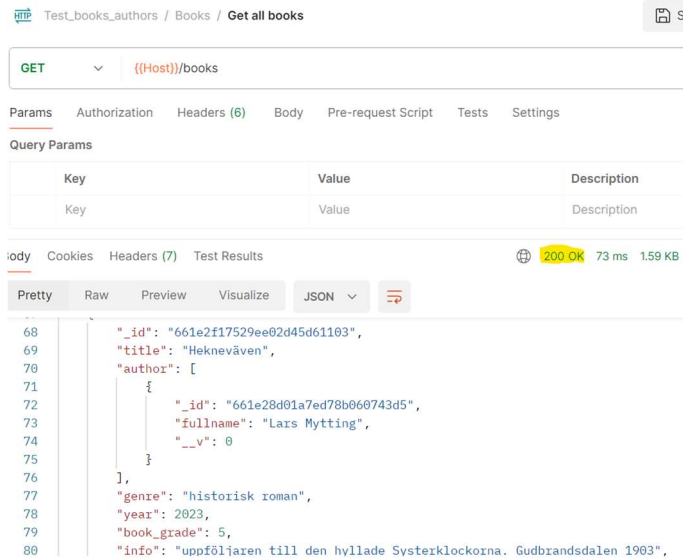


# Rapport testprocess Bookfinder API

## 1. Verify that the API returns the correct HTTP status code (e.g., 200 OK) for a successful GET request.

### - Manuellt

Sökte på alla böcker med en GET-request och fick fram förväntad info samt statuskod 200.



### - Automatiserat

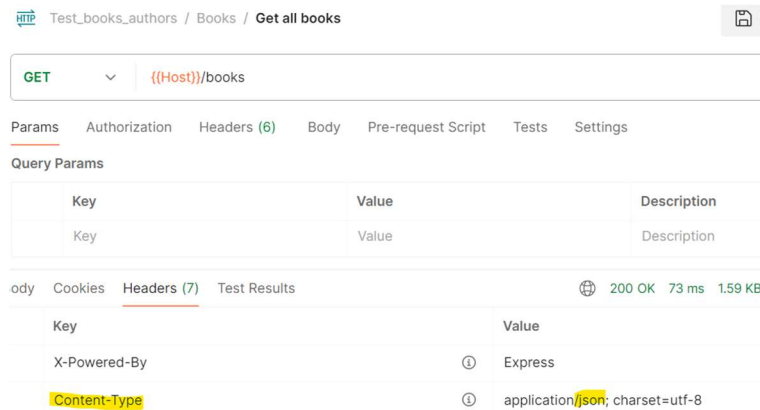
Skrev ett test för att kontrollera så statuskod 200 visas. Testresultatet visade PASS för "Status code is 200", som förväntat.



## 2. Check if the API returns the expected data format (e.g., JSON, XML in the response).

- Manuellt

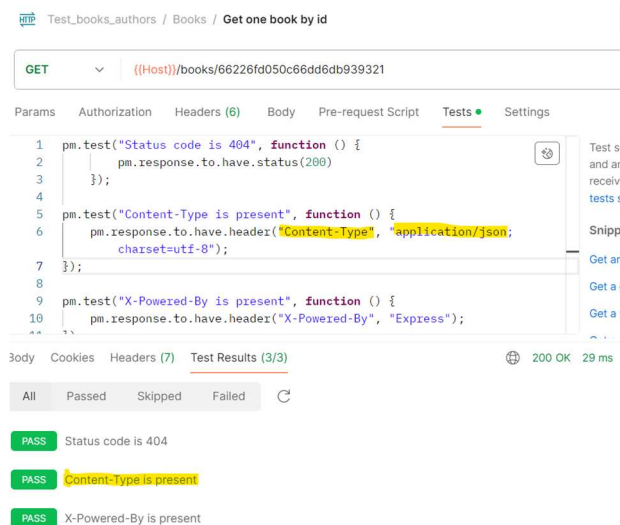
Sökte på alla böcker med en GET-request och ser att det i Headers och Content-Type visar det förväntade dataformatet JSON.



- Automatiserat

Skrev ett test för att kontrollera att Headers "Content type key" med value `application/json` finns tillgängligt.

Testet passerar och förväntat resultat visas, dvs. att content-type json is present.

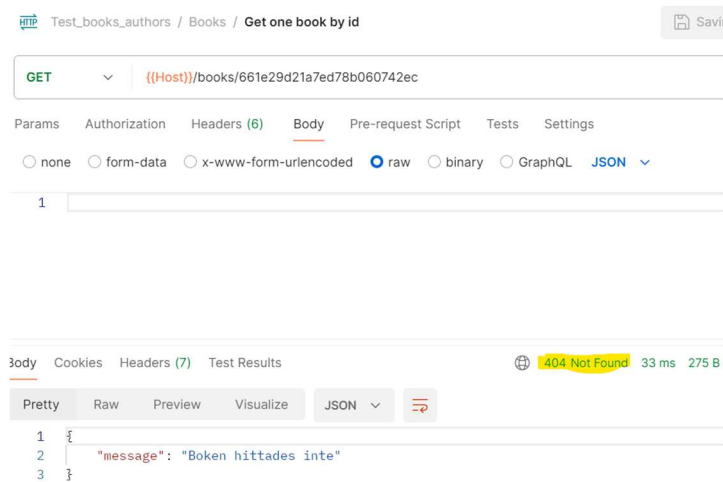


### 3. Ensure that the API returns the correct http status code (e.g., 400 Bad Request) for an invalid request.

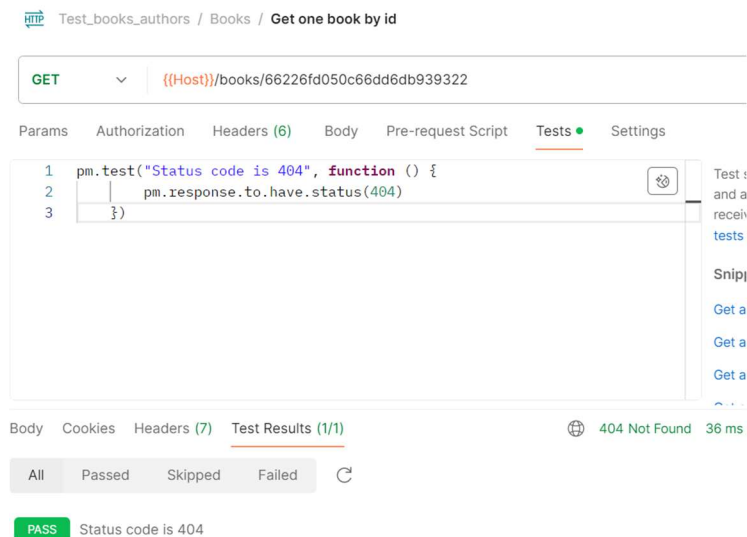
- Manuellt

Sökte på en bok med en GET-request och ett id som inte finns och fick då det förväntade felmeddelandet att boken inte finns samt statuskod 404 "Not Found".

(Jag valde statuskod 404 i detta testfall. Statuskod 400 testas av i test nr.12)

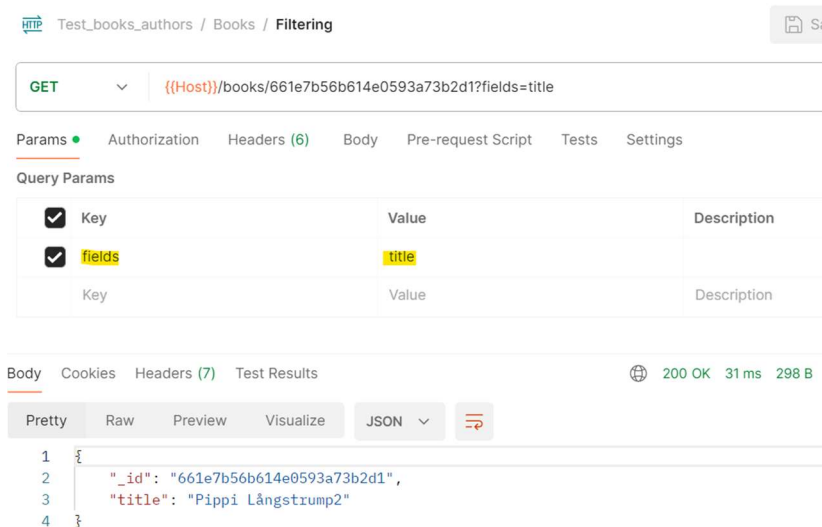


- Automatiserat  
Skrev ett test som som vid felaktigt inmatat id visar statuskod "404 Not Found", dvs. förväntat resultat.



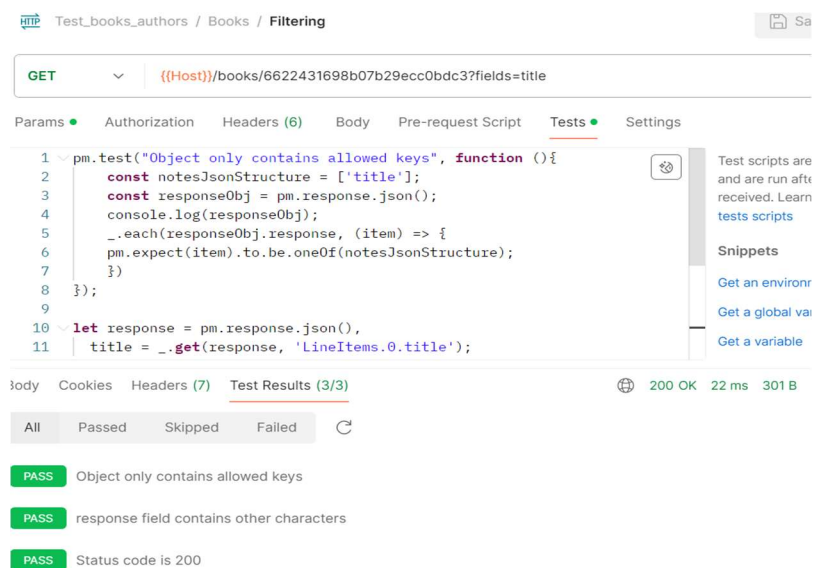
#### 4. Test if the API returns the correct data when querying with specific filters or search criteria.

- Manuellt  
Söker på en bok med en GET-request och ett id samt sätter parametern fields för att kunna söka på exempelvis title, year eller info. Förväntat resultat visas när jag gör en sökning på titel.



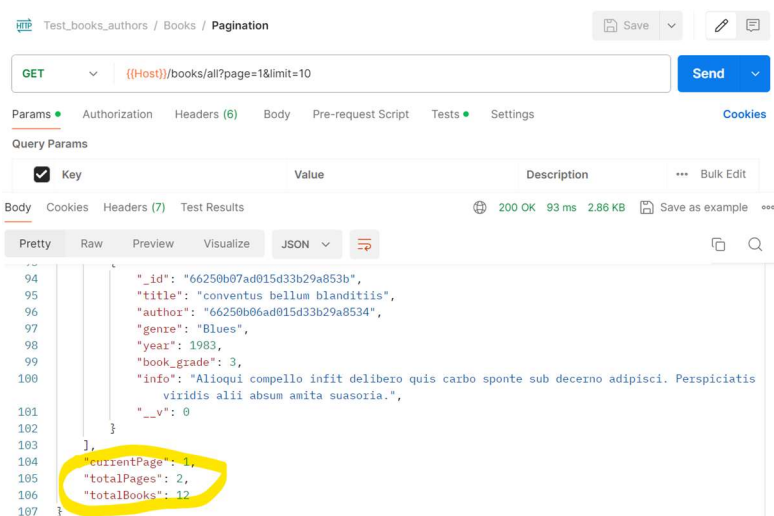
#### - Automatiserat

Automatiserar sökning genom att välja en bok med id, söker på titel och om svar endast innehåller titel går test igenom, statuskod 200 visas.



### 5. Verify that the API returns paginated results when a large number of records are requested.

- Manuellt – kan inte se hur detta ska kunna testas manuellt.
- Automatiserat  
Söker på alla böcker med en GET-request (api/books/all), använder query parametrarna page och limit. Väljer en limit på 10 böcker per sida. Seedat in 12st böcker och därav blir det 2 sidor totalt som visas.



```
server.get("/api/books/all", async (req, res) => {
  try {
    const page = parseInt(req.query.page) || 1 // aktuell sida
    const limit = parseInt(req.query.limit) || 10 // antal elementer på en sida

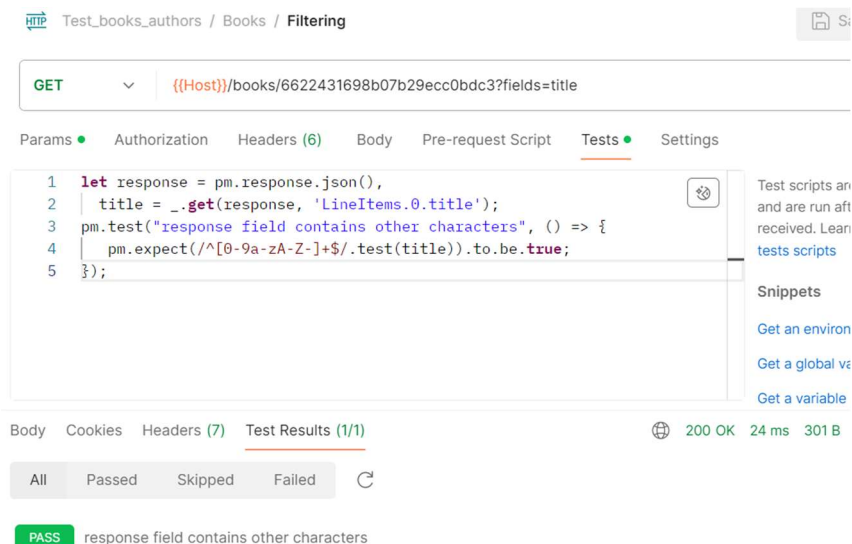
    const totalBooks = await Book.countDocuments();
    const totalPages = Math.ceil(totalBooks / limit);

    const books = await Book.find()
      .limit(limit);

    res.status(200).json({
      books,
      currentPage: page,
      totalPages,
      totalBooks
    });
  }
});
```

## 6. Check if the API handles special characters and non-English text correctly in input data and returned responses.

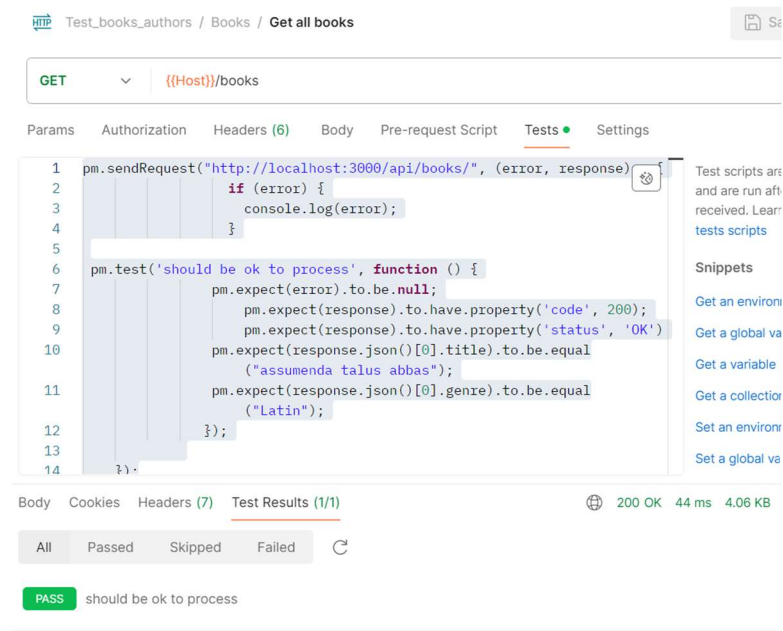
- Manuellt  
Med mina ögon testat av så att specialtecken och åäö, kan inte se att det ställer till problem på något sätt utan dessa tecken funkar fint att både mata in och få ut presenterat.
- Automatiskt  
Test skiven för att kontrollera om response field (i detta fall title) innehåller andra tecken än a-z, 0-9, A-Z.



## 7. Test the API's response when sending concurrent requests to ensure that it can handle multiple users and maintain data consistency.

- Manuellt kan inte se hur detta kan testas av manuellt.
- Automatiskt

Skapat ett test som gör en kedjerequest i api/books och hämtar titel och genre för första boken, samt statuskod. Testerna går igenom som förväntat.



## 8. Test if the API correctly handles different http methods (GET, POST, PUT, DELETE) for each endpoint and returns appropriate status codes and responses for each method.

- Manuella tester
- Sökte på alla böcker med en GET-request och fick upp förväntad info samt statuskod 200, se uppgift 1.
- Sökt på en bok med en GET-request och id och får fram förväntat resultat med statuskod 200.

Test\_books\_authors / Books / Get all books

GET `{{Host}}/books`

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

body Cookies Headers (7) Test Results 200 OK 73 ms 1.59 KB

Pretty Raw Preview Visualize JSON

```

68   "_id": "661e2f17529ee02d45d61103",
69   "title": "Hekneväven",
70   "author": [
71     {
72       "_id": "661e28d01a7ed78b060743d5",
73       "fullname": "Lars Mytting",
74       "_v": 0
75     }
76   ],
77   "genre": "historisk roman",
78   "year": 2023,
79   "book_grade": 5,
80   "info": "uppföljaren till den hyllade Systerklockorna. Gudbrandsdalen 1903",

```

- Skapade en bok med POST-request och boken skapades korrekt samt statuskod 201 för Created visades.

Test\_books\_authors / Books / Create a book

POST `{{Host}}/books`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```

1  {
2    "title": "Pippi Långstrump",
3    "author": "661e7aa5b614e0593a73b2cb",
4    "genre": "barn/ungdom",
5    "year": 1980,
6    "book_grade": 5,
7    "info": "Pippi bor tillsammans med Lilla Gubben och Herr Nilsson i Villa Villerkulla"
8  }
9
10

```

body Cookies Headers (7) Test Results 201 Created 79 ms 483 B

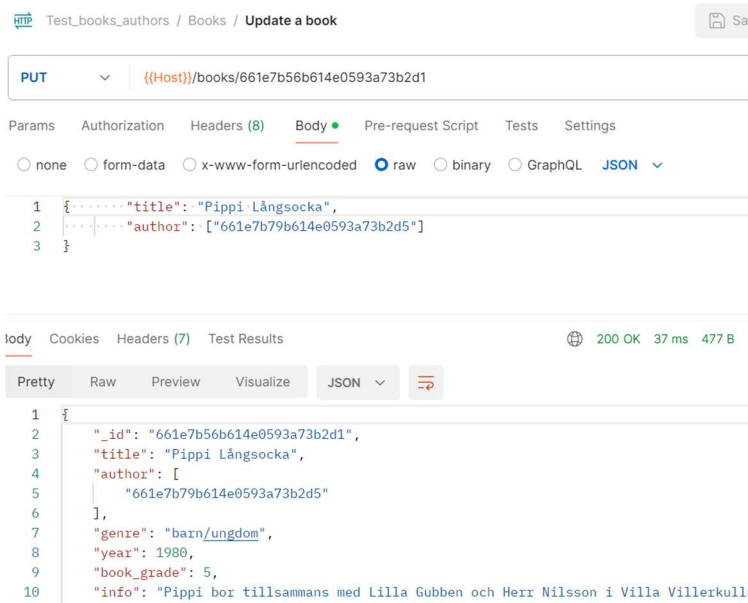
Pretty Raw Preview Visualize JSON

```

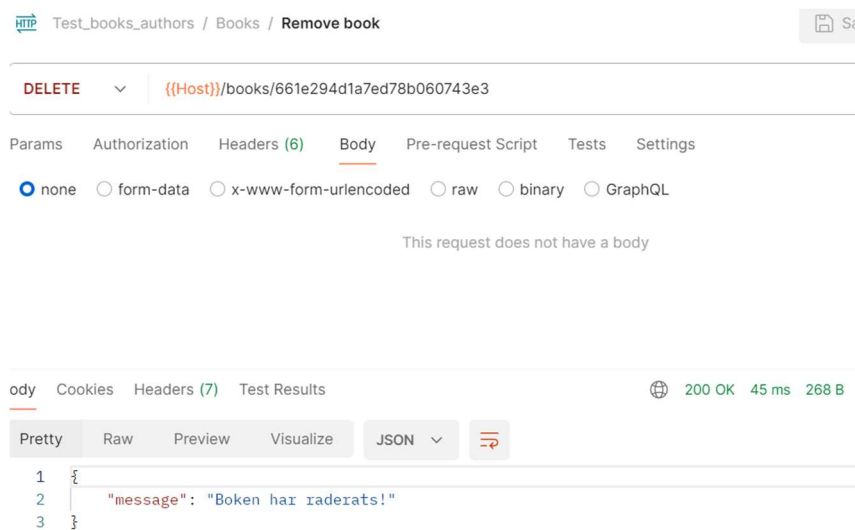
1  {
2    "title": "Pippi Långstrump",
3    "author": [
4      "661e7aa5b614e0593a73b2cb"
5    ],
6    "genre": "barn/ungdom",

```

- Uppdaterade en bok med PUT request och id och uppdateringen genomfördes och resultatet visades korrekt och statuskod 200 visades.



- Raderade en bok med DELETE request och id och boken raderades och statuskod 200 visades. Message: Boken har raderats.



- Automatiserade tester

Förberedde mina requests med data och startade sedan och körde en Run Collection i Postman innehållande GET, POST, PUT och DELETE för Books.

Resultatet var som förväntat för de requests som hade tester skrivna.



Test\_books\_authors - Run results

Run Again
Automate Run
+ New Run
Export Res

Ran today at 11:51:00 · View all runs

Source Runner	Environment	Iterations	Duration	All tests	Avg. Resp. Time
	none	1	1s 166ms	4	65 ms

All Tests
Passed (4)
Failed (0)
Skipped (0)
View Summary

Iteration 1

GET
Get all books

/books

200 OK 179 ms 6.049 KB

PASS
Status code is 200

PASS
should be ok to process

POST
Create a book

http://localhost:3000/api/books

201 Created 33 ms 479 B

No tests found

PUT
Update a book

http://localhost:3000/api/books/66226fd050c66dd6db939323

200 OK 26 ms 536 B

PASS
Status code is 200

PASS
Check if updated

DELETE
Remove book

/books/66250aec55805a97439d65ec

200 OK 36 ms

No tests found

## 9. Check if the API correctly handles updates to existing records, ensuring that changes are saved and reflected in subsequent requests.

- Manuellt – kan inte se hur detta fall kan testas manuellt.
- Automatiskt

Väljer att uppdatera titeln för en bok. Om titeln som uppdateras i body är likvärdig med titeln som finns i testet, går testet igenom. Vilket är förväntat resultat.

Test\_books\_authors / Books / Update a book

Sa

PUT
{{Host}}/books/66226fd050c66dd6db93931f

Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings

☐ none
☐ form-data
☐ x-www-form-urlencoded
☒ raw
☐ binary
☐ GraphQL
JSON

```

1 {
2   "title": "amplus thema venia",
3   "author": {
4     "_id": "66226fcf50c66dd6db939318",
5     "fullname": "Beth Runte",
6     "book": []
7   }
8 }

```

body
Cookies
Headers (7)
Test Results (2/2)
200 OK 25 ms 480 B

All
Passed
Skipped
Failed

PASS
Status code is 200

PASS
Check if updated

Test\_books\_authors / Books / Update a book

PUT {{Host}}/books/66226fd050c66dd6db93931f

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```

1 pm.test("Status code is 200", function () {
2     pm.response.to.have.status(200)
3 })
4
5 pm.test('Check if updated', function () {
6     pm.expect(pm.response.json().title).to.be.equal("amplus thema
7     venia");
8 })

```

Test scripts are run after the response is received. Learn more about tests scripts

Snippets

Get an environment variable

Get a global variable

Get a variable

Body Cookies Headers (7) Test Results (2/2) 200 OK 25 ms 480 B

All Passed Skipped Failed

PASS Status code is 200

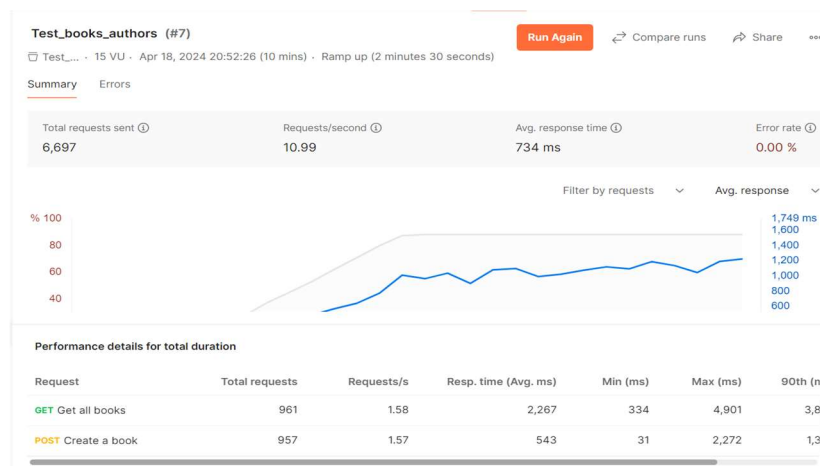
PASS Check if updated

## 10. Test the API's performance under heavy load, simulating a large number of users making requests simultaneously.

- Manuellt – kan inte se hur detta ska testas av manuellt.
- Automatiskt

Testade API performance testing i Postman genom att köra min collection av requests. Valde antal användare 15-20st och satte antal minuter som jag önskade köra testet.

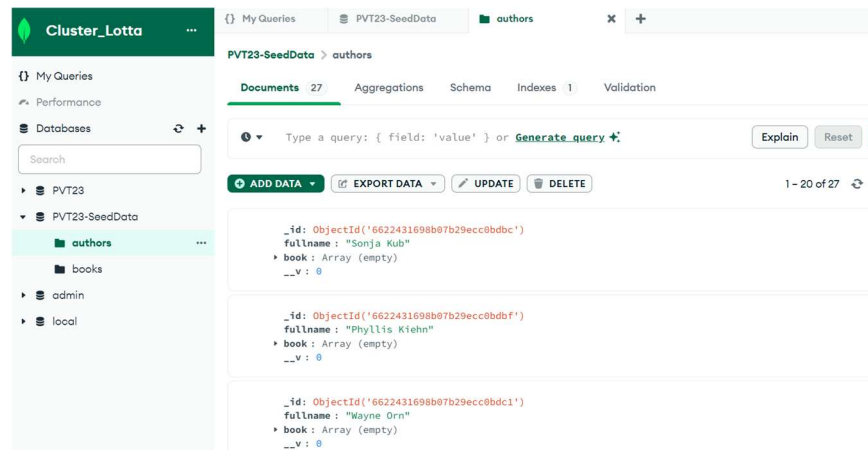
Fick avmarkera några av de requests som jag inte lyckats få korrekt utformade och fick därefter en error rate på 0%.



## 11. Verify that the API can recover gracefully from failures, such as database connection issues without compromising data integrity.

- Manuellt

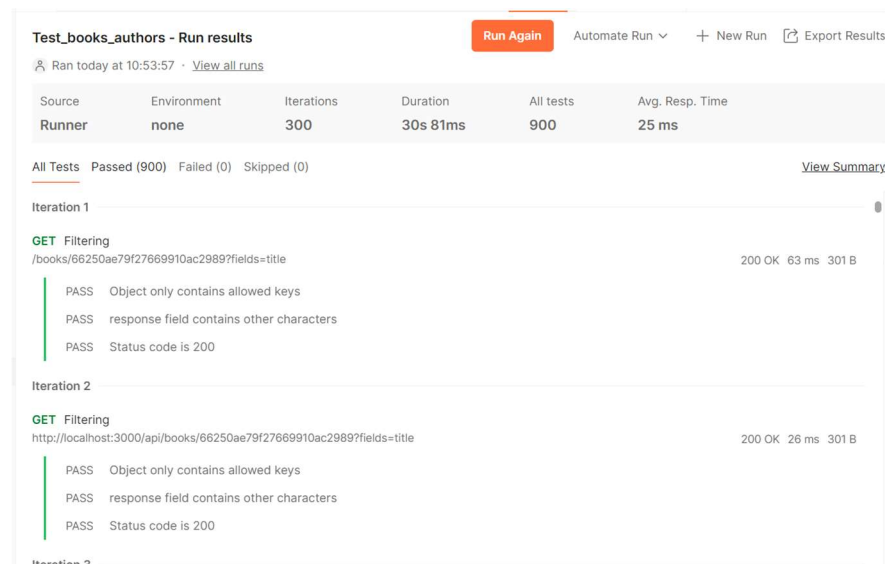
Disconnectar Mongo DB och connectar igen och ser då att all data i authors och books finns kvar under min PVT23-SeedData.



- Automatiskt

Väljer att köra en Get request med 300 iterationer. Bryts kopplingen till Mongo DB innan jag kör testet eller mitt i testet verkar inte spela någon roll, testet fortsätter att köra på och alla 300 iterationer lyckas.

Antar att det beror på att Postman sparar in datan vid första gången man kör testet och därför inte är beroende av databaskopplingen i dessa fall.



## 12. Test the API's ability to handle edge cases, such as requests with missing or invalid parameters, and ensure that appropriate error messages are returned.

- Manuellt och Automatiskt

Om jag försöker köra GET request på ett ID som har ett helt annat format än vad som förväntas (tex. lotta), så får jag felmeddelande "Felaktigt id" och felkod "400 Bad Request".



authors.js

```
server.get('/api/authors/:id', async (req, res) => {
  if (!mongoose.Types.ObjectId.isValid(req.params.id)) {
    return res.status(400).json({ message: "Felaktigt id" });
  }
})
```

### 13. Verify that the API correctly implements rate limiting to prevent abuse or excessive use of resources.

- Manuellt – kan inte se hur detta kan testas manuellt.
- Automatiskt

Installera **npm i express-rate-limit** paketet för rate limiting.

I server.js **import rateLimit from "express-rate-limit"** och skriver in nedan kod för rateLimit inställningar.

Sätter värdena på rate limit till 100 förfrågningar på en timme. Överskrids dessa får man upp ett meddelande och inga fler förfrågningar kan göras under denna period.

Limit sätts på samtliga requests under api't.

```
import rateLimit from "express-rate-limit"

const server = express()

const port = 3000

server.use(express.json())

const limiter = rateLimit({
  max: 100,
  windowMs: 60 * 60 * 1000,
  message: "Too many request from this IP, please try again in an hour",
});

server.use("/api", limiter);
```

I Postman under Headers finns nu följande tre "X-RateLimit" Headers för att identifiera konsumtions limit tillgängliga.

Jag har valt att testa av dessa tre från min *Get all authors* request för att se om de kan nås korrekt, vilket de gör.

The screenshot shows the Postman interface for a GET request to `{{Host}}/authors`. The 'Tests' tab is active, displaying three test scripts:

```
1 pm.test("X-RateLimit-Limit", function () {
2   pm.response.to.have.header("X-RateLimit-Limit", "100");
3 });
4
5 pm.test("X-RateLimit-Remaining", function () {
6   pm.response.to.have.header("X-RateLimit-Remaining");
7 });
8
9 pm.test("X-RateLimit-Reset", function () {
10  pm.response.to.have.header("X-RateLimit-Reset");
11 });
```

Below the tests, the 'Test Results (3/3)' section shows that all three tests passed:

- PASS** X-RateLimit-Limit
- PASS** X-RateLimit-Remaining
- PASS** X-RateLimit-Reset

The status bar at the bottom indicates a **200 OK** response with a response time of **58 ms**.

Länk till min Postman, finns i Readme.md på GitHub och även här:

<https://www.postman.com/lottasv/workspace/my-workspace>