

LU08.A03 - Bookshelf mit Authentifikation mit Bruno



Erweitere deine Bruno-Requests um Authentifikation, Autorisation und automatisierte Tests – nach Bruno-Best-Practices.

Zielbild

- **Ordnerstruktur** pro Fachgebiet (auth, books).
- **Kurz & klar benannte Requests** (kebab-case): login-admin.bru, read-book.bru, list-books.bru ...
- **Environments** je Zielsystem/Rolle statt Test-Steuerlogik im Script.
- **Token** nur **temporär** in einem **lokalen** Environment, das **nicht** eingechekkt wird.
- **Tests** sind klein, deterministisch, ohne Ablaufsteuerung.

Ausgangslage

Erweiterte API: <https://it.bzz.ch/book/ext/> (Login/Token, Rollen)

Rolle	Benutzername	Passwort	Berechtigung
admin	admin	admin	alle Funktionen
user	musterh	geheim	read, list
guest	-	-	nur login

Ohne gültiges Token → Rolle guest → nur login. Fehlende Berechtigung → **403**.

Projektstruktur (Beispiel)

```
bookshelf-bruno/  
  auth/  
    login-admin.bru  
    login-user.bru  
    login-invalid.bru  
  books/  
    read-book.bru  
    list-books.bru  
    save-book.bru      (optional)  
    delete-book.bru   (optional)  
  environments/
```

```
dev.json
dev.local.json      (in .gitignore!)
```

Hinweise

- dev.json: allgemeine, unverfängliche Variablen (baseUrl).
- dev.local.json: **Token** & sensible Werte (nicht ins Repo).
- In Bruno das passende **Environment aktivieren** (oben Environment).

Environments

environments/dev.json

```
{
  "name": "dev",
  "vars": {
    "baseUrl": "https://it.bzz.ch/book/ext"
  }
}
```

environments/dev.local.json (nicht einchecken)

```
{
  "name": "dev.local",
  "vars": {
    "token": "",
    "username": "admin",
    "password": "admin"
  }
}
```

Aktiviere zuerst dev, dann zusätzlich dev.local (oder führe beide Inhalte zu einem aktiven Environment zusammen - Hauptsache: token lebt **lokal**).

Requests (auth)

login-admin.bru

POST → {{baseUrl}}/login Body (JSON): {„username“: „admin“, „password“: „admin“}

Tests

```
test("Login admin -> 200 & token gesetzt", () => {
  expect(res.status).toEqual(200);
  const data = res.json();
  expect(data).to.have.property("token");
});
```

```
// Token nur im aktiven (lokalen) Environment halten:  
bru.setEnvVar("token", data.token, { persist: false });  
});
```

login-user.bru

POST → {{baseUrl}}/login Body (JSON): {„username“: „musterh“, „password“: „geheim“}

Tests

```
test("Login user -> 200 & token gesetzt", () => {  
  expect(res.status).toEqual(200);  
  const data = res.json();  
  bru.setEnvVar("token", data.token, { persist: false });  
});
```

login-invalid.bru

POST → {{baseUrl}}/login Body (JSON): {„username“: „wrong“, „password“: „wrong“}

Tests

```
test("Login invalid -> 401, kein token", () => {  
  expect(res.status).toEqual(401);  
});
```

Requests (books)

Gemeinsame Header: im Request Headers

- Key: Authorization
- Value: Bearer {{token}}

read-book.bru

GET → {{baseUrl}}/read/<book_uuid>

Tests (rollenbewusst, aber ohne Ablaufsteuerung)

```
test("Read Book – rollenabhängig", () => {  
  const s = res.status;  
  if (bru.getVar("token")) {  
    // Angemeldet: admin/user  
    expect(s).toEqual(200);  
    const b = res.json();  
    expect(b).to.have.property("book_uuid");  
  }  
});
```

```
    expect(b).to.have.property("title");
  } else {
    // guest (ohne Token)
    expect(s).to.equal(403);
  }
});
```

list-books.bru

GET → {{baseUrl}}/list

Tests

```
test("List Books – rollenabhängig", () => {
  const s = res.status;
  if (bru.getVar("token")) {
    expect(s).to.equal(200);
    const arr = res.json();
    expect(Array.isArray(arr)).to.equal(true);
    // ggf. bekannte Länge/Felder prüfen
  } else {
    expect(s).to.equal(403);
  }
});
```

Ausführung (Runner ohne Script-Steuerlogik)

Variante 1 (empfohlen & simpel): drei Läufe mit klaren Vorbedingungen

1. 'guest': 'token' in 'dev.local.json' leer lassen → 'list-books', 'read-book' ausführen → **403** erwartet.
2. 'user': 'login-user.bru' einmal senden → 'token' wird gesetzt → 'list/read' → **200**.
3. 'admin': 'login-admin.bru' einmal senden → 'list/read' → **200** (volle Rechte).

Variante 2 (Runner-Ordnung):

- Ordner auth vor books ausführen (login-*.bru zuerst), danach books/*.

Best Practice: Keine Ablaufsteuerung per Script (setNextRequest). Halte Tests kurz & deterministisch; die **Reihenfolge** steuert der **Runner/Ordner**.

Qualitäts-Checks (Best Practices)

- **DRY:** Gemeinsame Header (Authorization) in den jeweiligen Requests; Wiederholungen minimieren.
- **Kleine Assertions:** Je Testfile 1-3 präzise Assertions (Status, Struktur, zentrale Felder).
- **Keine Persistenz für Tokens:** `persist: false` verwenden; `dev.local.json` nicht committen.
- **Sprechende Namen:** Dateien/Ordner klar benennen (`login-admin`, `list-books`).
- **Reproduzierbar:** Jeder Request ist alleinstehend ausführbar (kein versteckter Zustand in Tests).

Abgabe

Packe dein Bruno-Projekt (Ordner mit allen `.bru`-Dateien und den **nicht-sensiblen** Environments) als **ZIP** und lade die **ZIP-Datei** in Moodle hoch. **Nicht** einchecken/abgeben: Dateien mit Token/Passwörtern (z. B. `dev.local.json`).

M450-LU08



Marcel Suter (angepasst)

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/modul/m450/learningunits/lu08/aufgaben/automationbruno?rev=1758183065>

Last update: **2025/09/18 10:11**

