Te Fac Programator

Bibliotech

9 Iulie 2020

INTRODUCERE

Pe măsură ce tehnologia avansează, din ce în ce mai multe procese sunt digitalizate. În acest sens, tema curentă propune dezvoltarea unei aplicații REST API pentru a gestiona cu ușurință atât cărțile unei biblioteci și utilizatorii acesteia, cât și interacțiunea dintre utilizatori și bibliotecă.

OBIECTIVE

1. Dezvoltarea unei aplicații REST API
2. Aprofundarea metodelor protocolului HTTP
3. Înțelegerea codurilor de status HTTP
4. Lucrul cu obiecte JSON

SPECIFICAȚII

În această aplicație vom avea 2 tipuri de utilizatori, Admin și Simple User. Aceștia, în funcție de permisiuni, vor fi capabili să adauge sau să elimine cărți, să împrumute cărți, să extindă termenul unui împrumut sau să returneze cărțile împrumutate.

De asemenea, utilizatorii pot primi sancțiuni pentru depășirea termenului de returnare.

În continuare, vom detalia funcționalitățile pe care biblioteca ar trebui să le aibă și pe care va trebui să le implementați.

Atenție! Pentru fiecare răspuns, încercați să returnați codul de status HTTP corect. O descriere a tuturor codurilor de status găsiți aici.

1. **Autentificare si înregistrare**

   Dorim sa avem functionalitati de înregistrare(register) si login(autentificare) pentru utilizatori. În acest sens, următoarele metode vor trebui implementate.

1.1. POST /register

Metoda /register are rolul de a crea un utilizator. Aceasta va primi 5 parametri:

- *first_name* - prenumele utilizatorului
- *last_name* - numele utilizatorului
- *email* - email-ul utilizatorului are rolul de username si va fi folosit pentru autentificare
- *password* - parola utilizatorului
- *type* - tipul utilizatorului va diferentia utilizatorii de tip Administrator de cei de tip Simple User

Răspunsul acestei metode va fi un obiect JSON ce conține *first_name, last_name, email* și *type* în cazul în care utilizatorul a fost creat cu succes și un obiect JSON ce va conține un mesaj de eroare pentru cazurile în care au fost trimise valori invalide ale parametrilor sau dacă utilizatorul există.

1.2. POST /login

Metoda /login are rolul de a autentifica un utilizator. Aceasta va primi 2 parametri:

- *email* - email-ul utilizatorului
- *password* - parola utilizatorului

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpul *auth_token*. *auth_token* este un string ce va fi folosit în apelarea metodelor ce necesită autentificare și are rolul de a identifica unic un utilizator. De asemenea, un mesaj de eroare va fi trimis în cazul în care un utilizator cu acei parametri nu va fi găsit.

2. Gestionarea cărților

În continuare sunt descrise metodele ce vor trebui implementate pentru a facilita adăugarea sau eliminarea cărților în/din bibliotecă și pentru verificarea lor.

2.1. POST /book (doar pentru Administratori)

Această metodă are rolul de a adăuga o carte în bibliotecă și primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *book_name* - numele cărții ce va fi introdusă

- *book_author* - autorul cărții
- *book_description* - descrierea cărții

Răspunsul acestei metode va fi un obiect JSON ce va conține următoarele câmpuri:

- *id* - id-ul cărții ce a fost introdusă
- *book_name* - numele cărții ce a fost introdusă
- *book_author* - autorul cărții
- *book_description* - descrierea cărții

Dacă această carte există deja în bibliotecă, un mesaj de eroare corespunzător va fi returnat.

2.2. **POST /books** (doar pentru Administratori)

Această metodă are rolul de a adăuga cărți în bibliotecă și primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *books* - o listă de obiecte JSON ce vor reprezenta cărțile

O carte va fi reprezentată printr-un obiect JSON cu următorii parametri:

- *book_name* - numele cărții ce va fi introdusă
- *book_author* - autorul cărții
- *book_description* - descrierea cărții

Răspunsul acestei metode va fi un obiect JSON format din câmpul:

- *books* - o listă de obiecte JSON ce reprezintă cărțile introduse

Un obiect JSON ce reprezintă o carte va conține:

- *id* - id-ul cărții ce a fost introdusă
- *book_name* - numele cărții ce a fost introdusă
- *book_author* - autorul cărții
- *book_description* - descrierea cărții

Dacă una dintre cărți există deja în bibliotecă, un mesaj de eroare corespunzător va fi returnat în locul acelei cărți.

2.3. GET /book

Metoda GET /book primește doi parametri:

- *auth_token* (opțional) - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *id* - identificatorul unic al cărții

și returnează un obiect JSON ce va conține următoarele câmpuri:

- *id* - o valoare ce identifică în mod unic o carte
- *title* - titlul cărții
- *author* - autorul cărții
- *description* - o descriere a cărții
- *status* - o valoare ce descrie starea acestei cărți (împrumutată/disponibilă/blocată)
- *rating* - un scor calculat în baza recenziilor oferite de utilizatori
- *reviews* - recenziile cărții

Câmpul *reviews* va fi reprezentat printr-un obiect JSON ce va conține recenziile oferite de utilizatori acestei cărți. O recenzie (review) va conține următoarele câmpuri:

- *rating* - scorul oferit cărții de autorul acestei recenzii
- *review* - aprecierea oferită de utilizator acestei cărți
- *author* – numele și prenumele autorului acestei recenzii

În cazul în care *auth_token* nu este trimis ca parametru atunci când se apelează metoda GET /books, câmpul *author* nu va fi trimis în recenzii.

2.4. GET /books

Metoda GET /books returnează un obiect JSON ce va conține câmpul:

- *books* - o listă de obiecte JSON ce reprezintă toate cărțile disponibile în bibliotecă.

O carte va fi reprezentată printr-un obiect JSON ce conține următoarele câmpuri:

- *id* - o valoare ce identifică în mod unic o carte
- *title* - titlul cărții
- *author* - autorul cărții
- *description* - o descriere a cărții

- *status* - o valoare ce descrie starea acestei cărți (împrumutată/disponibilă/blocată)
- *rating* - un scor calculat în baza recenziilor oferite de utilizatori

În cazul în care nicio carte nu există în bibliotecă, această metodă va întoarce un mesaj corespunzător.

3. Interacțiunea cu biblioteca

Pentru ca utilizatorii să poată interacționa cu biblioteca, vor trebui implementate metodele prezentate în continuare.

3.1. POST /transaction

Metoda POST /transaction va crea o cerere de a împrumuta o carte. Ea primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *book_id* - identificatorul unic al cărții
- *borrow_time* - o valoare ce indică numărul de zile pe durata cărora cartea va fi împrumutată, cu următoarea constrângere: 1 ≤ borrow_time ≤ 20

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpurile:

- *success* - un mesaj corespunzător
- *transaction_id* – id-ul acestei tranzacții

în cazul în care această carte este disponibilă, iar în caz contrar, acesta va conține câmpul *error* cu un mesaj corespunzător.

Atenție! Un utilizator este limitat la maximum 5 tranzacții într-un moment de timp.

3.2. GET /transaction

Metoda GET /transaction are rolul de a afișa informațiile unei tranzacții (statusul unui împurumut) și primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *transaction_id* - id-ul tranzacției

Răspunsul acestei metode îl reprezintă un obiect JSON format din următoarele câmpuri:

- *book_id* - id-ul cărții care face obiectul acestei tranzacții
- *borrow_time* - durata inițială a împrumutului

- *remaining_time* - durata până la termen
- *number_of_extensions* - numărul de extinderi al termenului de predare
- *status* - statusul acestei tranzacții (în desfășurare/încheiată/în întârziere)

Dacă o tranzacție cu id-ul specificat nu există, un mesaj de eroare corespunzător va fi returnat.

3.3. GET /transactions

Metoda GET /transactions are rolul de a afișa istoricul tranzacțiilor. Dacă această metodă este apelată de un utilizator simplu, ea va returna doar tranzacțiile acestui utilizator. Dacă, însă, un administrator apelează metoda, aceasta va returna toate tranzacțiile bibliotecii.

Metoda primește următorul parametru:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul

Răspunsul acestei metode îl reprezintă un obiect JSON format din câmpul:

- *transactions* - o listă de obiecte JSON ce conțin informații despre tranzacții

Un obiect JSON ce conține o tranzacție este format din câmpurile:

- *transaction_id* - id-ul tranzacției
- *book_id* - id-ul cărții care face obiectul acestei tranzacții
- *book_name* - numele cărții
- *status* - statusul acestei tranzacții (în desfășurare/încheiată/în întârziere)

Dacă utilizatorul nu a efectuat nicio tranzacție, un mesaj corespunzător va fi returnat.

3.4. POST /extend

Metoda POST /extend are rolul de a extinde timpul de împrumut al acestei cărți. Ea primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *transaction_id* - identificatorul unic al tranzacției (împrumutul acestei cărți)
- *extend_time* - o valoare utilizată pentru a extinde timpul de împrumut, cu următoarele constrângeri: $1 \leq extend\_time \leq 5$ și extinderea termenului unei tranzacții poate fi efectuată de maxim 2 ori.

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpul *success* - un mesaj corespunzător în cazul în care o extindere este disponibilă, iar în caz contrar, acesta va conține câmpul *error* cu un mesaj corespunzător.

3.5. POST /return

Metoda POST /return va crea o cerere de a returna o carte. Aceasta va primi următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *transaction_id* - identificatorul unic al tranzacției (împrumutul acestei cărți)

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpul *success* - un mesaj corespunzător în cazul în care această tranzacție este disponibilă, iar în caz contrar, acesta va conține câmpul *error* cu un mesaj corespunzător.

Dacă această metodă va fi apelată după termenul de returnare al cărții, se vor aplica sancțiuni:

- prima abatere va rezulta într-o atenționare - valoarea câmpului succes va conține această atenționare
- a doua abatere va rezulta în reducerea numărului de extinderi ale termenului unei tranzacții la o singură extindere pentru acest utilizator
- a treia abatere va rezulta în eliminarea numărului de extinderi ale termenului unei tranzacții pentru acest utilizator
- cea de-a patra abatere va rezulta în blocarea capabilității de a împrumuta cărți a acestui utilizator pentru o perioadă de 30 de zile

Numărul de abateri va fi decrementat după o perioadă de 30 de zile. De exemplu, dacă un utilizator are 4 abateri, după 30 de zile acest număr va fi decrementat și va ajunge la 3 abateri, adică în incapabilitatea de a extinde termenul pentru o carte.

Atenție! Apelarea metodei de returnare a unei cărți nu completează returnarea acesteia, ci doare creează o cerere de returnare. Această cerere va fi acceptată de un utilizator de tip Administrator.

3.6. GET /returns (doar pentru Administratori)

Metoda GET /returns are rolul de a prezenta cererile de returnare existente. Ea primește un singur parametru:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul

După verificarea că utilizatorul care apelează această metodă este un administrator, metoda va întoarce un obiect JSON ce va conține cererile existente de returnare, format din:

- *return_requests* - un obiect JSON ce va conține cererile de returnare

O cerere de returnare este formată din următorii parametri:

- *id* - id-ul cererii de returnare
- *transaction_id* - id-ul tranzacției ce se dorește a fi încheiată

Dacă nu există nicio cerere de returnare existentă, această metodă va întoarce un mesaj corespunzător.

3.7. **POST /return/end** (doar pentru Administratori)

Metoda POST /return/end va încheia o cerere de a returna o carte. Ea va primi următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *return_id* - id-ul cererii de returnare

Atenție! Încheierea unei cereri de returnare va încheia și tranzacția referențiată de aceasta.

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpul *success* - un mesaj corespunzător, iar în cazul în care nu se poate confirma această cerere, răspunsul va conține câmpul *error* cu un mesaj corespunzător.

3.8. **POST /review**

Metoda POST /review este folosită pentru a crea o recenzie a unei cărți. Aceasta primește următorii parametri:

- *auth_token* - stringul primit în urma autentificării, ce identifică unic utilizatorul
- *book_id* - identificatorul unic al cărții
- *rating* - un scor între 1 și 5
- *text* - un string reprezentând opinia unui utilizator asupra cărții citite

Răspunsul acestei metode va fi un obiect JSON ce va conține câmpul *success* - un mesaj corespunzător, iar în cazul în care nu se poate confirma această cerere, răspunsul va conține câmpul *error* cu un mesaj corespunzător.

BAREM

1. Înregistrare și autentificare — 10 puncte

1.1.	POST /register	5 puncte
1.2.	POST /login	5 puncte

2. Managerierea cărților — 15 puncte

2.1.	POST /book	5 puncte
2.2.	POST /books	2.5 puncte
2.3.	GET /book	5 puncte
2.4.	GET /books	2.5 puncte

3. Interacțiunea cu biblioteca — 65 puncte

3.1.	POST /transaction	10 puncte
3.2.	GET /transaction	5 puncte
3.3.	GET /transactions	10 puncte
3.4.	POST /extend	10 puncte
3.5.	POST /return	10 puncte
3.6.	GET /returns	10 puncte
3.7.	POST /return/end	5 puncte
3.8.	POST /review	5 puncte

4. Formatare — 10 puncte

4.1.	Lizibilitatea și structurarea codului	5 puncte

- funcții și variabile numite corespunzător
- spațierea codului
- separarea logicii (Separation of Concerns)

4.2. README și comentarii 5 puncte

Creați un fișier text numit README în care veți detalia:

- funcțiile implementate - efectiv cum ați gândit codul și cum funcționează

Adăugați comentarii în cod acolo unde considerați necesar (ex: funcții lungi).

5. BONUS 20 puncte

Bonusul constă în implementarea unei soluții de stocare persistentă a datelor. Acest lucru presupune păstrarea datelor între rulări consecutive ale aplicației.

Succes!

EXEMPLU DE FUNCȚIONARE

1. Creăm un utilizator de tip administrator:

   Apelăm metoda POST /register cu următorii parametrii:

```
{
    "first_name": "Clopotel",
    "last_name": "Admin",
    "email": "clopotel.admin@gmail.com",
    "password": "safepass1!",
    "type": 1
}
```

   Aceasta întoarce următorul răspuns:

```
{
    "first_name": "Clopotel",
    "last_name": "Admin",
    "email": "clopotel.admin@gmail.com",
    "type": 1
}
```

   Clopoțel dorește să se autentifice, așa că apelează metoda POST /login cu următorii parametri:

```
{
    "email": "clopotel.admin@gmail.com",
    "password": "safepass1!"
}
```

și primește răspunsul:

```
{
    "auth_token": "1c23skm803",
}
```

Clopoțel introduce o carte apeland metoda POST /book cu următorii parametri:

```
{
    "auth_token": "1c23skm803",
    "book_name": "Coronavirus",
    "book_author": "Wang Fang",
    "description": "Vine starea de urgenta iar stati chill."
}
```

În urma acestui apel, el primește răspunsul următor:

```
{
    "id": 1,
    "book_name": "Coronavirus",
    "book_author": "Wang Fang",
    "description": "Vine starea de urgenta din nou stati chill."
}
```

În continuare creăm nu utilizator simplu, apelând metoda POST /register cu următorii parametri:

```
{
    "first_name": "Romeo",
    "last_name": "Programache",
    "email": "romeo.programache@gmail.com",
    "password": "supersafepass1!",
    "type": 0
}
```

Am primit următorul răspuns:

```
{
    "first_name": "Romeo",
    "last_name": "Programache",
    "email": "romeo.programache@gmail.com",
    "type": 0
}
```

În urma autentificării, Romeo primește codul său de autentificare:

```
 "auth_token": "84dp23qrh0"
```

Acesta dorește să verifice cărțile existente în bibliotecă apelând GET /books:

```
{
    "books":[
```

```
        {
            "id": 1,
            "title": "Coronavirus",
            "author": "Wang Fang",
            "description": "Vine starea de urgenta iar stati
chill.",
            "status": 0,
            "rating": 0
        }
    ]
}
```


În continuare, Romeo dorește să împrumute această carte, așa că apelează metoda POST /transaction:


```
{
    "auth_token": "84dp23qrh0",
    "book_id": 1,
    "borrow_time": 10
}
```


După ce a apelat metoda de împrumut, Romeo primește răspunsul:


```
{
    "success": "Super, ai împrumutat Coronavirus!",
    "transaction_id": 1
}
```

După 9 zile, Romeo realizează ca nu poate termina cartea la timp, așa că face o cerere pentru a extinde timpul de împrumut apelând POST /extend

```
{
    "auth_token": "84dp23qrh0",
    "transaction_id": 1,
    "extend_time": 3
}
```

și primește răspunsul:

```
{
    "success": "Ai extins termenul cu 5 zile!"
}
```

Romeo realizează că a ajuns în a 16-a zi de când a împrumutat cartea și nu a făcut încă o cerere de returnare, așa că pune imediat mâna pe Postman și apelează POST /return:

```
{
    "auth_token": "84dp23qrh0",
    "transaction_id": 1,
}
```

Din păcate, acesta a depășit termenul fără să facă o nouă cerere de extindere a termenului așa că primește următorul răspuns:

```
{
      "success": "Ai creat o cerere de returnare cu succes, însă ai depășit termenul de returnare. Fiind la prima abatere, funcționalitățile bibliotecii nu vor fi restricționate, însă ai acumulat un punct penalizare care va expira peste 30 de zile."
}
```

Clopoțel, administratorul, verifică cererile de returnare apelând GET /returns:

```
{
    "auth_token": "1c23skm803",
}
```

și primește următorul răspuns:

```
{
    "return_requests":[
        {
            "id": 1,
            "transaction_id": 1
        }
    ]
}
```

El se gândește să afle mai multe informații depsre această tranzacție, așa că apelează metoda GET /transaction:

```
{
    "auth_token": "1c23skm803",
    "transaction_id": 1,
}
```

și primește răspunsul:

```
{
    "book_id": 1,
    "borrow_time": 10,
    "remaining_time": -1,
```

```
    "number_of_extensions": 1,

    "status": 2

}
```

Între timp, Romeo ajunge cu cartea la bibliotecă, așadar Clopoțel confirmă cererea de returnare apelând POST /return/end:

```
{

    "auth_token": "1c23skm803",

    "return_id": 1,

}
```

și primește răspunsul:

```
{

      "success": "Super! Cartea a fost înapoiată."

}
```