LU08.A03 - Bookshelf mit Authentifikation

Erweitere deine Sammlung von Requests um verschiedene Aspekte zum automatisierten Testen.

Ausgangslage

Die erweiterte Version der Bookshelf-API (`https://it.bzz.ch/book/ext/`) umfasst eine Authentifikation und Autorisation. Ein Aufrufer kann eine von drei Rollen haben:

Rolle	Benutzername	Passwort	Berechtigung
admin	admin	admin	Zugriff auf alle API-Funktionen
user	musterh	geheim	Zugriff auf die Funktionen `read` und `list`
guest	-	-	Zugriff auf die Funktion `login`

Ein Aufrufer der kein gültiges Token besitzt, erhält die Rolle `guest` zugewiesen. Der darf ausschliesslich die Funktion `login` aufrufen, um sich zu authentifizieren. Fehlt dem Aufrufer die Berechtigung für die aufgerufene Funktion, so antwortet die API mit http-Statuscode **403** (Forbidden)

Ziele

- Wir testen jede API-Funktion mit jeder Benutzerrolle.
- Wir verwenden die bestehenden Testfälle, passen aber die *tests* an.

Um diese Ziele zu erreichen, lassen wir die Testfälle 1x pro Rolle durchlaufen. Für jeden Durchlauf verwenden wir das Login für die entsprechende Rolle.

Vorbereitung

Für diese Aufgabe stelle ich dir eine Collection zur Verfügung, welche die wichtigsten Requests beinhaltet. Lade die Collection „ Bookshelf_Python.postman_collection.json“ von Moodle herunter. Importiere diese Collection in Postman.

Variablen

Aus den letzten Aufgaben haben wir bereits die Variablen für die URL und das Token. Zusätzlich verwende ich die Variable `role` um zu speichern, mit welche Rolle ich aktuell testen.

Zusätzliche Requests

Ganz ohne neue Requests geht es nicht. In der importierten Collection findest du 4 zusätzliche Requests.

01 Initialize

Dieser Request initialisiert und steuert die Verarbeitung der Testdurchläufe. Er sendet nur pro-forma eine Anfrage, sondern legt lediglich die Werte der Variablen fest. Im Reiter „Pre-request Script“ findet du die Beschreibung der Logik, die du umsetzen musst.

Daneben gibt es 3 Requests, welche die `login`-Funktion mit unterschiedlichen Daten aufrufen.

02 Login admin

Dieser Request sendet den Benutzernamen und Passwort des Administators. Bei *tests* musst du drei Punkte umsetzen:

- Prüfe ob die Response den http-Status 200 enthält.
- Speichere das `token` aus der JSON-Response in die Umgebungsariable `token`.
- Lege fest, dass als nächstes der Request „10 Read Book“ ausgeführt werden soll.

03 Login user

Dieser Request sendet den Benutzernamen und Passwort des Administators. Bei *tests* musst du drei Punkte umsetzen:

- Prüfe ob die Response den http-Status 200 enthält.
- Speichere das `token` aus der JSON-Response in die Umgebungsariable `token`.
- Lege fest, dass als nächstes der Request „10 Read Book“ ausgeführt werden soll.

04 Login invalid

Dieser Request sendet einen unbekannte Benutzernamen und Passwort. Bei *tests* musst du zwei Punkte umsetzen:

- Prüfe ob die Response den http-Status **401** enthält.
- Lege fest, dass als nächstes der Request „10 Read Book“ ausgeführt werden soll.

Anpassen der Requests

In der Logik der *tests* müssen wir die unterschiedlichen Benutzerrollen berücksichtigen. Je nach Benutzerrolle wird der gleiche Request einen anderen http-Statuscode und andere Daten liefern.

Zum Beispiel "10 Read Book"

- Ist die Benutzerrolle „admin“ oder „user“ erwarte ich wie bisher den http-Statuscode **200** und das korrekte Buch.
- Ist die Benutzerrolle „guest“ erwarte ich **neu** den http-Statuscode **403** und ein leeres JSON-Array als Antwort.

Passe die Logik der *tests* so an, dass sie bei jeder Benutzerrolle korrekt reagieren.

Let's go!

Sind die Anpassungen gemacht, kommt der grosse Moment: Wir lassen die ganze Collection 3x durchlaufen. Öffne dazu den Collection Runner und setze die Anzahl der Durchläufe auf 3. Wenn du alles richtig gemacht hast, sollten alle Requests erfolgreich durchlaufen.

M450-LU08