraf33@gmail.com",
    "phoneNumber": "71823928"
}
```

Response(200 OK):


```
{
    "id": 78,
    "name": "raf",
    "phoneNumber": "71823928"
}
```


Response (400 Bad Request- if the email format doesn't match the required format):


```
{
    "timestamp": "2025-02-03T13:39:22.810+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/borrowers/create"
}
```


- Get /borrowers/{borrowerId}

Description: displays the details of the borrower with the corresponding id
Path parameter:
borrowerId (long): the ID of the borrower whose details we want to display

Request example: http://localhost:8080/borrowers/77

Response (200 OK):


```
[
    {
        "id": 13,
        "name": "Nadine el Kara",
        "phoneNumber": "+961 32 568 358",
        "email": "nrk@mail.aub.edu"
    },
    {
        "id": 12,
        "name": "Clara Zablit",
        "phoneNumber": "+961 81 853 049",
        "email": "caz@mail.aub.edu"
    },
    {
        "id": 14,
        "name": "Serena Nacouzi",
        "phoneNumber": "+961 33 092 345",
        "email": "sen@mail.aub.edu"
    },
    {
        "id": 29,
        "name": "hicham chalak",
        "phoneNumber": "+961 78 294 029",
```

```
        "email": "hlk@mail.aub.edu"
    },
    {
        "id": 77,
        "name": "leen",
        "phoneNumber": "81853207",
        "email": "abdallahleen33@gmail.com"
    },
    {
        "id": 78,
        "name": "raf",
        "phoneNumber": "71823928",
        "email": "abdallahraf33@gmail.com"
    }
]
```


- Patch /borrowers/ change-email/ {borrowerId}

Description: changes the borrower's email with the corresponding ID
Path parameter:
borrowerId (long): the ID of the borrower.
Request Body:

```
nrk@gmail.com
```

Request example: http://localhost:8080/borrowers/change-email/13
Response (200 OK):


```
{
    "id": 13,
    "name": "Nadine el Kara",
    "phoneNumber": "+961 32 568 358",
    "email": "nrk@gmail.com"
}
```


- Patch borrowers/change-phone-number/ {borrowerId}

Description: changes the phone number of the corresponding borrower

Path parameter:
borrowerId (long): the ID of the borrower.

Request Body:

```
03393419
```

Request example: http://localhost:8080/borrowers/change-phone-number/13
Response (200 OK):


```
{
    "id": 13,
```

```
    "name": "Nadine el Kara",
    "phoneNumber": "03393419",
    "email": "nrk@gmail.com"
}
```


- Delete /borrowers/delete/{borrowerId}

Description: deletes the borrower entity with matching ID from the database
Path parameter:
borrowerId (long): the ID of the borrower we want to delete
Request example: http://localhost:8080/borrowers/delete/12\

Response (200 OK):
Borrower with id 12 deleted

6. Borrowing Transactions (BorrowingTransactionsController)

- Post /borrowing-transactions/create

Description: creates a new borrowing transaction (borrow, return etc).
Request body- borrowing a book:


```
{
    "status": "BORROW",
    "bookId": 66,
    "borrowerId": 14

}
```


Response (200 OK):


```
{
    "id": 79,
    "borrowDate": "2025-02-03",
    "returnDate": null,
    "status": "BORROW",
    "book": {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": false
    },
    "borrower": {
        "id": 14,
        "name": "Serena Nacouzi",
        "phoneNumber": "+961 33 092 345",
        "email": "sen@mail.aub.edu"
    }
}
```

Request Body- borrowing a book that isn't available:


```
{
    "status": "BORROW",
    "bookId": 66,
    "borrowerId": 29

}
```


Response (500 Internal Error- EntityNotFoundException: book is not available when trying to borrow an unavailable book):


```
{
    "timestamp": "2025-02-03T14:29:47.876+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/borrowing-transactions/create"
}
```


Request Body- returning a book:


```
{
    "status": "RETURN",
    "bookId": 66,
    "borrowerId": 14

}
```


Response (200 OK):


```
{
    "id": 79,
    "borrowDate": "2025-02-03",
    "returnDate": "2025-02-03",
    "status": "RETURN",
    "book": {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": true
    },
    "borrower": {
        "id": 14,
        "name": "Serena Nacouzi",
        "phoneNumber": "+961 33 092 345",
        "email": "sen@mail.aub.edu"
    }
}
```

Request Body- borrowing a book after it has been returned:


```
{
    "status": "BORROW",
    "bookId": 66,
    "borrowerId": 29

}
```


Response (200 OK):


```
{
    "id": 80,
    "borrowDate": "2025-02-03",
    "returnDate": null,
    "status": "BORROW",
    "book": {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": false
    },
    "borrower": {
        "id": 29,
        "name": "hicham chalak",
        "phoneNumber": "+961 78 294 029",
        "email": "hlk@mail.aub.edu"
    }
}
```


Request Body- invalid status:


```
{
    "status": "borroww",
    "bookId": 66,
    "borrowerId": 29

}
```


Response (400 Bad Request):


```
{
    "timestamp": "2025-02-03T14:34:10.776+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/borrowing-transactions/create"
}
```


- GET /borrowing-transaction/all-books-borrowed-by-user/{userId}

Description: returns a list of all books ever borrowed by a certain user
Path parameter:
userId (long): the user ID

Request example: http://localhost:8080/borrowing-transactions/all-books-borrowed-by-user/14

Response (200 OK):


```
[
    {
        "id": 66,
        "title": "Error-correction coding for digital communications",
        "isbn": "0306406152",
        "category": "FICTION",
        "availability": false
    }
]
```


- GET /borrowing-transaction/all-borrowers-of-book/{bookId}

Description: returns a list of all users who borrowed a certain book

Path parameter:
bookId (long): the book ID
Request Example: http://localhost:8080/borrowing-transactions/all-borrowers-of-book/66

Response (200 OK):


```
[
    {
        "id": 14,
        "name": "Serena Nacouzi",
        "phoneNumber": "+961 33 092 345",
        "email": "sen@mail.aub.edu"
    },
    {
        "id": 29,
        "name": "hicham chalak",
        "phoneNumber": "+961 78 294 029",
        "email": "hlk@mail.aub.edu"
    }
]
```