

Test rapport Bookfinder API version 0.9.1.1

Översikt över API

API:t för *Bookfinder* möjliggör interaktion med en bokdatabas, hanterar CRUD-operationer (Create, Read, Update, Delete) för böcker och författare.

Det är utvecklat med Node.js och använder Express.js som serverramverk med en MongoDB databas för datalagring. Bas-URL för API:t är **http://localhost:3000/api**, och API:t är för närvarande i version 0.9.1.1, vilket indikerar att det är under aktiv utveckling och kan innefatta beta-funktionalitet.

Testmiljö

Testerna utfördes i en kontrollerad lokal utvecklingsmiljö, vilket gjorde det möjligt att isolera testerna från externa störningar. För att skicka förfrågningar till API:t användes Postman, ett verktyg för API-testning som tillåter detaljerad konfiguration av HTTP-förfrågningar. Miljön inkluderade en lokalt konfigurerad server, optimerad för testfallens krav, och en MongoDB-instans för hantering av databasoperationer.

Testfall

Testfallen strukturerades i två huvudkategorier: *Automatiska Tester* och *Manuella Tester*. Varje kategori designades för att systematiskt validera API:ets funktioner från olika perspektiv, där de automatiska testerna fokuserade på regelmässighet och de manuella testerna på mer djupgående och ad hoc-scenarion.

Utförande av Test

Under testprocessen skickades HTTP-begäranden via Postman baserat på fördefinierade testscenarier. Varje förfrågan loggades med detaljerad information om begärandet och responsen för att underlätta en noggrann analys av API:ets beteende under olika förhållanden. Resultaten verifierades mot förväntade utfall för att säkerställa korrekt funktionalitet och prestanda.

Resultat och Slutsatser

Resultatsektionen presenterar en sammanfattning av testerna, inklusive framgångsrika utföranden och identifierade avvikelser. En detaljerad analys av de insamlade resultaten tillhandahålls för att belysa mönster, prestandaproblem och eventuella buggar i API:et.

Felhantering och Bugs

Detta avsnitt dokumenterar alla identifierade fel under testprocessen. Det inkluderar en diskussion om API:ets svar på felaktiga förfrågningar, exceptionshantering och åtgärder som har tagits för att rätta till identifierade problem.

Rekommendationer och Förbättringar

Baserat på testresultaten och de insamlade datanalisresultaten, erbjuds här konkreta förslag på förbättringar av både kod och funktionalitet. Detta kan inkludera förslag på optimeringar, potentiella säkerhetsförstärkningar och anpassningar för att förbättra användarupplevelsen.

Automatiska Tester

Automatiskt Testfall 1:

Validate GET Books - 200 OK

- **Syfte:** Att verifiera att API returnerar korrekt HTTP-statuskod (200 OK) för en lyckad GET-förfrågan.
- **Utförande:** Skicka en GET-förfrågan till **`http://localhost:3000/api/books?name=Awesome Steel Chair 1`**.
- **Förväntade Resultat:** API:et ska returnera 200 OK med korrekt dataformat (JSON/XML).
- **Faktiska Resultat:** API:et returnerade 200 OK, data i JSON-format.

GET 1. Validate GET Books that the API returns the correct HTTP status code (200 OK)
`http://localhost:3000/api/books?name=Awesome Steel Chair 1`

200 OK 19 ms 412 B

PASS Status code is 200

- **Slutsats:** API:et fungerar som förväntat under standardförhållanden.

Automatiskt Testfall 2:

Verify Data Format (JSON/XML)

- **Syfte:** Att verifiera att API returnerar förväntat dataformat i svaret.
- **Utförande:** Skicka en GET-förfrågan till **`http://localhost:3000/api/books?format=json`**.
- **Förväntade Resultat:** API:et ska returnera data i JSON-format.
- **Faktiska Resultat:** Data korrekt returnerad i JSON-format.

GET 2 Verify that the API returns the expected data format (e.g., JSON, XML) in the response.
`http://localhost:3000/api/books?name=Awesome Steel Chair 1`

200 OK 13 ms 412 B

PASS Status code is 200

PASS Response is in JSON format

- **Slutsats:** API:et stöder korrekt dataformat, verifiering lyckades.

Automatiskt Testfall 3:

Validate 400 Bad Request for Invalid Parameters

- **Syfte:** Att testa API:ets hantering av ogiltiga parametrar och säkerställa att korrekt statuskod (400 Bad Request) returneras.
- **Utförande:** Skicka en GET-förfrågan med ogiltiga parametrar **`http://localhost:3000/api/books?name=!@#`**.
- **Förväntade Resultat:** API:et ska returnera 400 Bad Request.
- **Faktiska Resultat:** 400 Bad Request korrekt mottagen.

GET 3. Validate an invalid requests GET Books that the API returns the correct HTTP status code (400 Bad Request)
`http://localhost:3000/api/books?name=!@#`

400 Bad Request 9 ms 286 B

PASS Status code is 400

- **Slutsats:** API hanterar felaktiga förfrågningar effektivt.

Automatiskt Testfall 4:

Validate GET Books with Specific Filters

- **Syfte:** Att kontrollera att API:et korrekt returnerar data baserat på specifika sökkriterier och filter.
- **Utförande:** Skicka en GET-förfrågan till
`http://localhost:3000/api/books?name=Awesome Steel Chair 1&isbn=3629663257859&price=353&genre=Jazz&releaseDate=2021-08-12`.
- **Förväntade Resultat:** API:et ska returnera data som matchar alla angivna filter.
- **Faktiska Resultat:** Data korrekt returnerad enligt de angivna filtren.

```
GET 4. Validate request GET Books with specific filters or search criteria and checks if the API returns the correct data.
http://localhost:3000/api/books?name=Awesome Steel Chair 1&isbn=3629663257859&price=353&genre=Jazz&releaseDate=2021-08-12 200 OK 16 ms 412 B

PASS Status code is 200
PASS Response is in JSON format
PASS Response correctly filtered
```

- **Slutsats:** API:ets funktionalitet för att hantera flera filter bekräftas, inga problem identifierades.

Automatiskt Testfall 5:

Validate Paginated GET Requests

- **Syfte:** Att verifiera att API:et korrekt returnerar paginerade resultat när stora mängder data efterfrågas.
- **Utförande:** Skicka en GET-förfrågan till
`http://localhost:3000/api/books?page=1&limit=25`.
- **Förväntade Resultat:** API:et ska returnera en delmängd av böckerna, begränsat till 25 poster per sida.
- **Faktiska Resultat:** Korrekt paginerade resultat erhöles.
-

```
GET 5. Validate GET Books that the API returns paginated results when a large number of records are requested.
http://localhost:3000/api/books?page=1&limit=25 200 OK 23 ms 4.175 KB

PASS Status code is 200
PASS Response contains correct number of items
```

- **Slutsats:** API:ets kapacitet att hantera paginerade förfrågningar bekräftas.

Automatiskt Testfall 6: Validate Special Characters and Non-English Text

- **Syfte:** Att testa API:ets förmåga att korrekt hantera specialtecken och icke-engelska texter i indata och returnerade svar.
- **Utförande:** Skicka en GET-förfrågan till **`http://localhost:3000/api/books?name=En bok med ÄÅÖ`**.
- **Förväntade Resultat:** API:et ska korrekt returnera data utan fel, även med specialtecken och icke-engelska tecken.
- **Faktiska Resultat:** API:et hanterade förfrågan utan problem, korrekt data returnerades.

```
GET 6. Validate if the API handles special characters and non-English text correctly in input data and returned responses using an automated testing tool.
http://localhost:3000/api/books?name=En bok med ÄÅÖ 200 OK 12 ms 405 B
PASS Response contains books with special characters the API handle it
```

- **Slutsats:** API:ets robusthet mot olika teckenuppsättningar bekräftas, presterar väl under testade villkor.

Automatiskt Testfall 7:Validate Concurrent Requests

- **Syfte:** Att verifiera att API:et kan hantera flera samtidiga förfrågningar utan att förlora datakonsistens eller prestanda.
- **Utförande:** Simulera flera samtidiga GET-förfrågningar till **`http://localhost:3000/api/books`**.
- **Förväntade Resultat:** API:et ska hantera alla förfrågningar korrekt och returnera konsekventa och korrekta svar.
- **Faktiska Resultat:** Samtliga förfrågningar hanterades effektivt, inga inkonsekvenser i data.

```
GET 7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books 200 OK 24 ms 10.332 KB
PASS Simulate concurrent requests
```

- **Slutsats:** API:ets skalbarhet och förmåga att hantera hög belastning bekräftas.

Automatiskt Testfall 8:

Validate PUT API correctly handles different HTTP methods for each endpoint

- **Syfte:** Att verifiera att API:et korrekt hanterar PUT-förfrågningar och returnerar lämpliga statuskoder och svar för varje endpoint.
- **Utförande:** Skicka en PUT-förfrågan till **http://localhost:3000/api/books/662aa196cdec73d79be79845** med nödvändiga uppdateringsparametrar.
- **Förväntade Resultat:** API:et ska korrekt uppdatera den angivna boken och returnera en lämplig respons med statuskoden 200 OK eller 204 No Content om ingen innehållsuppdatering sker.
- **Faktiska Resultat:** Boken uppdaterades korrekt, och API:et svarade med statuskod 200 OK.

```
PUT 8 Validate PUT API correctly handles different HTTP methods for each endpoint and returns appropriate status codes and responses for each method.
http://localhost:3000/api/books/662aa196cdec73d79be79845 200 OK 64 ms 412 B

PASS Verify data is updated correctly
```

- **Slutsats:** API:et hanterar PUT-förfrågningar effektivt och säkerställer att datan är konsekvent uppdaterad.

Automatiskt Testfall 9:

Validate POST API correctly handles different HTTP methods for each endpoint

- **Syfte:** Att verifiera att API:et korrekt hanterar POST-förfrågningar för att skapa nya poster och returnerar lämpliga statuskoder och svar.
- **Utförande:** Skicka en POST-förfrågan till **http://localhost:3000/api/books** med nödvändiga data för att skapa en ny bok.
- **Förväntade Resultat:** API:et ska acceptera förfrågan, skapa en ny bokpost och returnera statuskoden 201 Created med detaljer om den skapade boken.
- **Faktiska Resultat:** Ny bok skapades framgångsrikt, och API:et returnerade 201 Created med information om boken.

```
POST 9 Validate POST API correctly handles different HTTP methods PUT for each endpoint and returns appropriate status codes and responses for each method.
http://localhost:3000/api/books 201 Created 34 ms 391 B

PASS Response contains newly created book
PASS Response contains newly created
```

- **Slutsats:** API:ets funktion för att hantera POST-förfrågningar verifieras som robust och korrekt.

Automatiskt Testfall 10:

Validate performance test that simulates a large number of users making requests simultaneously

- **Syfte:** Att testa API:ets prestanda under tung belastning genom att simulera ett stort antal samtidiga förfrågningar.
- **Utförande:** Använde verktyg Postman för att skicka samtidiga GET-förfrågningar till **http://localhost:3000/api/books**.
- **Förväntade Resultat:** API:et ska hantera den höga belastningen effektivt utan signifikant prestandaförlust eller fel.

•

- **Faktiska Resultat:** API:et hanterade inte alla förfrågningar korrekt

•

GET 10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load.

http://localhost:3000/api/books

200 OK 23 ms 10.483 KB

FAIL Request failed | AssertionError: Request failed: undefined

•

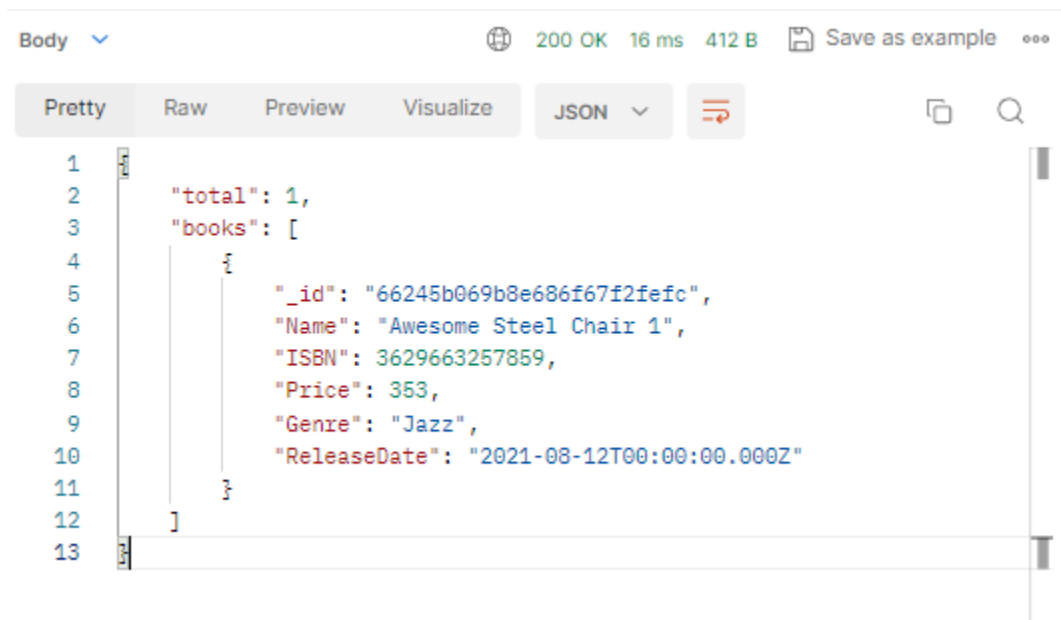
- **Slutsats:** API:ets skalbarhet och förmåga att hantera hög trafik behöver förbättras.

Manuella Testfall

Manuellt Testfall 1:

Validera GET-förfrågningar för böcker - 200 OK

- **Syfte:** Att verifiera att API:et returnerar korrekt HTTP-statuskod (200 OK) för en korrekt formulerad GET-förfrågan.
- **Utförande:**
 1. Skicka en GET-förfrågan till **`http://localhost:3000/api/books?name=Awesome Steel Chair 1`** med nödvändiga parametrar.
 2. Observera svaret från servern.
- **Förväntade Resultat:** API:et ska returnera 200 OK med korrekt dataformat och innehåll som överensstämmer med databasens poster.
- **Faktiska Resultat:**



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

- Status Bar:** 200 OK, 16 ms, 412 B, Save as example, and a menu icon.
- Response Body:** Pretty, Raw, Preview, Visualize, JSON, and a refresh icon.
- JSON Response:**

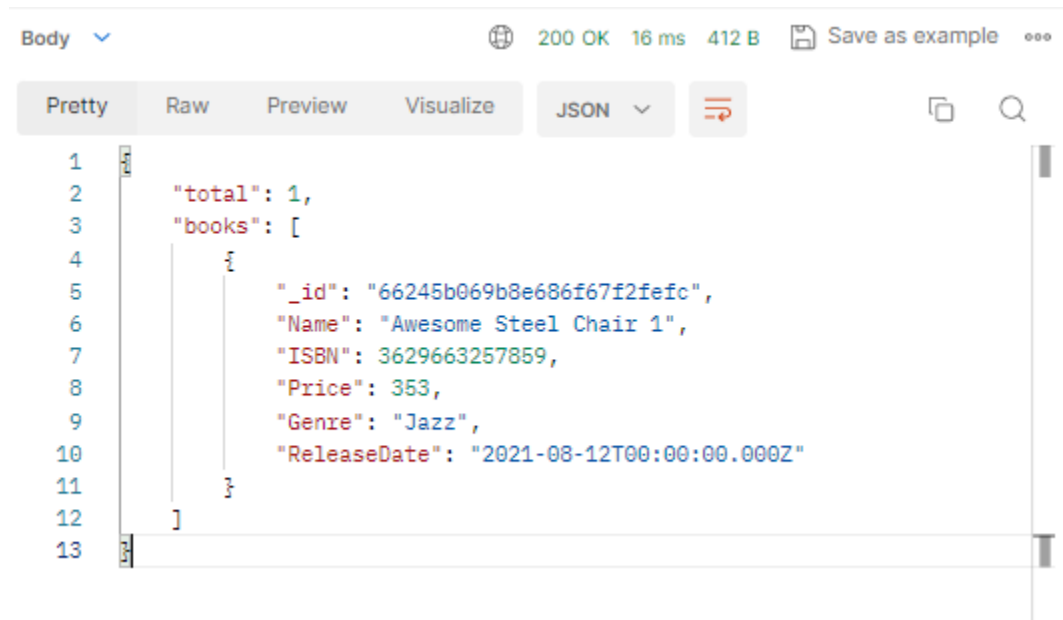
```
1 {
2   "total": 1,
3   "books": [
4     {
5       "_id": "66245b069b8e686f67f2f2f2f2f2f2f2",
6       "Name": "Awesome Steel Chair 1",
7       "ISBN": 3629663257859,
8       "Price": 353,
9       "Genre": "Jazz",
10      "ReleaseDate": "2021-08-12T00:00:00.000Z"
11    }
12  ]
13 }
```

- **Slutsats:** Utvärdera om resultaten överensstämmer med de förväntade resultaten och dokumentera eventuella avvikelser.

Manuellt Testfall 2:

Verifiera GET förfrågan returnerar i förväntat dataformat Format (JSON/XML)

- **Syfte:** Detta test syftar till att verifiera att API:et korrekt hanterar GET-förfrågningar och returnerar data i de specificerade formaten (t.ex. JSON eller XML), i enlighet med API-specifikationerna. Detta är avgörande för att säkerställa att API:et kan integreras smidigt med olika klienter som förväntar sig specifika dataformat.
- **Steg för Steg Utförande:**
 1. Skicka en GET-förfrågan till `http://localhost:3000/api/books?name=Awesome Steel Chair 1` med nödvändiga parametrar. Specificera att svaret önskas i JSON-format genom att inkludera en lämplig 'Accept' header i förfrågan.
 2. Observera svaret från servern, särskilt dataformatet av svaret.
- **Förväntade Resultat:** API:et ska returnera ett svar som överensstämmer med de förväntningar som definieras i testfallet. Det inkluderar:
 - Korrekt statuskod (t.ex., 200 OK).
 - Dataformatet ska överensstämma med det begärda formatet (i detta fall, JSON).
- **Faktiska Resultat:**



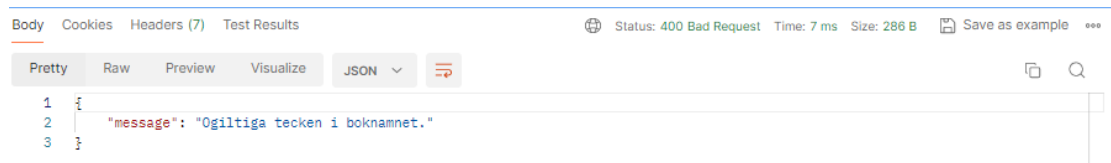
```
Body 200 OK 16 ms 412 B Save as example ...  
Pretty Raw Preview Visualize JSON  
1 {  
2   "total": 1,  
3   "books": [  
4     {  
5       "_id": "66245b069b8e686f67f2fefc",  
6       "Name": "Awesome Steel Chair 1",  
7       "ISBN": 3629663257859,  
8       "Price": 353,  
9       "Genre": "Jazz",  
10      "ReleaseDate": "2021-08-12T00:00:00.000Z"  
11    }  
12  ]  
13 }
```

- **Slutsats:** Basera slutsatsen på jämförelsen mellan de förväntade och de faktiska resultaten. Om API:et returnerade data i det korrekta formatet, bekräfta att det fungerar som förväntat. Om inte, identifiera detta som ett potentiellt område för förbättring eller felsökning.

Manuellt Testfall 3:

Validera ogiltiga GET-förfrågningar - 400 Bad Request

- **Syfte:** Att testa API:ets respons på ogiltiga GET-förfrågningar och verifiera att det returnerar statuskod 400.
- **Utförande:**
 1. Skicka en GET-förfrågan till
`http://localhost:3000/api/books?name=!%#.`
 2. Observera svaret från servern.
- **Förväntade Resultat:** API:et ska identifiera förfrågan som ogiltig och returnera statuskod 400 Bad Request.
- **Faktiska Resultat:**



- **Slutsats:** Konfirmera att API:et hanterar ogiltiga förfrågningar korrekt.

Manuellt Testfall 4: Validera GET-förfrågningar med specifika filter

- **Syfte:** Att verifiera att API:et korrekt returnerar data baserat på specifika sökkriterier och filter.
- **Utförande:**
 1. Skicka en GET-förfrågan till
`http://localhost:3000/api/books?name=Awesome Steel Chair 1&isbn=3629663257859&price=353&genre=Jazz&releaseDate=2021-08-12`.
 2. Observera och analysera det returnerade svaret.
- **Förväntade Resultat:** API:et ska returnera data som exakt matchar de angivna filtren.
- **Faktiska Resultat:**

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

- URL:** `http://localhost:3000/api/books?name=Awesome Steel Chair 1&isbn=3629663257859&price=353&genre=Jazz&releaseDate=2021-08-12`
- Method:** GET
- Status:** 200 OK
- Time:** 18 ms
- Size:** 412 B
- Response body:** Contains string
- JSON body:**

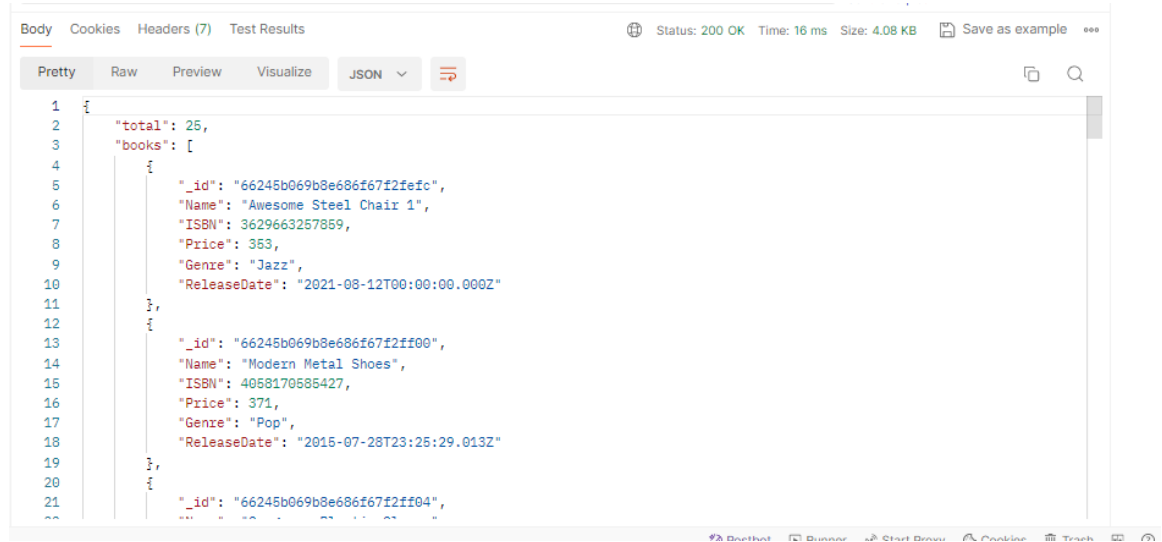
```
1 {
2   "total": 1,
3   "books": [
4     {
5       "_id": "66245b069b8e686f67f2f6fc",
6       "Name": "Awesome Steel Chair 1",
7       "ISBN": "3629663257859",
8       "Price": 353,
9       "Genre": "Jazz",
10      "ReleaseDate": "2021-08-12T00:00:00.000Z"
11    }
12  ]
13 }
```

- **Slutsats:** API et returnerar korrekt filtrering och funktionaliten hanteras enligt dokumentation

Manuellt Testfall 5:

Validera paginerade GET-förfrågningar

- **Syfte:** Att säkerställa att API:et korrekt paginerar resultat vid stora datamängder.
- **Utförande:**
 1. Skicka en GET-förfrågan till
`http://localhost:3000/api/books?page=1&limit=25`.
 2. Observera hur datan pagineras och returneras.
- **Förväntade Resultat:** API:et ska effektivt dela upp och returnera första sidan med 25 poster.
- **Faktiska Resultat:**



```
1 {
2   "total": 25,
3   "books": [
4     {
5       "_id": "66245b069b8e686f67f2fe1c",
6       "Name": "Awesome Steel Chair 1",
7       "ISBN": 3629663257859,
8       "Price": 353,
9       "Genre": "Jazz",
10      "ReleaseDate": "2021-08-12T00:00:00.000Z"
11    },
12    {
13      "_id": "66245b069b8e686f67f2ff00",
14      "Name": "Modern Metal Shoes",
15      "ISBN": 4058170585427,
16      "Price": 371,
17      "Genre": "Pop",
18      "ReleaseDate": "2015-07-28T23:25:29.013Z"
19    },
20    {
21      "_id": "66245b069b8e686f67f2ff04",
22      "Name": "Modern Metal Shoes",
23      "ISBN": 4058170585427,
24      "Price": 371,
25      "Genre": "Pop",
26      "ReleaseDate": "2015-07-28T23:25:29.013Z"
27    }
28  ]
29 }
```

- **Slutsats:** Bedöm om pagineringen fungerar som förväntat och notera eventuella problem.

Manuellt Testfall 6:

Validera hantering av specialtecken och icke-engelska texter

- **Syfte:** Att verifiera att API:et korrekt hanterar förfrågningar med specialtecken och icke-engelska tecken i texten.
- **Utförande:**
 1. Skicka en GET-förfrågan till **http://localhost:3000/api/books?name=En bok med ÄÄÖ.**
 2. Observera API:ets respons och hur det hanterar specialtecken.
- **Förväntade Resultat:** API:et ska korrekt returnera data utan fel eller teckenkorruption.
- **Faktiska Resultat**

SARAS Bookfinder / Manuella Tester / 6. Validate if the API handles special characters and non-English text correctly in input data and returned responses using an automated testing tool.

GET http://localhost:3000/api/books?name=En bok med ÄÄÖ

Params Authorization Headers (7) Body Pre-request Script Tests Settings

1

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Snippets

- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request
- Status code: Code is 200
- Response body: Contains string

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 16 ms Size: 405 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "total": 1,
3   "books": [
4     {
5       "_id": "662a947e197d6bb9477a627f",
6       "Name": "En bok med ÄÄÖ!",
7       "ISBN": 123456,
8       "Price": 3000,
9       "Genre": "drama",
10      "ReleaseDate": "2024-01-20T00:00:00.000Z"
11    }
12  ]
13 }
```

- **Slutsats:** API t hanterar korrekt förfrågan

Manuellt Testfall 7:

Validera samtidiga förfrågningar

- **Syfte:** Att testa API:ets förmåga att hantera flera samtidiga förfrågningar och bibehålla prestanda och datakonsistens.
- **Utförande:**
 1. Skicka flera samtidiga GET-förfrågningar till **http://localhost:3000/api/books**.
 2. Analysera hur API:et hanterar dessa samtidiga anrop.
- **Förväntade Resultat:** API:et ska effektivt hantera alla förfrågningar utan fel eller prestandaförlust.
- **Faktiska Resultat:**

SARAS Bookfinder - Run results

Run AgainAutomate Run+ New RunExport Results

Ran today at 12:04:44 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	10	1s 214ms	0	20 ms

All TestsPassed (0)Failed (0)Skipped (0)

Generate TestsView Summary

Iteration 1

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK18 ms10.483 KB

No tests found

Iteration 2

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK16 ms10.483 KB

No tests found

Iteration 3

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK22 ms10.483 KB

No tests found

Iteration 4

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK22 ms10.483 KB

No tests found

Iteration 5

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK21 ms10.483 KB

No tests found

Iteration 6

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK19 ms10.483 KB

No tests found

Iteration 7

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK21 ms10.483 KB

No tests found

Iteration 8

GET7. Validate concurrent requests to the API to ensure that it can handle multiple users and maintain data consistency.
http://localhost:3000/api/books

200 OK19 ms10.483 KB

No tests found

PostbotRunnerStart ProxyCookiesTrash

- **Slutsats:** Slutsats baserad på API:ets förmåga att hantera hög belastning så är det lyckat

Manuellt Testfall 8:

Validera PUT-förfrågningar för att uppdatera en bok

- **Syfte:** Att manuellt kontrollera att API:et korrekt hanterar PUT-förfrågningar och returnerar lämpliga svar och statuskoder.
- **Utförande:**
 1. Skicka en PUT-förfrågan till **http://localhost:3000/api/books/662aa196cdec73d79be79845** med uppdaterad information om en bok.
 2. Observera API:ets respons, speciellt statuskoden och hur datan har uppdaterats.
- **Förväntade Resultat:** API:et bör uppdatera bokposten korrekt och returnera statuskoden 200 OK eller 204 No Content om inga synliga uppdateringar gjorts.
- **Faktiska Resultat:**

The screenshot displays a REST client interface with a PUT request to the URL `http://localhost:3000/api/books/662aa196cdec73d79be79845`. The request body is a JSON object with the following fields:

```
1 {
2   "Name": "En bok om POST sucsess 2 ",
3   "ISBN": 123456,
4   "Price": 3000,
5   "Genre": "drama",
6   "ReleaseDate": "2024-01-20"
7 }
```

The response status is 200 OK. The response body is a JSON object with the following fields:

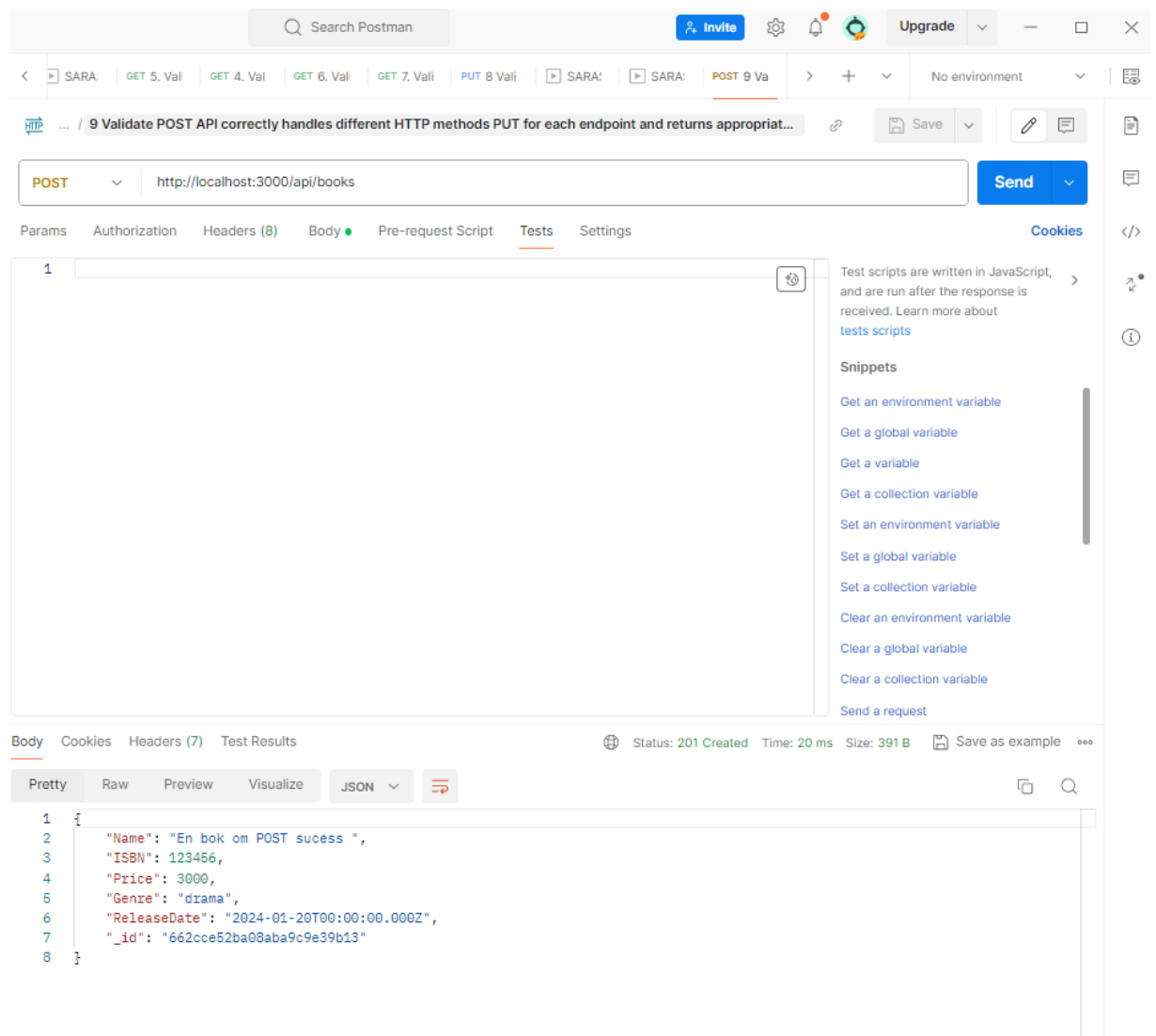
```
1 {
2   "updated": true,
3   "book": {
4     "_id": "662aa196cdec73d79be79845",
5     "Name": "En bok om POST sucsess 2 ",
6     "ISBN": 123456,
7     "Price": 3000,
8     "Genre": "drama",
9     "ReleaseDate": "2024-01-20T00:00:00.000Z"
10  }
11 }
```

- **Slutsats:** Bedöm om API:ets hantering av PUT-förfrågningar är korrekt och om uppdateringarna reflekteras på rätt sätt.

Manuellt Testfall 9:

Validera POST-förfrågningar för att skapa nya bokposter

- **Syfte:** Att manuellt verifiera att API:et korrekt hanterar POST-förfrågningar för att skapa nya bokposter.
- **Utförande:**
 1. Skicka en POST-förfrågan till **http://localhost:3000/api/books** med data för en ny bok.
 2. Observera API:ets respons, inklusive statuskoden och detaljer om den skapade boken.
- **Förväntade Resultat:** API:et bör acceptera förfrågan och skapa en ny bok, returnera statuskoden 201 Created med bokens detaljer.
- **Faktiska Resultat:**



- **Slutsats:** Lyckat resultat API:et hanterar förfrågan korrekt

Manuellt Testfall 10: Prestandatestning under hög belastning

- **Syfte:** Att verifiera API:ets prestanda och stabilitet under scenarier med hög belastning genom att simulera många simultana förfrågningar.
- **Utförande:**
 1. Använd Postman för att skicka högt antal simultana GET-förfrågningar till **http://localhost:3000/api/books**.
 2. Analysera API:ets respons, inklusive svarstider och eventuella fel.
- **Förväntade Resultat:** API:et ska kunna hantera hög belastning utan signifikant prestandaförlust eller fel.
- **Faktiska Resultat:**

The screenshot displays the Postman interface for a test run titled "SARAS Bookfinder - Run results". The test was executed at 12:08:56. The summary table shows 10 iterations, a duration of 1s 305ms, 0 failed tests, and an average response time of 16 ms. The test details for each iteration are as follows:

Iteration	Method	Path	Status	Response Time	Size
1	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	19 ms	10.634 KB
2	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	16 ms	10.634 KB
3	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	14 ms	10.634 KB
4	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	16 ms	10.634 KB
5	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	14 ms	10.634 KB
6	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	17 ms	10.634 KB
7	GET	10 Validate performance test that simulates a large number of users making requests simultaneously to check the API's performance under heavy load. http://localhost:3000/api/books	200 OK	16 ms	10.634 KB

All tests passed successfully. The interface also shows a "No tests found" message for each iteration, indicating that the test suite is empty or the test runner is not finding the tests.

- **Slutsats** API:et hanterar förfrågningarna under hög belastning