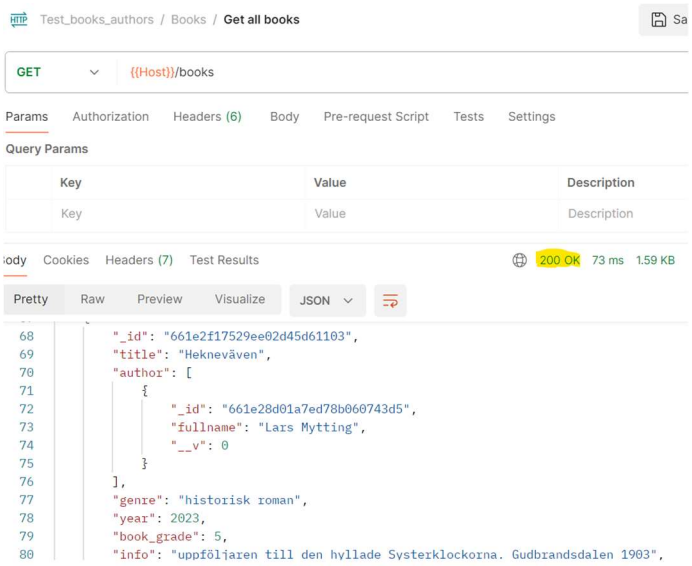
Rapport testprocess Bookfinder API

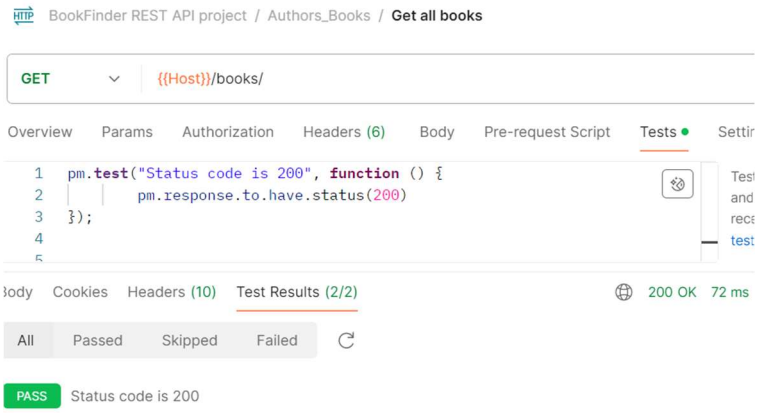
1. **Verify that the API returns the correct HTTP status code (e.g., 200 OK) for a successful GET request.**

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



    - Automatiserat

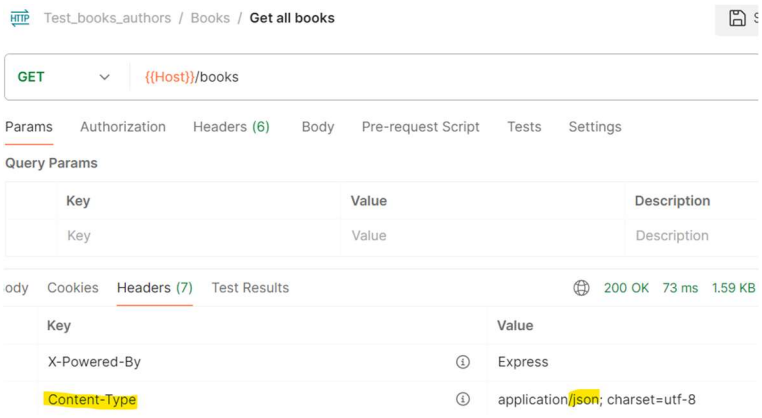
      Skrev ett test för att kontrollera så statuskod 200 visas. Testresultatet visade PASS för ”Status code is 200”, som förväntat.



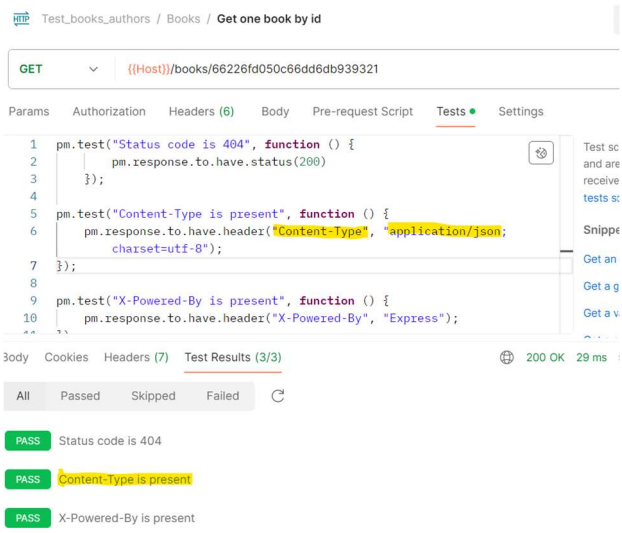
2. **Check if the API returns the expected data format (e.g., JSON, XML in the response.**

    - Manuellt

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



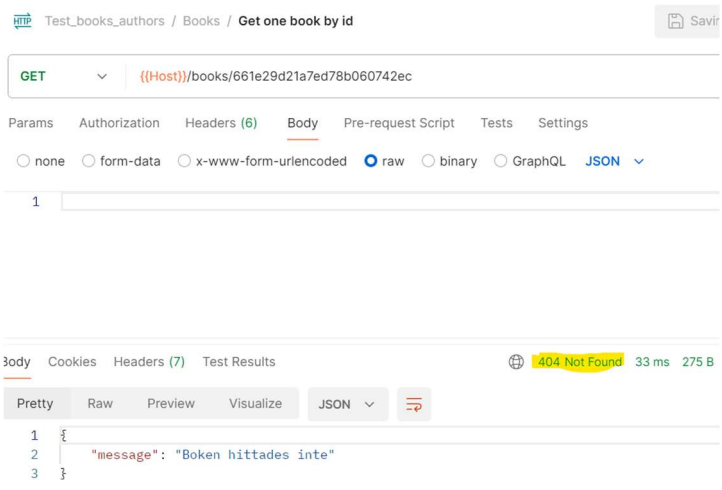
- Automatiserat
  Skrev ett test för att kontrollera att Headers ”Content type key” med value application/json finns tillgängligt.
  Testet passerar och förväntat resultat visas, dvs. att content-type json is present.



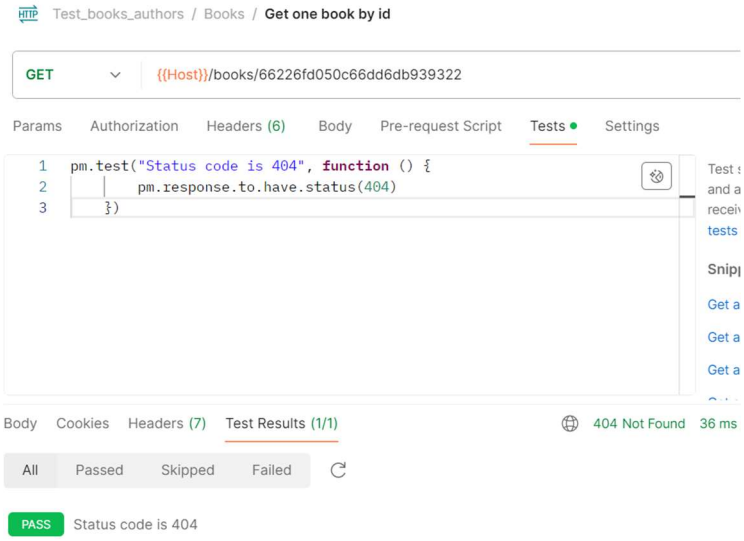
3. **Ensure that the API returns the correct http status code (e.g., 400 Bad Request) for an invalid request.**

- 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)

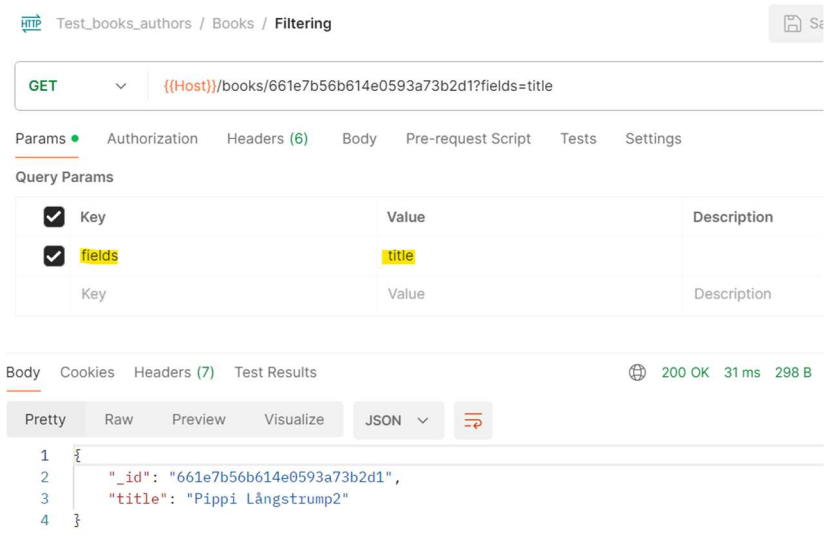


- Automatiserat
  Skrev ett test som vid felaktigt inmatat id visar statuskod ”404 Not Found”, dvs. förväntat resultat.

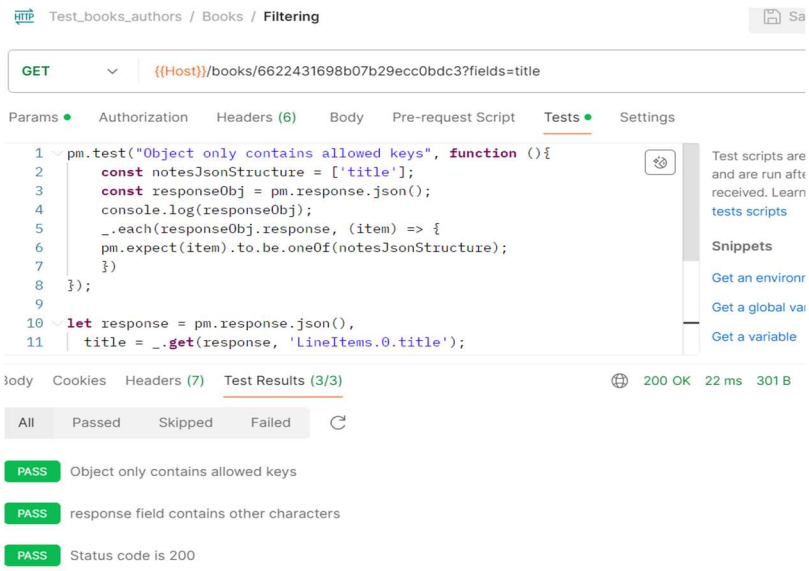


4. **Test if the API returns the correct data when querying with specific filters or search criteria.**

   - Manuellt
     Söker på en bok med en GET-request och ett id alternativt {{bookId}} för att söka på senast inlagda bok. Sätter parametern fields för att kunna söka på exempelvis title, year eller info. Förväntat resultat visas när jag gör en sökning på titel.



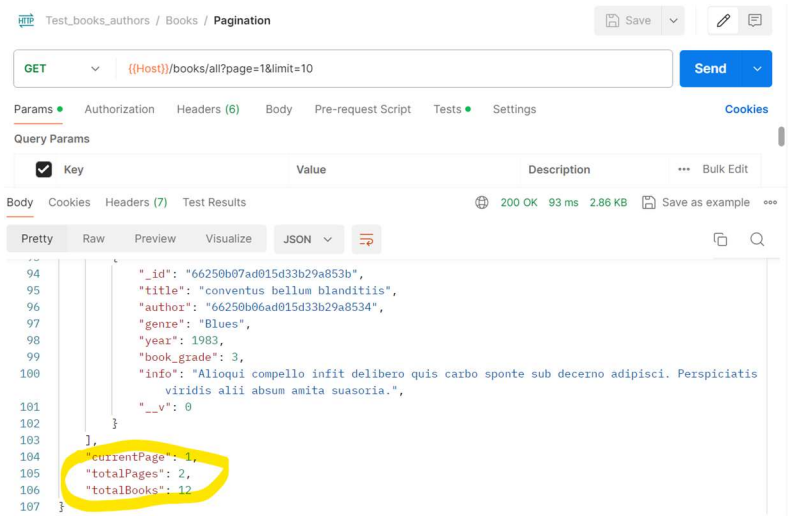
- Automatiserat
  Automatiserar sökning genom att välja en boks id (alternativt {{bookId}} för att göra filtrering på senast inlagda bok), söker på titel och om svar endast innehåller titel går test igenom, statuskod 200 visas.



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.

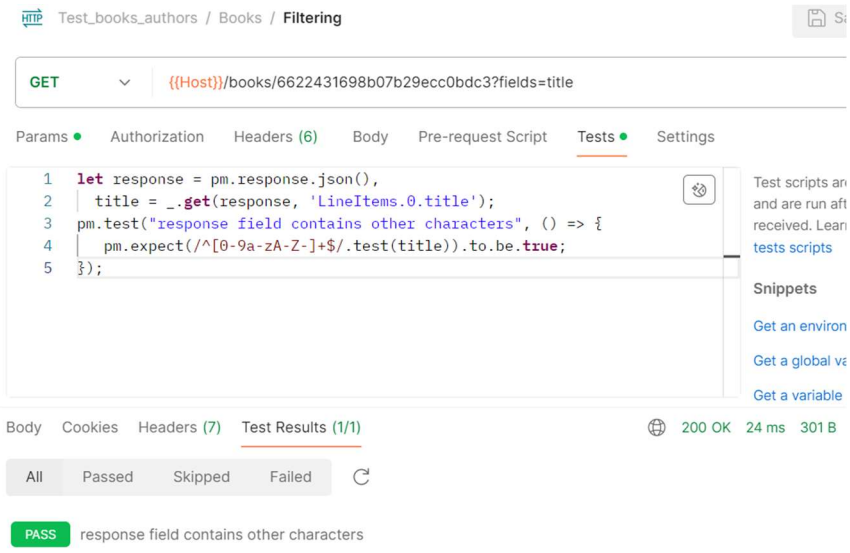



```
server.get("/api/books/all", async (req, res) => {
  try {
    const page = parseInt(req.query.page) || 1 // aktuell sida
    const limit = parseInt(req.query.limit) || 10 // antal elementer på en sida

    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.**

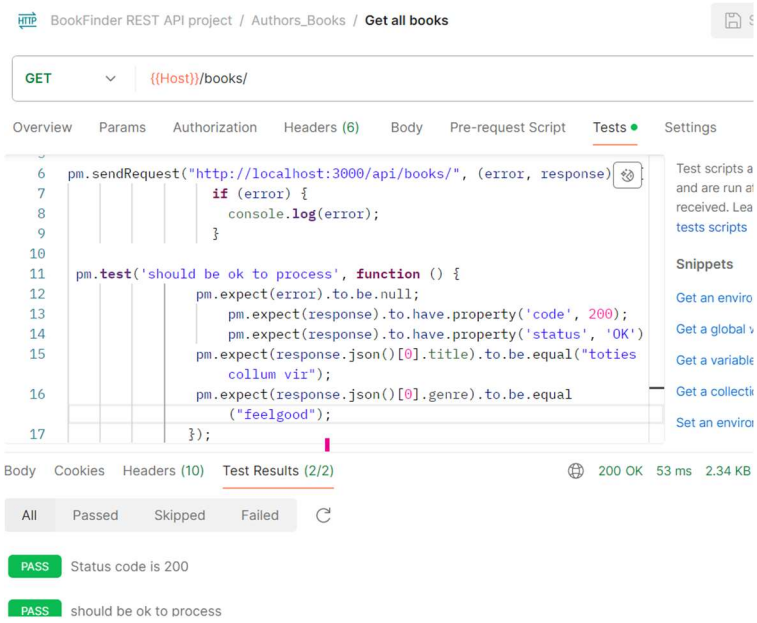
   - Manuellt
     Gör en PUT request för att se om det funkar att uppdatera titel och mata in specialtecken och åäö. Resultatet blev att det inte ställer till problem på något sätt utan dessa tecken funkar fint att både mata in och få ut presenterat.

   - Automatiskt
     Test skriven för att kontrollera om response field (i detta fall title) innehåller andra tecken än a-z, 0-9, A-Z. Skulle så vara fallet så skrivs meddelandet ”response field contains other characters” ut.



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

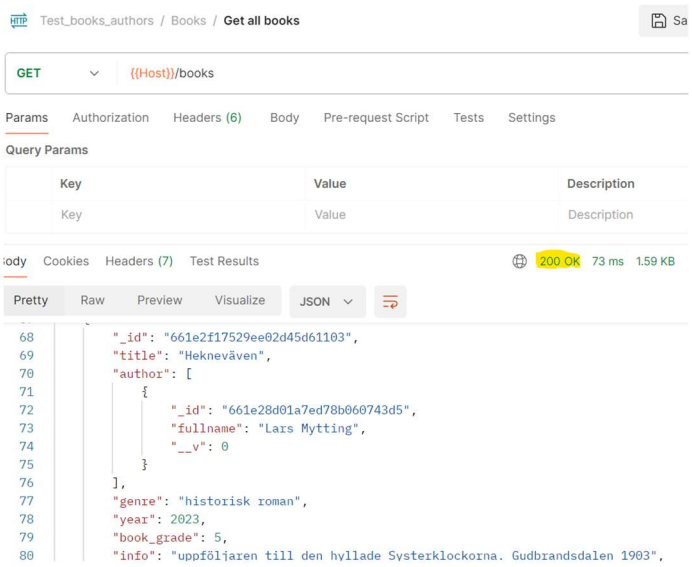
   - Manuellt kan inte se hur detta kan testas av manuellt.
   - Automatiskt
     Skapat ett test som gör en kedjerequest i api/books och hämtar titel och genre för första boken, samt statuskod. Testerna går igenom som förväntat.



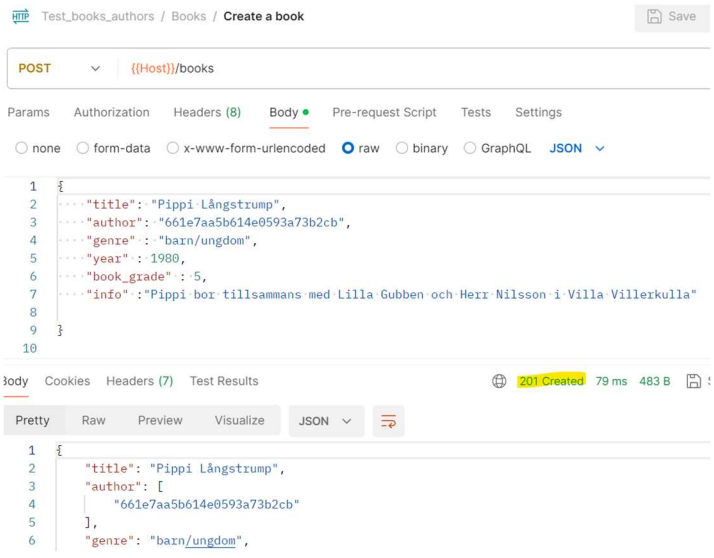
8. **Test if the API correctly handles different http methods (GET, POST, PUT, DELETE) for each endpoint and returns appropriate status codes and responses for each method.**

- **Manuella tester**

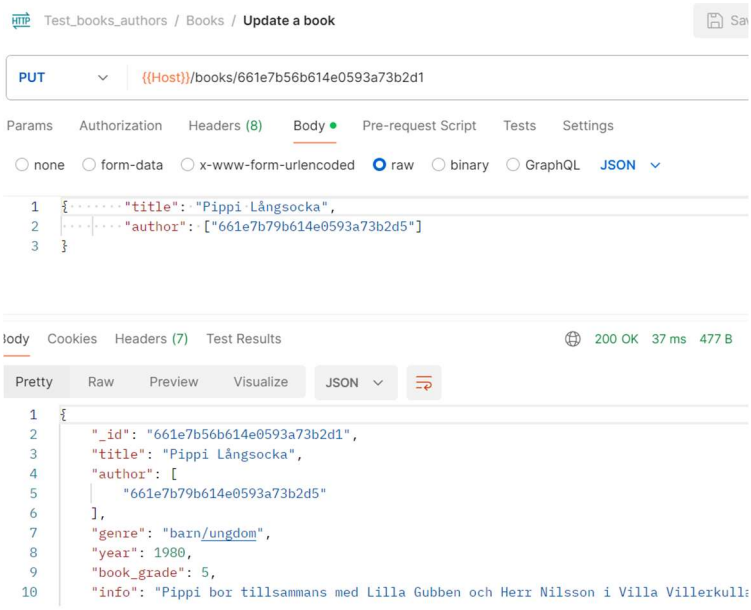
- Sökte på alla böcker med en GET-request och fick upp förväntad info samt statuskod 200, se uppgift 1.
- Sökt på en bok med en GET-request och id (alternativt {{bookId}} för att söka på senast inlagda bok) och får fram förväntat resultat med statuskod 200.



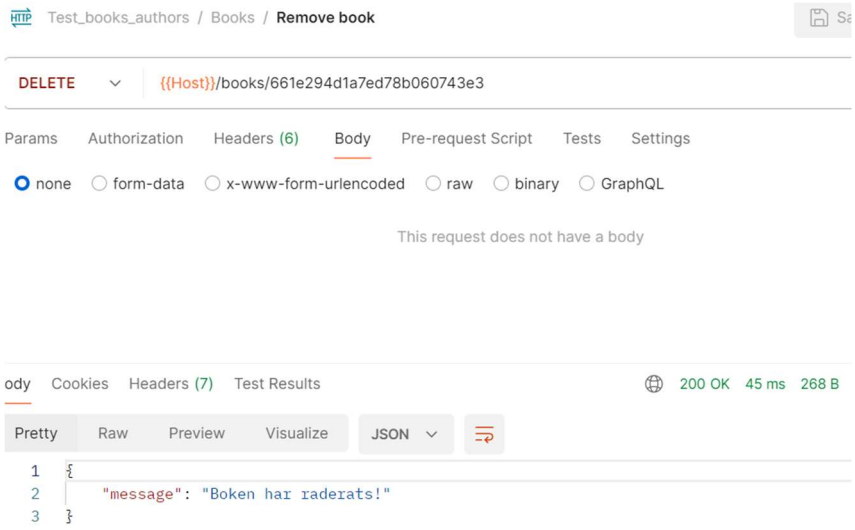
- Skapade en bok med POST-request och boken skapades korrekt samt statuskod 201 för Created visades.



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



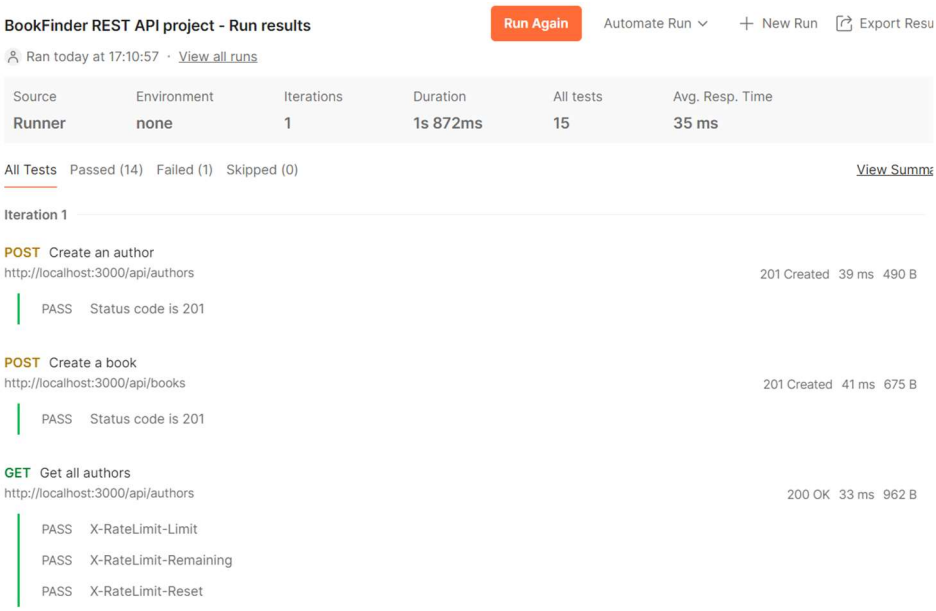
- Raderade en bok med DELETE request och id (alternativt {{bookId}} för att radera senast inlagda bok) och boken raderades och statuskod 200 visades. Message: Boken har raderats.



- **Automatiserade tester**

  Förberedde mina requests med data och startade sedan en Run Folder på Authors_Books i Postman.

  Resultatet var som förväntat, alla tester Passed utom en som failar då titeln som uppdateras i body skiljer sig mot titeln som finns i testet. Vill man att detta test också ska passera ändra till lika titlar enl. nr 9 nedan.

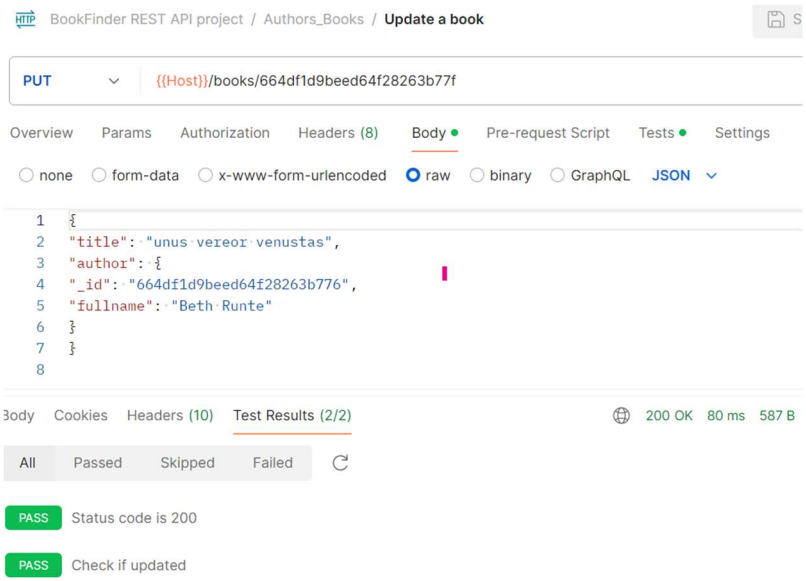


9. **Check if the API correctly handles updates to existing records, ensuring that changes are saved and reflected in subsequent requests.**

   - 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.

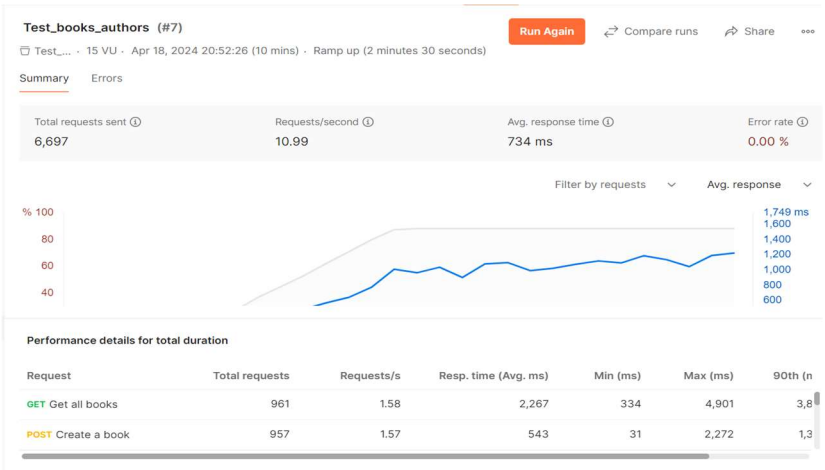


10. Test the API's performance under heavy load, simulating a large number of users making requests simultaneously.

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

- Automatiskt

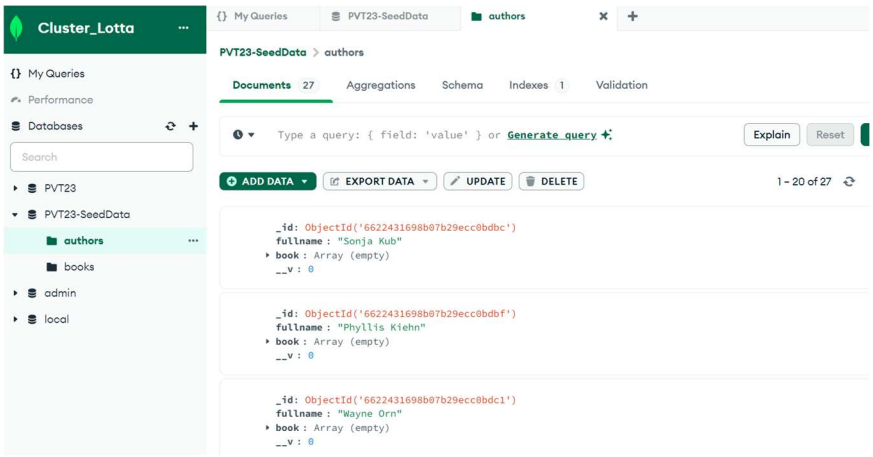
  Testade API performance testing i Postman genom att köra Run collection av requests. Valde antal användare 15-20st och satte antal minuter som jag önskade köra testet.
  Fick avmarkera några av de requests som jag inte lyckats få korrekt utformade och fick därefter en error rate på 0%.



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

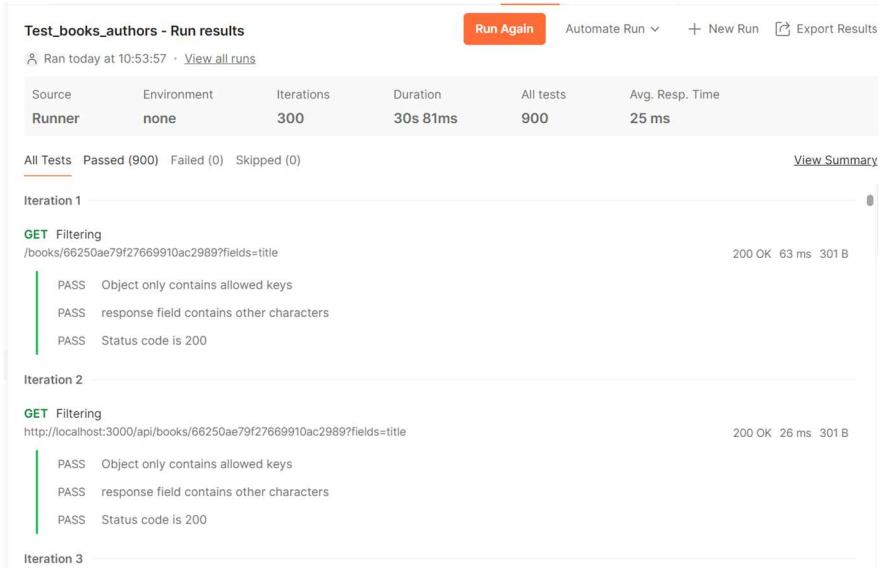
- Manuellt

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.

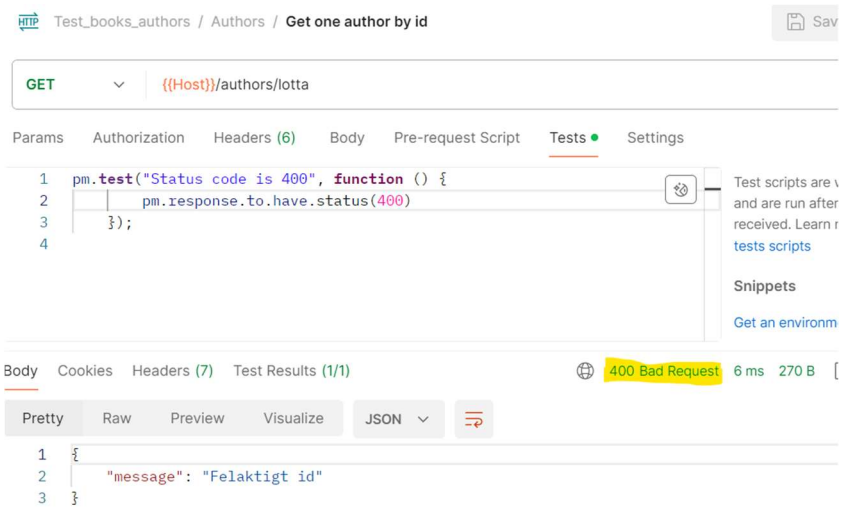


-

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.

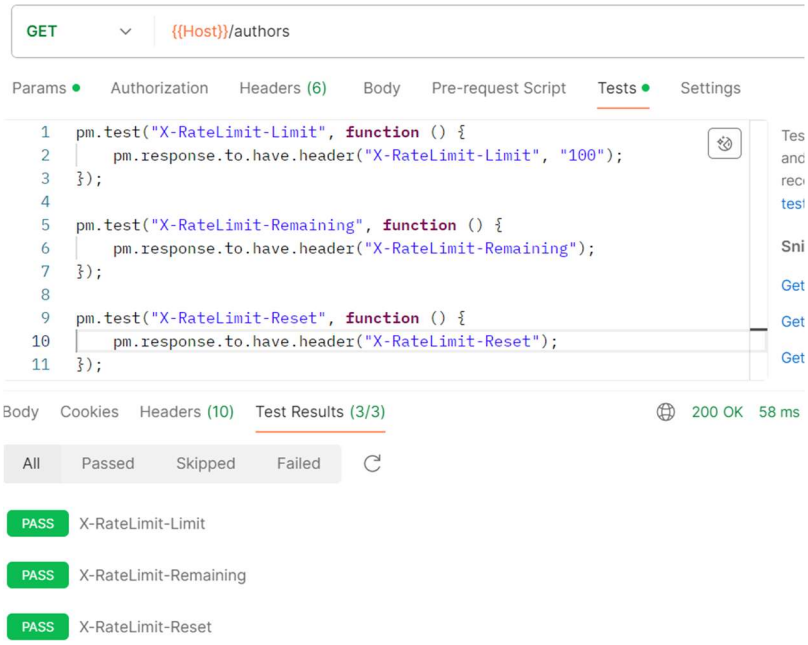
- Automatiskt

  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: "To many requests 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 att konsumtions limit är tillgänglig.

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.



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

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