

Universitatea Transilvania din Braşov,  
Facultatea de Matematică şi Informatică

# **DOCUMENTAȚIA PROIECTULUI**

## **ReadingTracker**

**Studenti**

Vaida Raluca

Iacob Giulia

## CUPRINS

<b>1) Prezentarea proiectului .....</b>	<b>3</b>
<b>2) Tehnologiile folosite .....</b>	<b>4</b>
<b>3) Schema bazei de date .....</b>	<b>4</b>
<b>4) Prezentarea API-ului .....</b>	<b>6</b>
<b>5) Prezentarea utilizării aplicaţiei .....</b>	<b>14</b>
<b>6) Concluzii şi contribuţii .....</b>	<b>15</b>
<b>7) Link GitHub .....</b>	<b>15</b>

## **Prezentarea aplicaţiei**

Proiectul implementat constituie o aplicaţie .NET al cărei scop o reprezintă gestionarea informaţiilor despre o colecţie de cărţi şi despre utilizatorii care le citesc, aceştia având posibilitatea de a se conecta prin intermediul unui email şi al unei parole.

### **Ce probleme rezolvă aplicaţia ?**

- Evidenţa cărţilor citite de un utilizator
- Înregistrarea şi autentificarea rapidă a utilizatorilor
- Posibilitatea de a adăuga sau de a şterge o carte din colecţie

Proiectul este organizat pe o arhitectură cu mai multe straturi, având trei componente principale, şi anume:

#### **1) ReadingTracker.Api**

Expune endpoint-urile REST, controlează fluxul de date, conţine controllerele (pentru interacţiunea cu utilizatorii, gestionarea lecturilor, GET/POST/PUT/DELETE pentru cărţi şi cititori) şi middlewares.

#### **2) ReadingTracker.Core**

Conţine logica şi modelele de bază, fiind prezente Entităţile, DTO-urile, Interfaces, Services şi Tratarea excepţiilor.

### **3) ReadingTracker.Database**

Conţine clasa DbContext, configuraţiile pentru Entity Framework Core şi implementările repository-urilor.

## **Tehnologiile folosite**

În vederea implementării aplicaţiei backend, principalele tehnologii de care ne-am folosit au fost ASP.NET Core, Entity Framework Core şi SQL Server pentru lucrul cu baza de date, JWT (JSON Web Token) pentru autentificare şi Swagger pentru testarea API-ului.

## **Baza de date**

Baza noastră de date cuprinde trei tabele, şi anume:

### **1) Books**

Conţine detalii cu privire la cărţile din colecţie. Câmpurile tabelului sunt:

- ✓ Id – cheie primară
- ✓ ISBN
- ✓ Name
- ✓ Author
- ✓ Genre

## 2) Readers

Conţine detalii cu privire la utilizatori, adică la cititori. Câmpurile tabelului sunt:

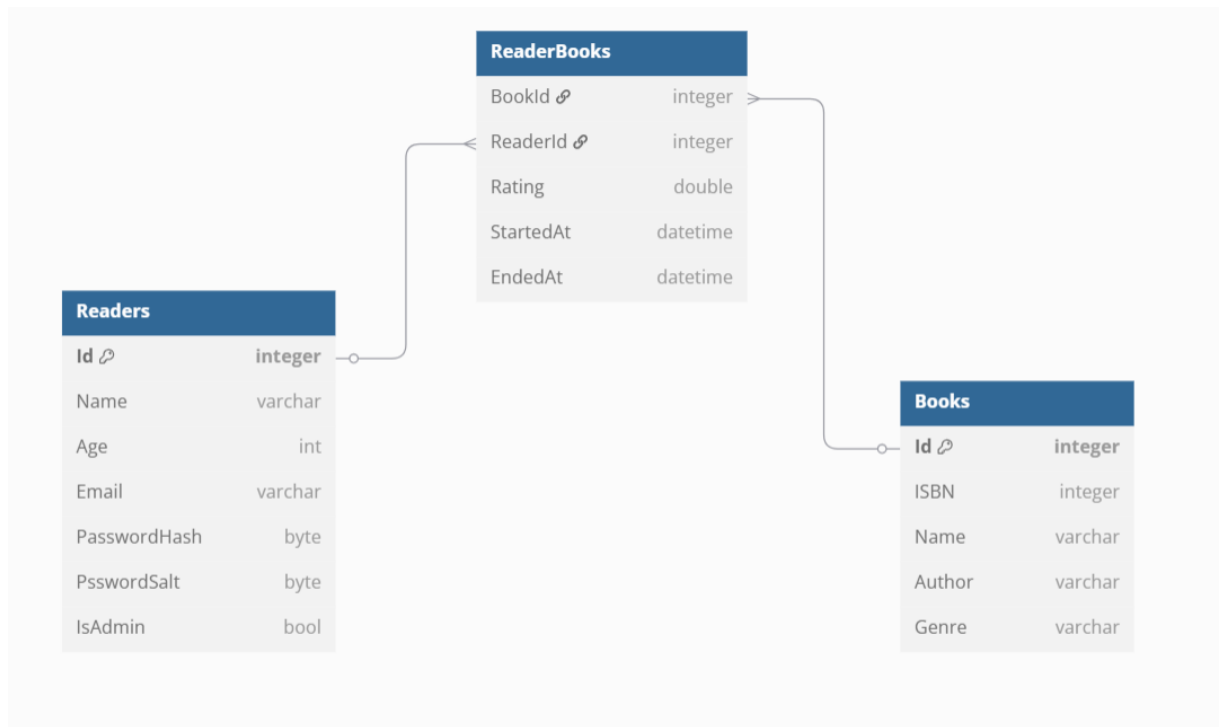
- ✓ Id – cheie primară
- ✓ Name
- ✓ Age
- ✓ Email
- ✓ PasswordHash
- ✓ PasswordSalt
- ✓ IsAdim

Între aceste două tabele se stabileşte o relaţie de many-to-many (un utilizator poate să citească mai multe cărţi şi o carte poate să fie citită de mai mulţi utilizatori), prin urmare am adăugat o tabelă intermediară:

### **ReaderBooks**

- ✓ BookId – cheie primară
- ✓ ReaderId – cheie primară
- ✓ Rating
- ✓ StartedAt
- ✓ EndedAt

## Diagrama bazei de date



## Prezentarea API-ului

### Pentru tabela Books

Avem implementate 3 operații GET. Putem să selectăm o carte în funcție de Id-ul ei, în funcție de nume și putem să sortăm cărțile crescător/descrescător în funcție de un anumit câmp.

# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică

## Exemplu – sortare de cărţi crescător după nume

```
curl -X 'GET' \
  'https://localhost:5192/api/Books?sortBy=name&order=asc' \
  -H 'accept: text/plain'
```

Request URL

`https://localhost:5192/api/Books?sortBy=name&order=asc`

Server response

Code	Details
200	<p>Response body</p> <pre>[   {     "id": 6,     "isbn": "1235",     "name": "Harry Pottet",     "author": "JK Rowling",     "genre": "Fantasy"   },   {     "id": 5,     "isbn": "1234",     "name": "Lord of The Rings",     "author": "JRR Tolkien",     "genre": "Fantasy"   },   {     "id": 10,     "isbn": "1230",     "name": "None of This Is True",     "author": "Lisa Jewell",     "genre": "Thriller"   },   {     "id": 0,     "isbn": "1238",     "name": "Recursion",     "author": "Blake Crouch",     "genre": "Science Fiction"   } ]</pre> <p>Download</p>

## Exemplu – selectarea cărţii cu Id-ul egal cu 5

```
curl -X 'GET' \
  'https://localhost:5192/api/Books/id/5' \
  -H 'accept: text/plain'
```

Request URL

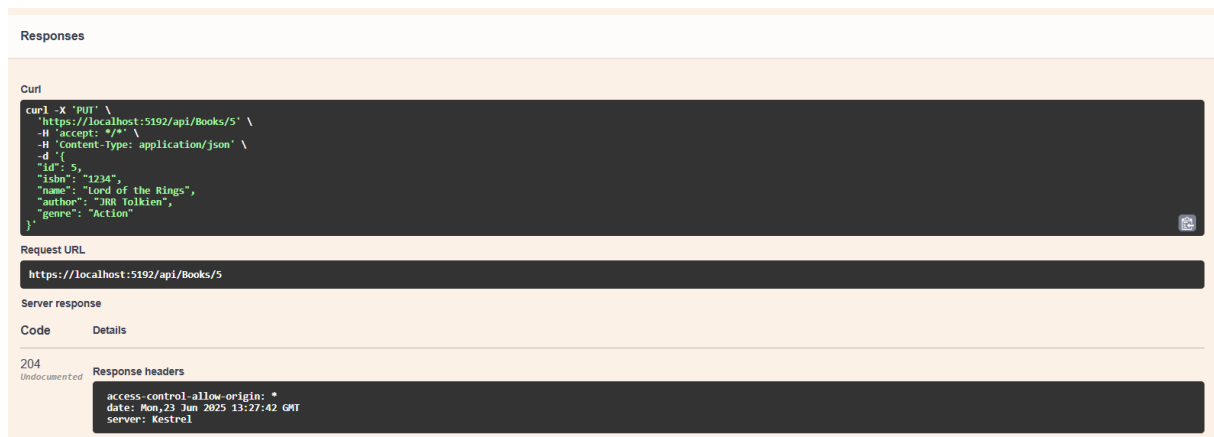
`https://localhost:5192/api/Books/id/5`

Server response

Code	Details
200	<p>Response body</p> <pre>{   "id": 5,   "isbn": "1234",   "name": "Lord of The Rings",   "author": "JRR Tolkien",   "genre": "Fantasy" }</pre> <p>Download</p> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon, 23 Jun 2025 13:23:14 GMT server: Kestrel</pre>

Universitatea Transilvania din Braşov,  
Facultatea de Matematică şi Informatică

S-a implementat operaţia de PUT cu care putem face update la campurile unei înregistrări. Spre exemplu, am modificat înregistrarea cu Id-ul 5, schimbând genul cărţii de la fantasy la action.



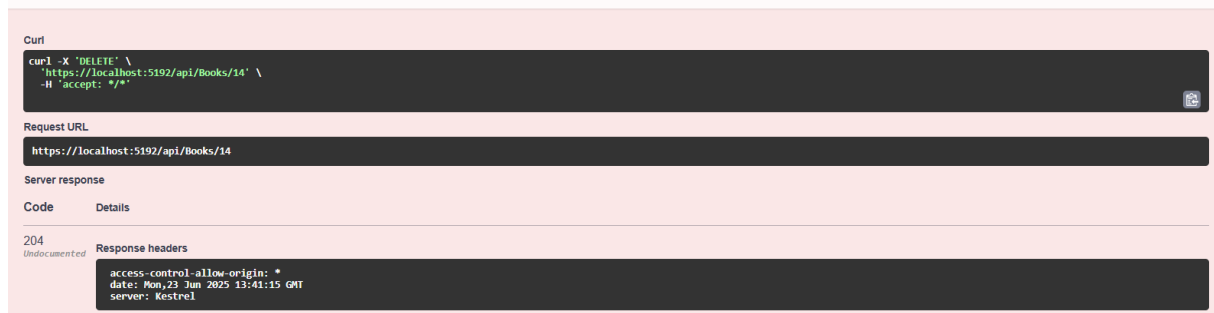
S-a implementat operaţia POST cu care putem să adăugăm o înregistrare nouă printre cele deja prezente. Spre exemplu, am adăugat romanul „The Shining” de Stephen King la listă.





# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică

S-a implementat operaţia DELETE cu care putem şterge una din cărţile existente în listă. **Spre exemplu, am şters cartea nou adăugată la exemplul precedent.**



```
Curl
curl -X 'DELETE' \
  'https://localhost:5192/api/Books/14' \
  -H 'accept: */*'

Request URL
https://localhost:5192/api/Books/14

Server response
Code    Details
204
Undocumented
Response headers
access-control-allow-origin: *
date: Mon, 23 Jun 2025 13:41:15 GMT
server: Kestrel
```

## Pentru tabela Readers

Am implementat 2 operaţii GET. Cu prima putem afişa lista de cititori pe care îi putem sorta crescător sau descrescător după un anumit criteriu. Cu a doua operaţia putem selecta un utilizator în funcţie de Id-ul său.

### Exemplu – selectarea utilizatorului cu Id-ul 20



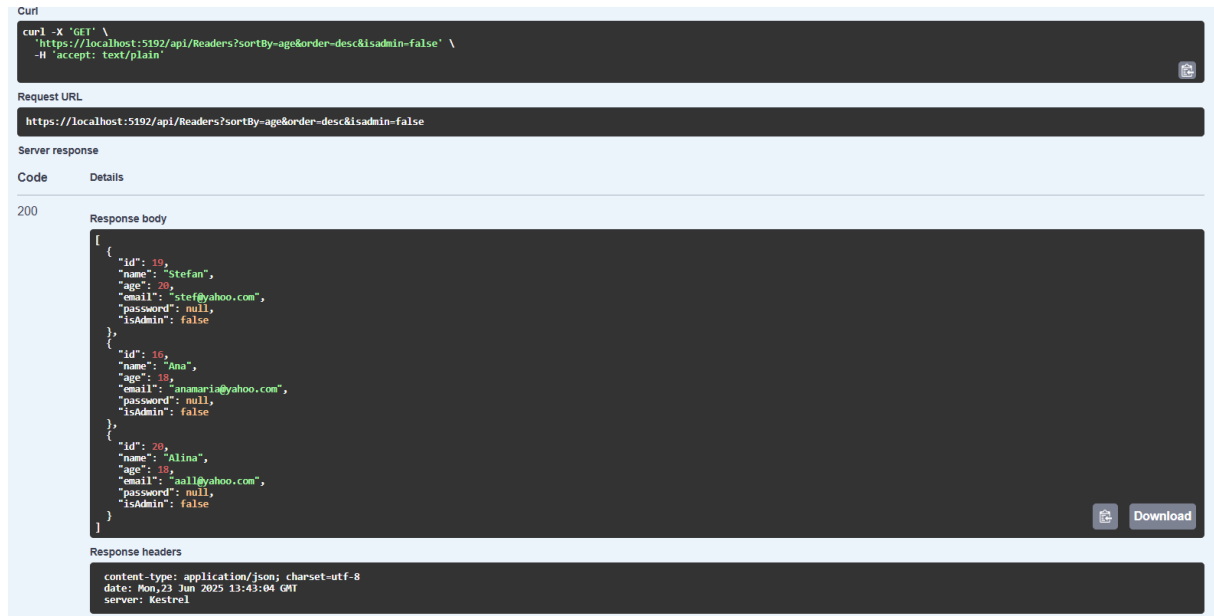
```
Curl
curl -X 'GET' \
  'https://localhost:5192/api/Readers/20' \
  -H 'accept: text/plain'

Request URL
https://localhost:5192/api/Readers/20

Server response
Code    Details
200
Response body
{
  "id": 20,
  "name": "Alina",
  "age": 18,
  "email": "aall@yahoo.com",
  "password": null,
  "isAdmin": false
}
```

# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică

## Exemplu – lista sortată descrescător după vârstă a utilizatorilor care nu au rol de admin



Curl

```
curl -X 'GET' \
  'https://localhost:5192/api/Readers?sortBy=age&order=desc&isAdmin=false' \
  -H 'accept: text/plain'
```

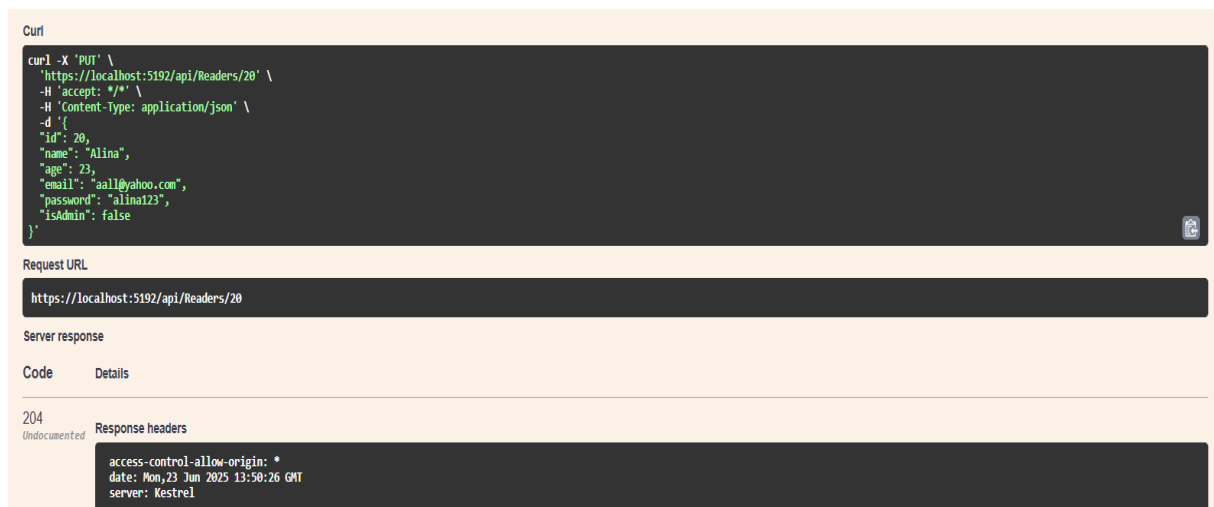
Request URL

https://localhost:5192/api/Readers?sortBy=age&order=desc&isAdmin=false

Server response

Code	Details
200	<p>Response body</p> <pre>[   {     "id": 19,     "name": "Stefan",     "age": 20,     "email": "stef@yahoo.com",     "password": null,     "isAdmin": false   },   {     "id": 16,     "name": "Ana",     "age": 18,     "email": "anamaria@yahoo.com",     "password": null,     "isAdmin": false   },   {     "id": 20,     "name": "Alina",     "age": 18,     "email": "aall@yahoo.com",     "password": null,     "isAdmin": false   } ]</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon, 23 Jun 2025 13:43:04 GMT server: Kestrel</pre>

Cu operaţia de PUT putem să actualizăm informaţiile legate de un anumit utilizator. Spre exemplu, pentru utilizatorul cu Id-ul 20 voi schimba vârsta de la 18 ani la 23.



Curl

```
curl -X 'PUT' \
  'https://localhost:5192/api/Readers/20' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 20,
    "name": "Alina",
    "age": 23,
    "email": "aall@yahoo.com",
    "password": "alina123",
    "isAdmin": false
  }'
```

Request URL

https://localhost:5192/api/Readers/20

Server response

Code	Details
204	<p>Response headers</p> <pre>access-control-allow-origin: * date: Mon, 23 Jun 2025 13:50:26 GMT server: Kestrel</pre>

S-au implementat 2 operații de POST. Prima dintre ele e pentru înregistrarea unui utilizator, iar a doua pentru logarea în cont al acestuia.

### Exemplu – înregistrarea unui nou utilizator

The screenshot shows a REST client interface with the following details:

- Curl:** A curl command for a POST request to `https://localhost:5192/api/Readers/register`. The headers are `-H 'accept: text/plain'` and `-H 'Content-type: application/json'`. The body is a JSON object: `{ "name": "Giulia", "age": 28, "email": "giuliaciob@yahoo.com", "password": "giulia123", "isAdmin": true }`.
- Request URL:** `https://localhost:5192/api/Readers/register`
- Server response:** A 200 status code.
- Response body:** A JSON object containing a token and user information: `{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2x1IjoiaWltcmVudC1jYySwIGl0IiwiaWF0IjoiZm1lbnQwLjE0eSImInhdCGRWTkMDYANjgweSIAW1joiOmFja29vZC1mf1ZTGIKZy6250Zn5kInB.zeFd3pqIPcQwepTnbGfMa1PbpTq5Smr54frce09Y", "email": "giuliaciob@yahoo.com", "name": "Giulia" }`
- Response headers:** `access-control-allow-origin: *  
content-type: application/json; charset=utf-8  
date: Mon, 23 Jun 2025 13:54:01 GMT  
server: Kestrel`

## Exemplu – logarea unui utilizator

```
Curl curl -X 'POST' \
      'https://localhost:5192/api/Readers/login' \
      -H 'accept: text/plain' \
      -H 'Content-Type: application/json' \
      -d '{
        "email": "giuliaiacob@yahoo.com",
        "password": "giulia123"
      }'
```

Request URL

https://localhost:5192/api/Readers/login

Server response

Code	Details
200	<p>Response body</p> <pre>{   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiaWRTakw1CjE2YySQA0lIiwiaWF0IjoiZTJ1IG1haWVjb2JAcWZOb29tY291IiwibmMiOiJodXRudWJg20TU3LClleWA0Je300Ty9jTSMTcSmlhdCI6MTY0YANjk1NDYxNzIjoiQmFja2Vuc2lmc1ZC1k1KzYyZS02ZSk1bn0uVm1hM3PeAKINPChk5FVG-dG3fJrmQSPqfzX_0t-E",   "email": "giuliaiacob@yahoo.com",   "name": "Giulia" }</pre>

S-a implementat operația DELETE cu care se poate șterge un cont de utilizator. **Spre exemplu, voi aplica DELETE pe contul pe care l-am creat la primul exemplu de POST.**

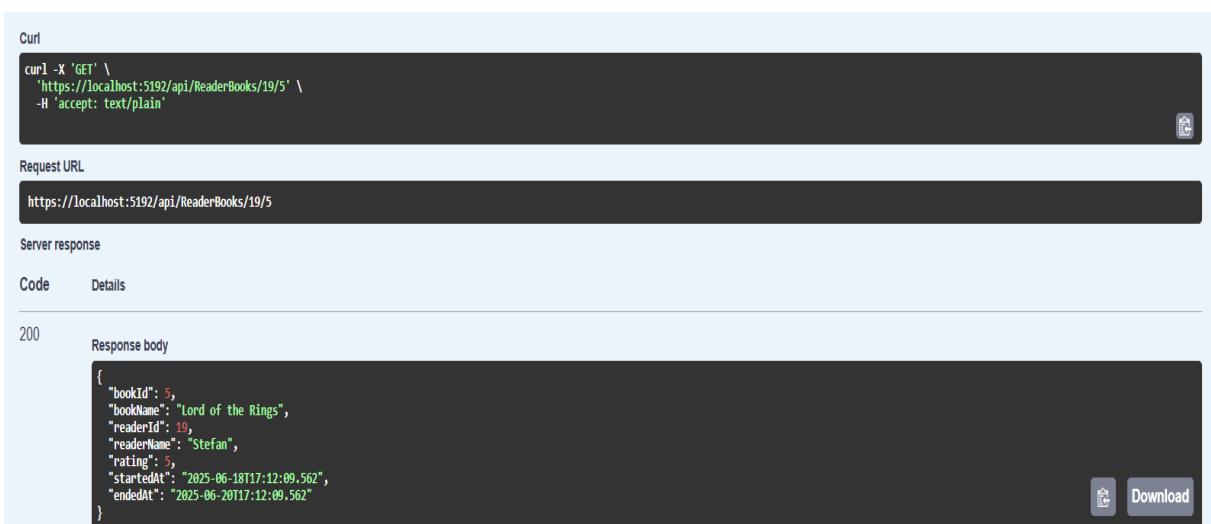
# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică



## Pentru tabela ReaderBooks

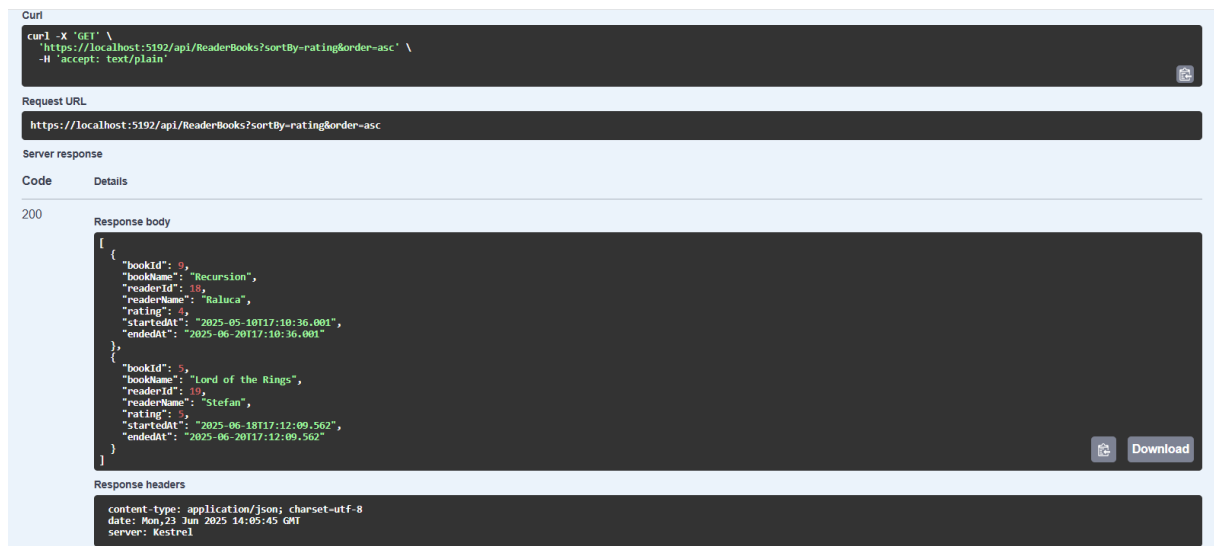
S-au implementat 2 operații de GET. Cu una dintre ele putem sorta înregistrările din tabelă crescător sau descrescător după un anumit criteriu, iar cu cealaltă operație putem selecta o înregistrare în funcție de BookId și ReaderId.

## Exemplu – Selectarea înregistrării cu BookId = 5 și ReaderId = 19



# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică

## Exemplu – sortarea crescătoare a elementelor în funcţie de Rating



The screenshot shows a REST client interface. The 'Curl' tab contains the command: `curl -X 'GET' \ 'https://localhost:5192/api/ReaderBooks?sortBy=rating&order=asc' \ -H 'accept: text/plain'`. The 'Request URL' tab shows: `https://localhost:5192/api/ReaderBooks?sortBy=rating&order=asc`. The 'Server response' tab shows a status code of 200 and a 'Response body' containing a JSON array of two book records. The first record has bookId 9, bookName 'Recursion', readerId 18, readerName 'Raluca', rating 4, and the second record has bookId 5, bookName 'Lord of the Rings', readerId 19, readerName 'Stefan', rating 5. The 'Response headers' tab shows: `content-type: application/json; charset=utf-8`, `date: Mon, 23 Jun 2025 14:05:45 GMT`, and `server: Kestrel`.

```
curl -X 'GET' \
'https://localhost:5192/api/ReaderBooks?sortBy=rating&order=asc' \
-H 'accept: text/plain'
```

Request URL

https://localhost:5192/api/ReaderBooks?sortBy=rating&order=asc

Server response

Code Details

200

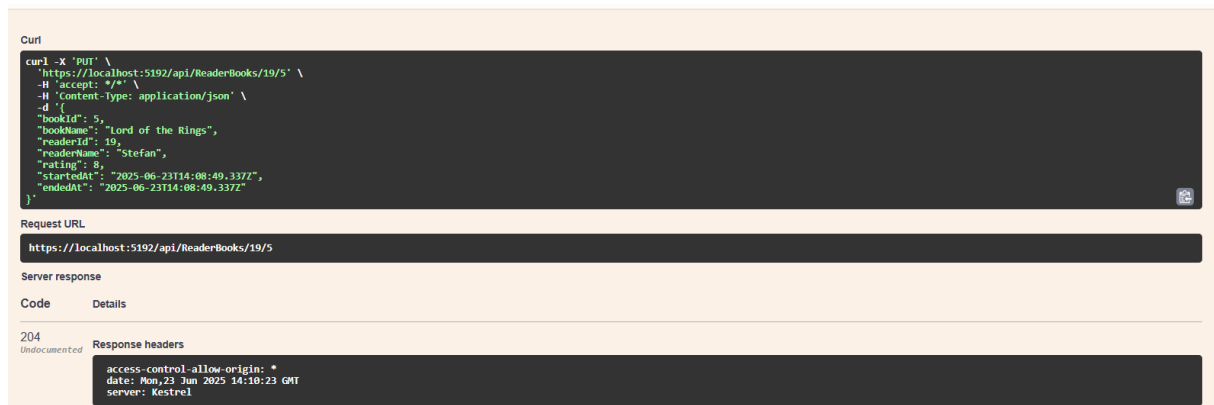
Response body

```
{
  "bookId": 9,
  "bookName": "Recursion",
  "readerId": 18,
  "readerName": "Raluca",
  "rating": 4,
  "startedAt": "2025-05-10T17:10:36.001",
  "endedAt": "2025-06-20T17:10:36.001"
},
{
  "bookId": 5,
  "bookName": "Lord of the Rings",
  "readerId": 19,
  "readerName": "Stefan",
  "rating": 5,
  "startedAt": "2025-06-18T17:12:09.562",
  "endedAt": "2025-06-20T17:12:09.562"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 23 Jun 2025 14:05:45 GMT
server: Kestrel
```

S-a implementat operaţia de PUT cu care putem să actualizăm informaţiile dintr-o anumită înregistrare. Spre exemplu, pentru înregistrarea cu BookId = 5 şi ReaderId = 19 am modificat rating-ul, de la 5 actualizându-l la 8.



The screenshot shows a REST client interface. The 'Curl' tab contains the command: `curl -X 'PUT' \ 'https://localhost:5192/api/ReaderBooks/19/5' \ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d '{ "bookId": 5, "bookName": "Lord of the Rings", "readerId": 19, "readerName": "Stefan", "rating": 8, "startedAt": "2025-06-23T14:08:49.337Z", "endedAt": "2025-06-23T14:08:49.337Z" }'`. The 'Request URL' tab shows: `https://localhost:5192/api/ReaderBooks/19/5`. The 'Server response' tab shows a status code of 204 and 'Response headers' showing: `access-control-allow-origin: *`, `date: Mon, 23 Jun 2025 14:10:23 GMT`, and `server: Kestrel`.

```
curl -X 'PUT' \
'https://localhost:5192/api/ReaderBooks/19/5' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "bookId": 5,
  "bookName": "Lord of the Rings",
  "readerId": 19,
  "readerName": "Stefan",
  "rating": 8,
  "startedAt": "2025-06-23T14:08:49.337Z",
  "endedAt": "2025-06-23T14:08:49.337Z"
}'
```

Request URL

https://localhost:5192/api/ReaderBooks/19/5

Server response

Code Details

204

Response headers

```
access-control-allow-origin: *
date: Mon, 23 Jun 2025 14:10:23 GMT
server: Kestrel
```

S-a implementat operaţia de POST cu care putem să adăugăm o nouă înregistrare în tabelă:

# Universitatea Transilvania din Braşov, Facultatea de Matematică şi Informatică

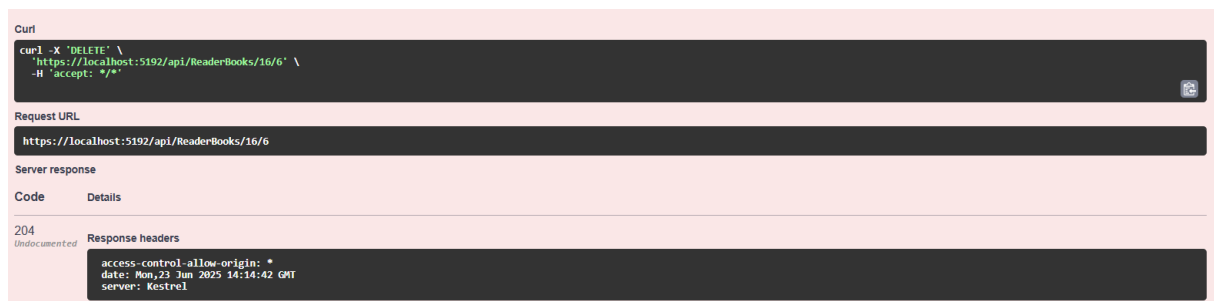


```
Curl
curl -X 'POST' \
  'https://localhost:5192/api/ReaderBooks' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "bookId": 6,
    "bookName": "Harry Pottet",
    "readerId": 16,
    "readerName": "Ana",
    "rating": 10,
    "startedAt": "2025-06-22T14:11:07.177Z",
    "endedAt": "2025-06-23T14:11:07.177Z"
  }'

Request URL
https://localhost:5192/api/ReaderBooks

Server response
Code    Details
201
Undocumented
Response body
{
  "bookId": 6,
  "bookName": "Harry Pottet",
  "readerId": 16,
  "readerName": "Ana",
  "rating": 10,
  "startedAt": "2025-06-22T14:11:07.177Z",
  "endedAt": "2025-06-23T14:11:07.177Z"
}
```

S-a implementat operaţia DELETE cu care se poate şterge o înregistrare din tabelă:



```
Curl
curl -X 'DELETE' \
  'https://localhost:5192/api/ReaderBooks/16/6' \
  -H 'accept: */*'

Request URL
https://localhost:5192/api/ReaderBooks/16/6

Server response
Code    Details
204
Undocumented
Response headers
access-control-allow-origin: *
date: Mon, 23 Jun 2025 14:14:42 GMT
server: Kestrel
```

## Prezentarea utilizării aplicaţiei

Aplicaţia presupune gestiunea unor colecţii de cărţi şi a unei liste de utilizatori sau cititori conectaţi cu email şi parolă. E permisă ştergerea, actualizarea şi selectarea oricărei înregistrări din fiecare tabelă. De menţionat că la momentul creării unui nou cont, parola utilizatorului este hash-uită astfel că nu poate să apară direct cu numele ei în baza de date. De asemenea, în momentul în care se cere efectuarea unei operaţii care necesită existenţa unei anumite valori din baza de date, se aruncă excepţie dacă valoarea nu se găseşte în baza de date. Emailul şi parola sunt, totodată, obligatorii de trecut la înregistrare, astfel apare eroare.

## Concluzii şi contribuţii

- Ne-am familiarizat cu crearea şi construirea unei aplicaţii ASP.NET Core
- Ne-am consolidat cunoştinţele în ceea ce priveşte limbajul C#, lucrul cu bazele de date şi Entity Framework Core + SQL Server
- Ne-am familiarizat cu modul de lucru al lui Swagger
- Ştim cum să proiectăm o bază de date într-un mod corect şi complet
- Am învăţat să folosim Dependency Injection în .NET configurând servicii pe baza de interfeţe pentru a fi injectate automat acolo unde sunt necesare

Vaida Raluca – implementare codului propriu-zis

Iacob Giulia - schema bazei de date, testarea API-ului, documentaţia

## Link GitHub

<https://github.com/VaidaRaluca/ReadingTracker>