Tartalom

[Backend futtatása: 2](#_Toc195643086)

[Felhasználói: 2](#_Toc195643087)

[Bevezetés: 2](#_Toc195643088)

[Api/login (POST) 3](#_Toc195643089)

[Api/register (POST) 4](#_Toc195643090)

[Api/verify-account (POST) 5](#_Toc195643091)

[Api/forgot-password (POST) 5](#_Toc195643092)

[Api/user (GET) 6](#_Toc195643093)

[Api/user (PUT) 7](#_Toc195643094)

[Api/user (DELETE) 7](#_Toc195643095)

[Api/finalize-registration (PUT) 8](#_Toc195643096)

[Api/change-password (PUT) 9](#_Toc195643097)

[Api/top-borrowings/{limit} (GET) 9](#_Toc195643098)

[Api/borrowings (GET) 10](#_Toc195643099)

[Api/available-books (GET) 11](#_Toc195643100)

[Api/upload-img (POST) 12](#_Toc195643101)

[img/{ISBN} (GET) 13](#_Toc195643102)

[Api/books (GET) 15](#_Toc195643103)

[Api/all-(books, users, borrowings) (GET) 18](#_Toc195643104)

[Api/reserve (GET) 18](#_Toc195643105)

[Api/reserve (POST) 18](#_Toc195643106)

[Api/reserve/{ISBN} (DELETE) 19](#_Toc195643107)

[**Fejlesztői:** 20](#_Toc195643108)

[Router.php 20](#_Toc195643109)

[Helper.php 20](#_Toc195643110)

[SendEmail.php 21](#_Toc195643111)

[BooksTable.php 22](#_Toc195643112)

[BorrowingsTable.php 22](#_Toc195643113)

[ReservationTable.php 23](#_Toc195643114)

[Table.php 24](#_Toc195643115)

[UserTable.php 25](#_Toc195643116)

[BooksController.php 26](#_Toc195643117)

[UserController.php 26](#_Toc195643118)

[Response.php 27](#_Toc195643119)

[User.php 28](#_Toc195643120)

[Model.php 29](#_Toc195643121)

[Image.php 30](#_Toc195643122)

[Books.php 31](#_Toc195643123)

[Borrowing.php 31](#_Toc195643124)

[ReservationController.php 32](#_Toc195643125)

[ImageController.php 33](#_Toc195643126)

[BorrowingsController.php 33](#_Toc195643127)

[Token.php 34](#_Toc195643128)

[router.php 35](#_Toc195643129)

Backend Dokumentáció

# Backend futtatása:

A backend servert a start\_backend .bat fájl elindításával lehet futtatni vagy a   
*php -S 127.0.0.1:8000 router.php* paranccsal. Amennyiben nem elérhető a globális php parancs akkor "elérési\út\xampp\php\php.exe" -S localhost:8000 router.php

# Felhasználói:

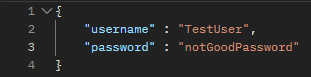
## Bevezetés:

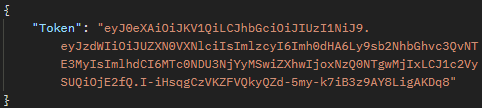
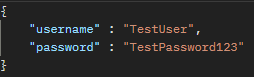
Az API (Application Programming Interface) különböző végpontokkal rendelkezik. A kérés típusától függően lehet GET, POST, PUT, DELETE. Azonos végpont cím rendelkezhet más-más kérési típussal. A végpontok a következőek (http://localhost:8000):

* /api/login
* /api/register
* /api/verify-account
* /api/forgot-password
* /api/user
  + get
  + put
  + delete
* /api/finalize-registration
* /api/change-password
* /api/top-borrowings/{limit}
* /api/borrowings
* /api/available-books
* /api/upload-img
* /img/{ISBN}
* /api/books
* /api/all-books
* /api/all-users
* /api/all-borrowings
* /api/reserve
  + get
  + post
  + delete

## Api/login (POST)

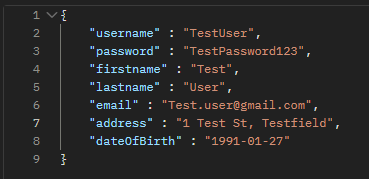
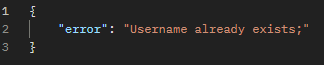
A bejelentkezés POST kérési metódust vár. A végpont megcímzésekor 2 adatot vár a kérési metódus törzsében json formátumban. Az egyik a *username* a másik a *password*. Ezek az adatok alapján eldönti a szoftver, hogy a megadott adatok megegyeznek-e az adatbázisban lévőkkel. Amennyiben nem vagy rossz formátumban vannak az adatok, akkor különböző hibaüzeneteket írhat ki json formátumban.

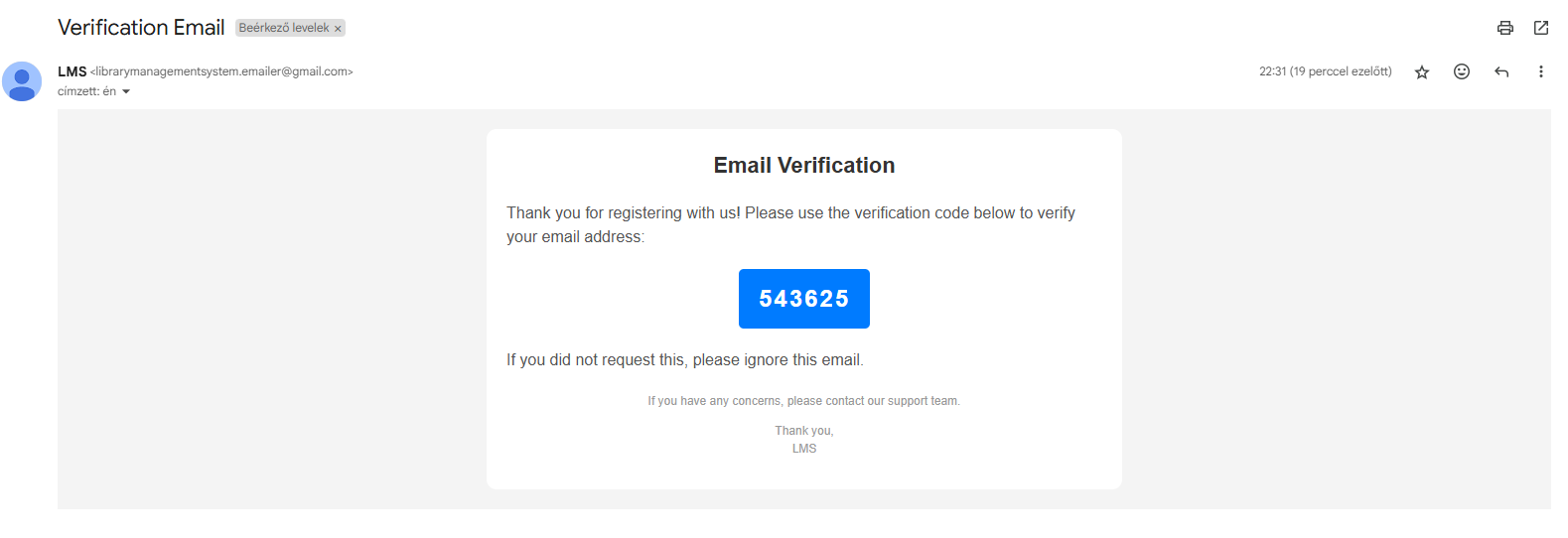
  
  
Helyes adatok megadásával egy tokent küld vissza.

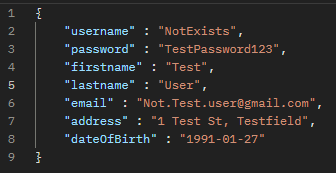


## Api/register (POST)

A regisztrációs végpont új felhasználó létrehozására szolgál. A végpont a kérés törzsében JSON formátumban kapott adatokat vár, és ellenőrzi azok helyességét. A sikeres regisztráció esetén a felhasználó adatai bekerülnek az adatbázisba. Hibás vagy hiányzó adatok esetén JSON formátumban hibaüzenetet küld vissza.

Helyes adatok megadásakor a megadott e-mail címre érkezik egy új levél, ami tartalmazza az ellenőrző kódot. 

## Api/verify-account (POST)

A verifikációs végpontot akkor használják, amikor egy felhasználó ellenőrzi a regisztrációját egy visszaigazoló kód megadásával. A végpont a kérés törzsében JSON formátumban kapott *verificationCode* adatot vár. Ha a kód helyes, frissíti a felhasználó állapotát az adatbázisban. Ha a kód érvénytelen, hibaüzenetet küld vissza.

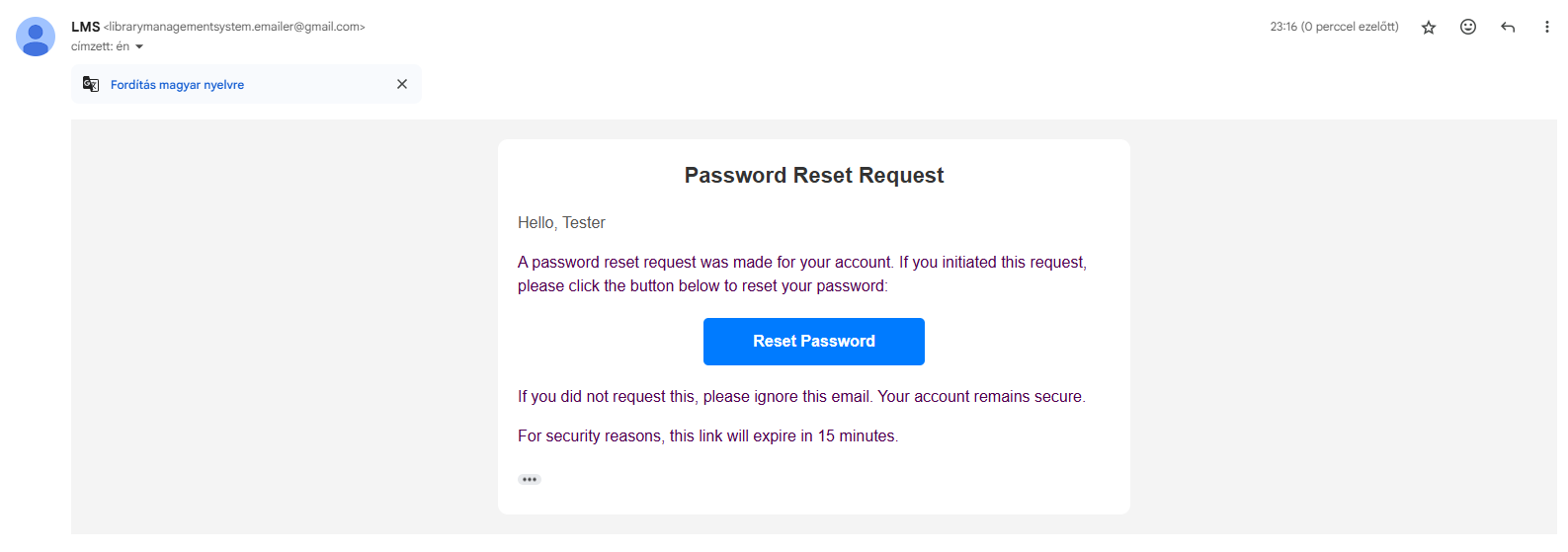
Helyes adatokkal:

## Api/forgot-password (POST)

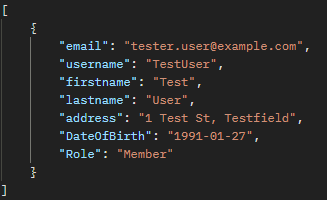
Ez a végpont lehetővé teszi a felhasználó számára, hogy elfelejtett jelszót kérjen. Az e-mail cím alapján egy egyedi token kerül generálásra és elküldésre a felhasználó számára egy visszaállító linken keresztül.

 **** **** 

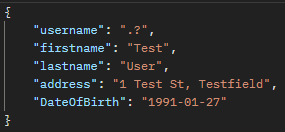
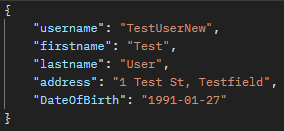
Helyes adatokkal:  
  
  

## Api/user (GET)

Ennek a végpontnak az eléréséhez szükséges az azonosítás. A bejelentkezéskor kapott token alapján azonosít a backend. Ez a token a header-ben kell, hogy legyen. A felhasználó adatait listázza ki.

## Api/user (PUT)

A tokennel való azonosítás után a végpont megvizsgálja az adatokat. Ami üres tehát nem módosult azt az adatot nem viszi fel az adatbázisba. A felhasználó a jelszavát is módosíthatja, de csak akkor, ha megadta a jelenlegi jelszavát, viszont az e-mail címét nem tudja módosítani. A felhasználó csak olyan adatokat adhat meg amiket megenged a validáció amennyiben ez a feltétel nem teljesül hibaüzenet kerül elküldésre.   
   
   
Helyes adatokkal:  
  
 

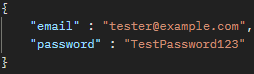
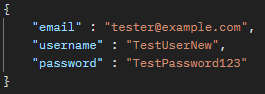
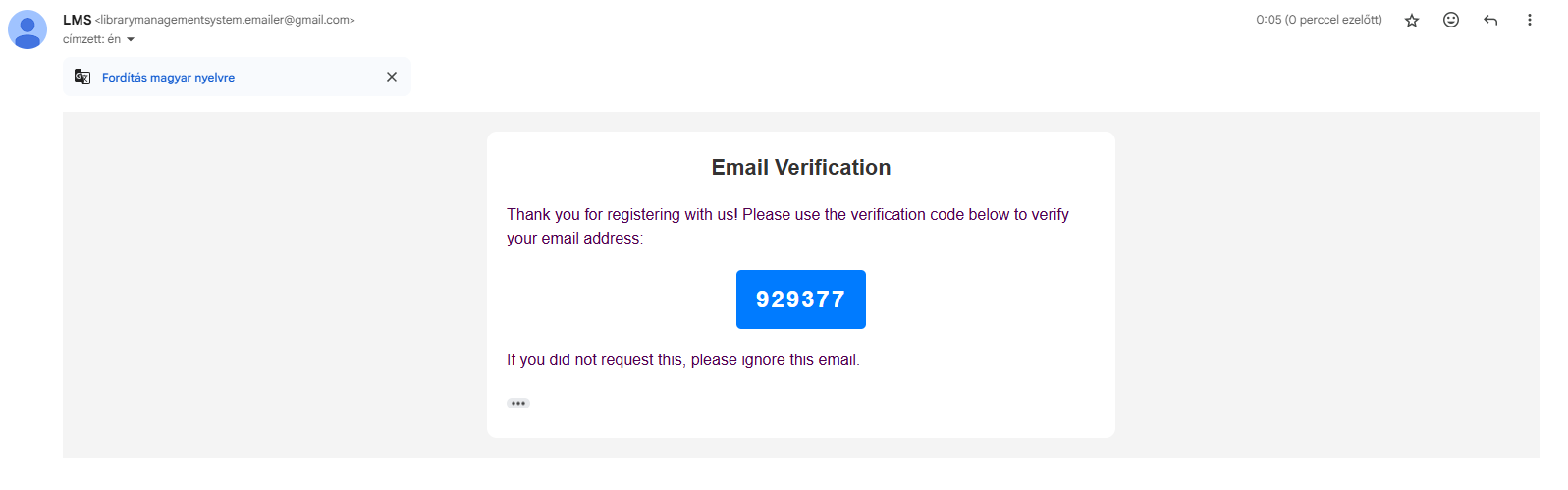
## Api/user (DELETE)

A felhasználói tokent dekódolva az ID alapján kitörli az adott felhasználót. A token ha már nem létező ID-val rendelkező felhasználóra mutat akkor nem fut le a az adatbázis módosítás. Ilyenkor egy hibaüzenet látható.

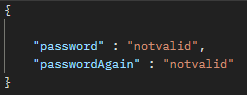
   
   

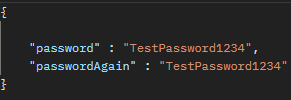

A kérés futtatása újra a törlés után a következő eredményre jut:  
   


## Api/finalize-registration (PUT)

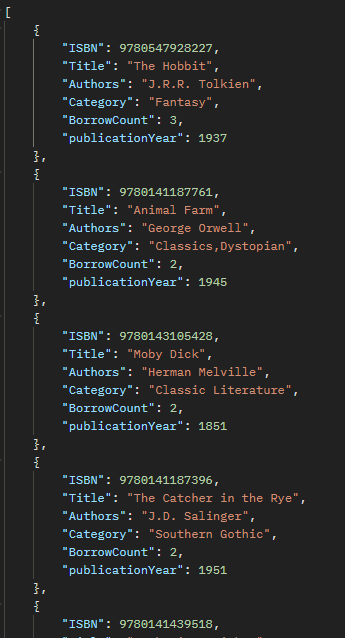
Ez a végpont csak azoknak a felhasználóknak elérhető, akik nem regisztráltak a weboldalon, de már regisztráltak a könyvtárban. Az kérés törzsében kell *email*, *username* és *password* azonosító. Ezek hiányában hibaüzenet várható. A password a könyvtáros által felvitt felhasználónak a generált jelszava. A felhasználónév és jelszó megadásával bejelentkezik a felhasználó, de csak akkor, ha már nem regisztrált az oldalon, ha a bejelentkezés sikeres akkor a felhasználónak módosul az e-mail címe. A sikeres regisztrációt egy e-mail is jelzi, ami már a regisztrációnál is megjelent, tehát az ellenőrző levél.  
    
   
Helyes adatokkal:  
   

## Api/change-password (PUT)

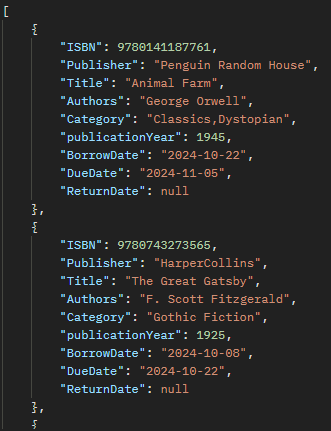
A végpont csak tokennel elérhető. Ez a kérés várhatóan csak akkor fog megtörténni, ha a felhasználó kérte, hogy módosíthassa a jelszavát. A várt adatok a *password* és a *passwordAgain.* Természetesen az új jelszónak meg kell felelnie az elvárt szabályoknak, ha ez nem történik meg akkor hibaüzenetbe fut a felhasználó.  
    
 

Helyes adatokkal:  
  
 

## Api/top-borrowings/{limit} (GET)

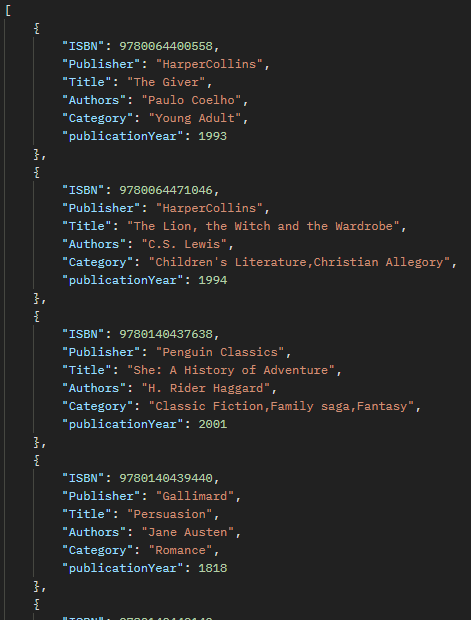
A végpont megcímzésekor szükséges egy számot megadni. A szám azt jelenti, hogy a lekérdezésben ez a szám fog szerepelni, mint limit. A lekérdezés, ha jó akkor a könyv(ek) adataival fog visszatérni, ami ISBN, Title, Authors, Category, BorrowCount, publicationYear. A BorrowCount az az érték ahányszor kikölcsönözték a könyvet. A megjelenítés legtöbbször kölcsönzött könyvvel kezdi a json-t  
  

## Api/borrowings (GET)

A végpont elérésekor kilistázza a felhasználóhoz tartozó összes kölcsönzést. Az azonosítás tokennel történik. A könyv(ek) adatait és a kölcsönzéssel kapcsolatos információkat íratja ki.  

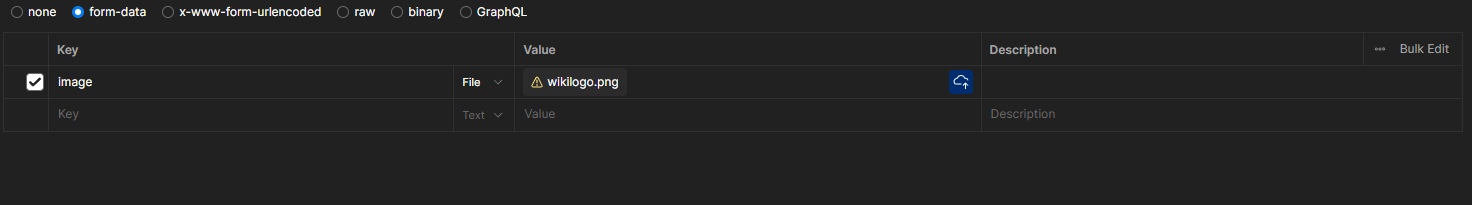
## Api/available-books (GET)

A végpont eléréséhez nem szükséges az azonosítás. Kilistázza az összes elérhető könyvet és azok adatait. Egy könyv akkor nem elérhető, ha valaki kikölcsönözte vagy lefoglalta.

## Api/upload-img (POST)

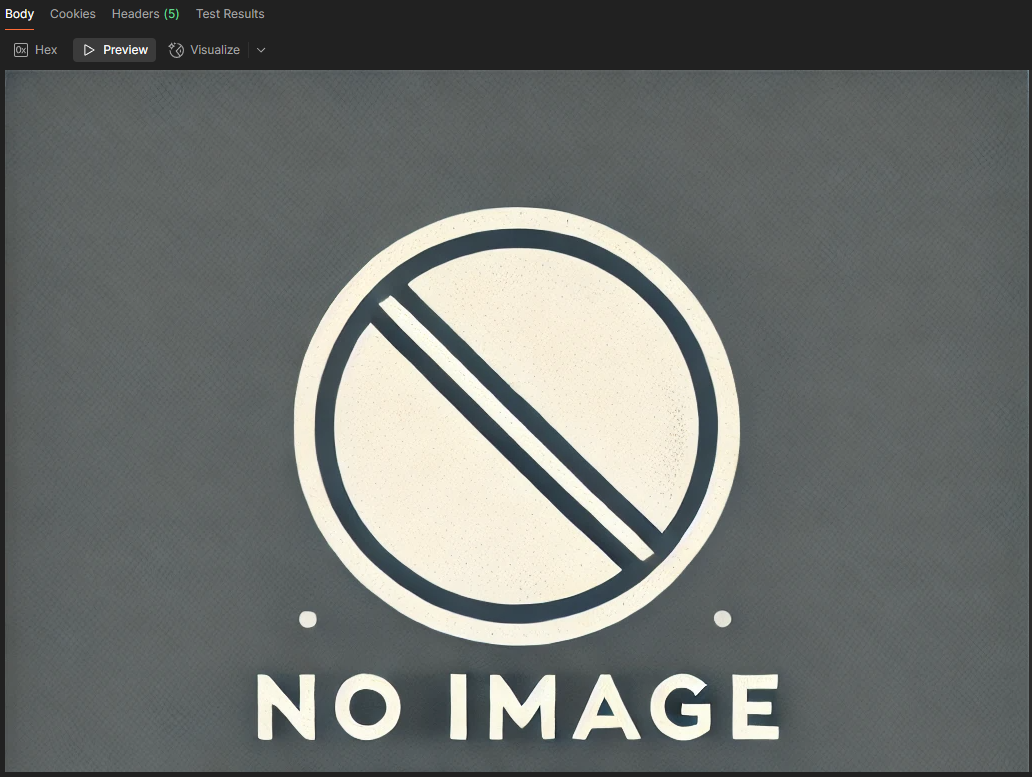
A kép feltöltéséhez a /api/upload-img végpontot kell megcímezni. megvizsgálásra kerül, hogy van-e a kérésben kép, amennyiben igen akkor a képet további ellenörzések után feltölti a *booksImages* mappába. A kép nevének az ISBN-nek kell lennie és a kiterjesztés CSAK jpg vagy png vagy jpeg lehet. A backend nem dolgozik fel más formátumot.  
  

Helyesen feltöltött képpel:  
 

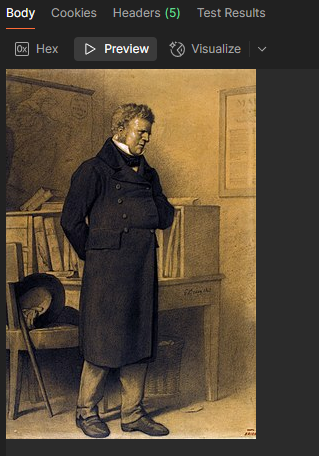
## img/{ISBN} (GET)

A végponthoz nem kell azonosítás, mivel ezeket az adatokat bárki elérheti. A várt adat egy az url-be beírt *ISBN* szám, ami a könyvhöz tartozó képet tölti be, ha a könyv nem található a *booksImages* mappában, akkor a missing.png fájlt tölti be alapértelmezetten. Abban az esetben, ha az ISBN nem megfelelő formátumban kerül átadásra egy hibakód kerül átadásra. A fájl formátumát nem kell megadni, mert a végpont automatikusan lekéri a 3 opció közül az összeset, és ha talált egy fájlt akkor azt tölti be.  
  


Nem található kép:



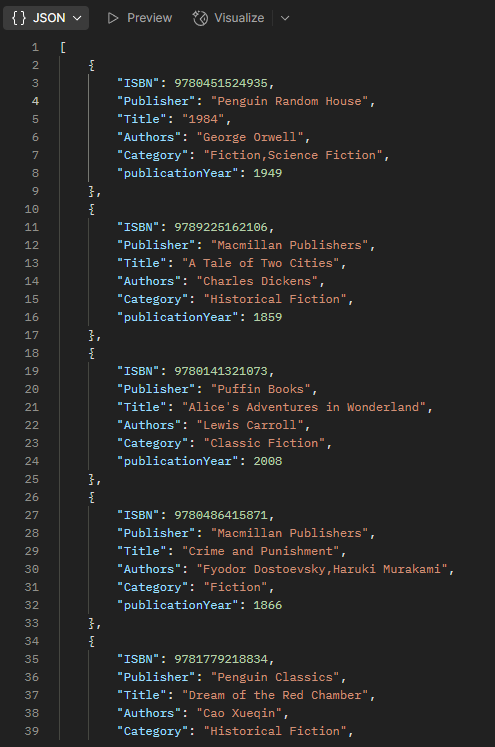
Helyes adatokkal:



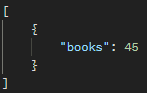
## Api/books (GET)

Az api/books végpont a könyveket listázza ki. Az url-be több adat is megadható például *api/books/category/fiction,* így az összes olyan könyv kerül kilistázásra, amelynek a kategóriája *fiction.* A végpont elérésekor semmi adatot megadva az összes könyvet és adatait küldi vissza json formátumban.

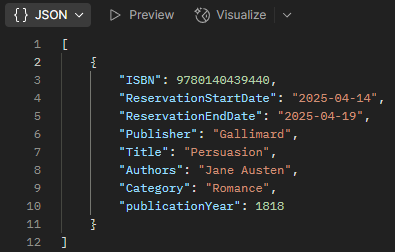


Adatok megadásával:  


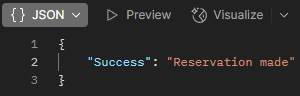
## Api/all-(books, users, borrowings) (GET)

A fenti végpontok elérése azonos feladat elvégzésére szolgál, csak más-más táblában valósul meg. A feladata, hogy egy értékkel térjen vissza, ami megszámolt adatot tartalmazza például a */api/all-books* elérésekor azt számolja meg, hogy mennyi a jelenlegi könyveknek a száma.  
  


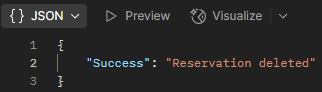
## Api/reserve (GET)

A végpont elérésekor azonosítani kell a felhasználót egy tokennel. A megadott token alapján lekérdezésre kerül az összes olyan könyv, amit az adott felhasználó foglalt le. Az összes lefoglalás nem haladhatja meg a 3-at.  


## Api/reserve (POST)

A végpont megcímzésekor szükséges elküldeni a kérés törzsében az ISBN-t. A végpont tokent használ az azonosításhoz, anélkül nem is lehet elérni a végpontot. Az eléréskor a backend megvizsgálja az adatokat, ha az adatok nem jók az alapján hibaüzenetet küld vissza. A helyes adatok megadásával az adatbázisban lefoglalja a felhasználó részére a könyvet. A könyv, addig amíg le nem jár a foglalás nem lesz látható az elérhető könyvek között.  
  
  


## Api/reserve/{ISBN} (DELETE)

Az api/reserve/{ISBN} végpont eléréséhez azonosítás szükséges tokennel. Az ISBN-t az url-be kell beírni, aminek meg kell felelnie a feltételeknek. Az ISBN-nek helyes formátumúnak kell lennie és a felhasználóhoz kell tartoznia, ha ez nem teljesül hibaüzenet kerül visszaküldésre a backendtől.  


# **Fejlesztői:**

## Router.php

Ez a fájl a `Router` osztályt definiálja, amely a HTTP kérések kezeléséért  
és a megfelelő vezérlőkhöz (controller) való irányításáért felelős.  
Támogatja a többféle HTTP metódust (GET, POST, PUT, DELETE), valamint biztosítja  
a hitelesítést, fájlkezelést és az URL-ekből történő változók kinyerését.

Funkciók:  
- `post()`: POST kérések kezelése egy adott végpontra.  
- `get()`: GET kérések kezelése egy adott végpontra, opcionális URL-változó kinyeréssel.  
- `put()`: PUT kérések kezelése egy adott végpontra.  
- `delete()`: DELETE kérések kezelése egy adott végpontra.  
- Hitelesítés: Token alapú hitelesítés támogatása védett útvonalakhoz.  
- URL feldolgozás: Változók és adatok kinyerése az URL-ből dinamikus végpontokhoz.  
- Segédfüggvények integrálása: Segédfüggvények használata bemenetellenőrzéshez és a kérés törzsének feldolgozásához.

Használat:  
- Az útvonalakat a statikus metódusok (`post`, `get`, `put`, `delete`) meghívásával lehet definiálni,  
megadva a végpontot, a vezérlőt, a függvényt és opcionálisan további paramétereket (pl. hitelesítés).  
- A vezérlőknek implementálniuk kell a megadott függvényeket, hogy kezelni tudják az irányított kéréseket.

Függőségek:  
- `Helper\Helper`: Segédfüggvényeket biztosít a bemenetek ellenőrzéséhez és a kérés törzsének feldolgozásához.  
- `App\Authorize\Token`: A tokenek ellenőrzéséért felelős a hitelesítés során.

## Helper.php

Ez a fájl a `Helper` osztályt definiálja, amely segédfüggvényeket biztosít a bemenetek  
érvényesítéséhez és a kérés törzsének feldolgozásához. Célja a HTTP kérésekből érkező adatok  
épségének és biztonságának biztosítása.

Funkciók:  
- `validateTheInput()`: Egyetlen bemeneti érték érvényesítése és megtisztítása.  
- `validateTheInputArray()`: Több bemeneti értékből álló tömb érvényesítése és megtisztítása.  
- `getPostBody()`: A nyers POST kérés törzsének beolvasása és dekódolása, opcionálisan JSON-formátum ellenőrzésével.

Használat:  
- A `validateTheInput()` függvényt egyéni bemeneti értékek megtisztítására használd.  
- A `validateTheInputArray()` segítségével tömbök, például lekérdezési paraméterek vagy URL-változók tisztíthatók.  
- A `getPostBody()` használható POST kérések JSON törzsének értelmezésére és érvényesítésére.

Függőségek:  
- `ApiResponse\Response`: A HTTP hibaválaszok kezelésére szolgál, ha az érvényesítés sikertelen.

## SendEmail.php

Ez a fájl a PHPMailer alapfájlja. A forráskód elérhető itt: <https://github.com/PHPMailer/PHPMailer>  
Ez a fájl kezeli az e-mailek küldését. A projekt e-mail címe: [librarymanagementsystem.emailer@gmail.com](mailto:librarymanagementsystem.emailer@gmail.com), így minden, a rendszer által küldött e-mail erről a címről fog érkezni.

Funkciók:  
A sendEmail függvény 4 paramétert vár:  
- `sendEmail()`: E-mail küldése a szoftver e-mail címéről

Használat:  
- A `sendEmail()`: függvényt egy e-mail elküldésére használd, ehhez definiálj egy HTML szöveget előtte

Függőségek:  
- PHPMailer\PHPMailer az e-mail küldéséhez szükséges funkciókat és osztályokat tartalmazza   
- `ApiResponse\Response`: A HTTP hibaválaszok kezelésére szolgál, ha az érvényesítés sikertelen.

## BooksTable.php

Ez a fájl a `BooksTable` osztályt definiálja, amely a `Table` osztályból származik, és  
kifejezetten a könyvekkel kapcsolatos adatok kezelésére és lekérdezésére biztosít metódusokat.

Funkciók:  
- `selectByParams()`: Könyvek lekérdezése megadott mezők, operátorok, értékek és típusok alapján.  
- `countAllBooks()`: Az adatbázisban található összes könyv megszámlálása.  
- `selectByISBN()`: Egy könyv adatainak lekérdezése ISBN alapján.  
- `availableBooks()`: Egy adott ISBN-hez tartozó könyv elérhetőségének ellenőrzése.  
- `selectAvailableBooks()`: Azoknak a könyveknek a listázása, amelyek jelenleg elérhetők (nincsenek kikölcsönözve vagy lefoglalva).

Használat:  
- Ezekkel a metódusokkal lehet kapcsolatba lépni a `books` táblával és a hozzá kapcsolódó táblákkal (pl. `publishers`, `authors`, `categories`).  
- A `selectByParams()` és `selectAvailableBooks()` metódusok támogatják az összetett lekérdezéseket join-olással és csoportosítással.

Függőségek:  
- A `Table` osztályból származik, amely az alap lekérdezésépítési funkciókat biztosítja.  
- Használja a kapcsolódó adatbázistáblákat, mint például: `books`, `publishers`, `authors`, `categories`, `borrowings` és `reservations`.

## BorrowingsTable.php

Ez a fájl a `BorrowingsTable` osztályt definiálja, amely a `Table` osztályból származik,  
és kifejezetten a kölcsönzésekkel kapcsolatos adatok kezelésére és lekérdezésére biztosít metódusokat.

Funkciók:  
- `allBorrowings()`: Az összes kölcsönzés megszámlálása az adatbázisban.  
- `bookInBorrowings()`: Annak ellenőrzése, hogy egy adott könyvet (ISBN alapján) jelenleg kölcsönözték-e, és még nem került vissza.  
- `selectMyBooks()`: Egy adott felhasználó által kölcsönzött könyvek listázása, a kölcsönzés részleteivel együtt.  
- `selectTopBorrowedBook()`: A legtöbbször kölcsönzött könyvek lekérdezése, megadott darabszámig korlátozva.

Használat:  
- Ezekkel a metódusokkal lehet kapcsolatba lépni a `borrowings` táblával és a kapcsolódó táblákkal (pl. `books`, `authors`, `categories`).  
- A `selectMyBooks()` és `selectTopBorrowedBook()` metódusok támogatják az összetett lekérdezéseket join-okkal, csoportosítással és rendezéssel.

Függőségek:  
- A `Table` osztályból származik, amely az alap lekérdezésépítési funkciókat biztosítja.  
- Használja a kapcsolódó adatbázistáblákat, mint például: `borrowings`, `books`, `authors`, `categories`, és `publishers`.

## ReservationTable.php

Ez a fájl a `ReservationTable` osztályt definiálja, amely a `Table` osztályból származik,  
és kifejezetten a foglalásokkal kapcsolatos adatok kezelésére és lekérdezésére biztosít metódusokat.

Funkciók:  
- `reserve()`: Új foglalás létrehozása egy adott könyvre (ISBN alapján) és felhasználóra.  
- `deleteReservation()`: Egy adott könyvre (ISBN alapján) vonatkozó foglalás törlése.  
- `bookInReservation()`: Annak ellenőrzése, hogy egy adott könyv (ISBN alapján) jelenleg le van-e foglalva.  
- `countUserReservations()`: Egy adott felhasználó által létrehozott foglalások számának lekérdezése.  
- `selectByISBN()`: Foglalás részleteinek lekérdezése egy adott könyvre (ISBN alapján).  
- `selectMyReservations()`: Egy adott felhasználó által létrehozott foglalások listázása, a foglalás részleteivel együtt.

Használat:  
- Ezekkel a metódusokkal lehet kapcsolatba lépni a `reservations` táblával és a kapcsolódó táblákkal (pl. `books`, `authors`, `categories`).  
- A `selectMyReservations()` metódus támogatja az összetett lekérdezéseket join-okkal, csoportosítással és szűréssel.

Függőségek:  
- A `Table` osztályból származik, amely az alap lekérdezésépítési funkciókat biztosítja.  
- Használja a kapcsolódó adatbázistáblákat, mint például: `reservations`, `books`, `authors`, `categories`, és `publishers`.

## Table.php

Ez a fájl a `Table` osztályt definiálja, amely alapot biztosít az adatbázis-lekérdezések felépítéséhez.  
Olyan metódusokat tartalmaz, amelyek lehetővé teszik SQL lekérdezések (pl. SELECT, INSERT, UPDATE, DELETE, JOIN)  
dinamikus összeállítását és végrehajtását.

Funkciók:  
- Lekérdezésépítés:  
- `select()`: SELECT lekérdezés összeállítása megadott mezőkkel és táblákkal.  
- `insert()`: INSERT lekérdezés összeállítása megadott mezőkkel és értékekkel.  
- `update()`: UPDATE lekérdezés összeállítása megadott mezőkkel és értékekkel.  
- `delete()`: DELETE lekérdezés összeállítása a megadott táblára.  
- Lekérdezésmódosítók:  
- `where()`: WHERE feltételek hozzáadása operátorokkal, mint pl. `=`, `LIKE`, `IN`, `IS NULL`.  
- `groupBy()`: GROUP BY feltételek hozzáadása a lekérdezéshez.  
- `orderBy()`: ORDER BY feltételek hozzáadása növekvő vagy csökkenő sorrendben.  
- `limit()`: LIMIT feltétel megadása az eredmények számának korlátozására.  
- `innerJoin()` és `leftJoin()`: JOIN feltételek hozzáadása az adatok több táblából történő összekapcsolásához.  
- Végrehajtás:  
- `execute()`: A felépített lekérdezés végrehajtása, opcionálisan az eredmények lekérésével.  
- `toString()`: A lekérdezés szöveges formátumba alakítása hibakereséshez vagy al-lekérdezésekhez.  
- Egyéb lehetőségek:  
- Automatikus paraméter kötés előkészített utasításokhoz (prepared statements).  
- A lekérdezési állapot automatikus visszaállítása végrehajtás után, hogy az osztály újra felhasználható legyen.

Használat:  
- Származtass ebből az osztályból konkrét táblákhoz tartozó lekérdező osztályokat (pl. `BooksTable`, `BorrowingsTable`).  
- Használd a rendelkezésre álló metódusokat SQL lekérdezések dinamikus felépítésére és futtatására.

Függőségek:  
- `Database\Connection`: Az adatbázis-kapcsolat létrehozására szolgál a lekérdezések végrehajtásához.

## UserTable.php

Ez a fájl a `UserTable` osztályt definiálja, amely a `Table` osztályból származik,  
és kifejezetten a felhasználókkal kapcsolatos adatok kezelésére és lekérdezésére biztosít metódusokat.

Funkciók:  
- `selectByUsername()`: Felhasználó adatainak lekérdezése felhasználónév alapján.  
- `rowExists()`: Annak ellenőrzése, hogy létezik-e adott feltételeknek megfelelő sor a `users` táblában.  
- `updateToVerified()`: Felhasználó email megerősítési állapotának frissítése.  
- `insertToUser()`: Új felhasználó beszúrása a `users` táblába.  
- `numberOfUsers()`: Felhasználók számának lekérdezése az adatbázisból.  
- `selectUserData()`: Részletes felhasználói adatok lekérdezése, beleértve a szerepkör-információkat is.  
- `updateUserData()`: Egy adott felhasználó adatainak frissítése.  
- `deleteUserRow()`: Felhasználó törlése a `users` táblából.  
- `selectByEmail()`: Felhasználói adatok lekérdezése email-cím alapján.  
- `changePassword()`: Felhasználó jelszavának frissítése.  
- `allByID()`: Összes felhasználói adat lekérdezése felhasználóazonosító alapján.

Használat:  
- Ezekkel a metódusokkal lehet kapcsolatba lépni a `users` táblával, például hitelesítés, regisztráció,  
profilfrissítés és fiókkezelés céljából.  
- A `selectUserData()` és `updateUserData()` metódusok támogatják az összetett lekérdezéseket join-okkal és frissítésekkel.

Függőségek:  
- A `Table` osztályból származik, amely az alap lekérdezésépítési funkciókat biztosítja.  
- A `Helper` osztályt használja a bemeneti adatok validálására.

## BooksController.php

Ez a fájl a `BooksController` osztályt definiálja, amely a könyvekkel kapcsolatos HTTP-kéréseket kezeli.  
Kapcsolatot teremt a kliens és a `BooksTable` adatbázis-lekérdezések között, elvégzi a bemeneti adatok validálását,  
és megfelelő válaszokat ad vissza.

Funkciók:  
- `getFromDBByParams()`: Könyvek lekérdezése az adatbázisból megadott paraméterek alapján.  
- `countAllBooks()`: A könyvek teljes számának visszaadása az adatbázisból.  
- `availableBooks()`: Az éppen elérhető (nem kölcsönzött vagy lefoglalt) könyvek listájának lekérdezése.

Használat:  
- Ezt a kontrollert használd könyvekkel kapcsolatos műveletekhez, például kereséshez, számláláshoz  
vagy elérhetőség ellenőrzéséhez.  
- A bemeneti adatokat a `Helper` és a `Books` validáló osztályok segítségével ellenőrzi, mielőtt lekérdezést végezne.

Függőségek:  
- `ApiResponse\Response`: HTTP válaszok küldésére szolgál.  
- `Database\Queries\BooksTable`: A könyvekhez kapcsolódó adatbázis-lekérdezési metódusokat biztosítja.  
- `Helper\Helper`: A bemeneti adatok validálására szolgál.  
- `App\Validations\Books`: Könyvekkel kapcsolatos speciális validálási metódusokat tartalmaz.

## UserController.php

Ez a fájl a `UserController` osztályt definiálja, amely a felhasználókezeléssel kapcsolatos HTTP-kéréseket kezeli.  
Kapcsolatot teremt a kliens és a `UserTable` adatbázis-lekérdezések között, elvégzi a bemeneti adatok validálását,  
hitelesítést, és megfelelő válaszokat ad vissza.

Funkciók:  
- `login()`: Felhasználó hitelesítése felhasználónév és jelszó alapján, majd token visszaadása.  
- `verifyUser()`: Felhasználói token ellenőrzése.  
- `register()`: Új felhasználó regisztrálása, adatok validálása és megerősítő e-mail küldése.  
- `verifyAccount()`: Felhasználói fiók megerősítése egy megerősítő kód segítségével.  
- `allUsers()`: A felhasználók számának lekérdezése az adatbázisból.  
- `userData()`: Felhasználói adatok lekérdezése azonosító alapján.  
- `updateUser()`: Felhasználói adatok frissítése azonosító alapján validálás után.  
- `deleteUser()`: Felhasználó törlése azonosító alapján.  
- `forgotPassword()`: Jelszó-visszaállító e-mail küldése a felhasználónak.  
- `changePassword()`: Jelszó megváltoztatása validálás után.  
- `finalizeRegistration()`: Regisztráció véglegesítése e-mail frissítésével és megerősítő e-mail küldésével.

Használat:  
- Ezt a kontrollert használd felhasználókkal kapcsolatos műveletekhez, mint például bejelentkezés,  
regisztráció, profilfrissítés vagy jelszókezelés.  
- A bemeneti adatokat a `Helper` és `User` validáló osztályok ellenőrzik, mielőtt adatbázis-lekérdezés történik.

Függőségek:  
- `ApiResponse\Response`: HTTP válaszok küldésére szolgál.  
- `Database\Queries\UserTable`: A felhasználókkal kapcsolatos adatbázis-lekérdezési metódusokat biztosítja.  
- `Helper\Helper`: Bemeneti adatok validálására szolgál.  
- `App\Validations\User`: Felhasználókkal kapcsolatos speciális validálási metódusokat tartalmaz.  
- `App\Authorize\Token`: Tokenek generálásáért és ellenőrzéséért felelős.  
- `Emailer\SendEmail` és `Emailer\EmailBodies`: E-mailek küldésére és tartalomgenerálásra szolgál.

## Response.php

Ez a fájl a `Response` osztályt definiálja, amely hasznos metódusokat biztosít HTTP válaszok küldésére  
JSON formátumban. Az osztály célja az API végpontok hibás és sikeres válaszainak egységes kezelése.

Funkciók:  
- `httpError($code, $message)`: HTTP hibaválasz küldése megadott státuszkóddal és hibaüzenettel.  
- Ha az üzenet numerikus, akkor a megfelelő hibaüzenetet az `errorCodes.txt` fájlból szerzi be.  
- `httpSuccess($code, $json)`: HTTP sikeres válasz küldése megadott státuszkóddal és JSON adatokkal.

Használat:  
- Használjuk a `httpError()` metódust hibás válaszok küldésére a megfelelő HTTP státuszkóddal és hibaüzenetekkel.  
- Használjuk a `httpSuccess()` metódust sikeres válaszok küldésére JSON formátumban.

Függőségek:  
- Az `errorCodes.txt` fájlra támaszkodik, amely a numerikus hiba kódokat a hozzájuk tartozó hibaüzenetekre térképezi.

Megjegyzések:  
- Mindkét metódus automatikusan beállítja a `Content-Type` fejlécet `application/json` értékre,  
és a válasz küldése után lezárja a szkriptet.

## User.php

Ez a fájl a `User` osztályt definiálja, amely validálási metódusokat biztosít a felhasználókkal kapcsolatos adatok számára.  
Biztosítja a felhasználói adatok integritását és helyességét, például jelszavak, felhasználónevek, e-mailek és egyéb felhasználói attribútumok esetén.  
Ezen kívül kezel autentikációs és megerősítési logikát is.

Funkciók:  
- `checkPassword()`: Validálja és hasheli a jelszót, biztosítva, hogy megfeleljen a biztonsági követelményeknek.  
- `checkUsername()`: Validálja a felhasználónevet, és ellenőrzi annak egyediségét az adatbázisban.  
- `checkEmail()`: Validálja az e-mail címet, és ellenőrzi annak egyediségét az adatbázisban.  
- `checkFirstLastName()`: Validálja a kereszt- és családi neveket a megfelelő formázás érdekében.  
- `loginAuth()`: Hitelesíti a felhasználót a felhasználónév és jelszó ellenőrzésével.  
- `createAuthCode()`: Generál egy 6 számjegyű hitelesítő kódot.  
- `checkAuthCode()`: Validálja a hitelesítő kódot az e-mail megerősítése érdekében.  
- `updateToVerified()`: A felhasználót megerősítettre jelöli a sikeres e-mail megerősítés után.  
- `userPasswordIsMatch()`: Ellenőrzi, hogy a megadott jelszó megegyezik-e az adatbázisban tárolt hash-elt jelszóval.  
- `validateAddress()`: Validálja a felhasználói címet a megfelelő formázás és hosszúság érdekében.  
- `validateEmail()`: Validálja az e-mail cím formátumát.  
- `validateDateOfBirth()`: Validálja a felhasználó születési dátumát, hogy megfeleljen az életkori követelményeknek.  
- `isUserVerified()`: Ellenőrzi, hogy a felhasználó megerősítve van-e az e-mail cím vagy a fiók státusza alapján.

Használat:  
- Használjuk ezeket a metódusokat a felhasználói adatok validálására regisztráció, bejelentkezés és profilfrissítés során.  
- Az olyan metódusok, mint a `loginAuth()` és `checkAuthCode()`, alapvetőek az autentikációs és megerősítési folyamatokban.

Függőségek:  
- `ApiResponse\Response`: HTTP hibás vagy sikeres válaszok küldésére szolgál.  
- `Helper\Helper`: Bemeneti adatok validálására szolgáló segédfunkciók biztosítása.  
- `Database\Queries\UserTable`: Adatbázis-interakciók a felhasználói adatok ellenőrzésére és frissítések végrehajtására.

## Model.php

Ez a fájl a `Model` osztályt definiálja, amely segédfunkciókat biztosít az adatok validálásához és feldolgozásához.  
Az osztály alapértelmezettként szolgál az input validálásához, típuskezeléshez és egyéb gyakori műveletekhez,  
amelyek szükségesek a validálási munkafolyamatokhoz.

Funkciók:  
- `checkRequiredData()`: Validálja és kinyeri a szükséges és opcionális adatokat egy bemeneti tömbből.  
- `makeTypesArray()`: SQL lekérdezésekhez szükséges adattípusok tömbjét generálja a bemeneti mezők és operátorok alapján.  
- `callingValidateFunctions()`: Dinamikusan hívja meg a validálási funkciókat a megadott mezők számára.  
- `validateISBN()`: Validálja az ISBN formátumot (10 vagy 13 számjegy).  
- `validateID()`: Validálja egy azonosítót, hogy csak numerikus karaktereket tartalmazzon.  
- `removeNullValues()`: Eltávolítja a null vagy üres értékeket egy bemeneti tömbből.

Használat:  
- Használja ezt az osztályt alapként a validálási logikában más osztályok számára.  
- Az olyan metódusok, mint a `checkRequiredData()` és `callingValidateFunctions()`, hasznosak az input adatok strukturált feldolgozásához és validálásához.

Függőségek:  
- `ApiResponse\Response`: HTTP hibás válaszok küldésére szolgál, amikor a validálás nem sikerül.

## Image.php

Ez a fájl az `Image` osztályt definiálja, amely validálási funkciókat biztosít a képfájlok kezelésére.  
Biztosítja, hogy a feltöltött képek megfeleljenek bizonyos követelményeknek, mint például fájl típus, méret és létezés.

Funkciók:  
- `validateImage($img)`: Validálja a feltöltött képfájlt a létezés, típus és méret szempontjából.  
- `validateExtentions($fileType)`: Validálja a fájl típusát, és biztosítja, hogy az engedélyezett képformátumok egyike legyen (pl. PNG, JPG, JPEG).  
- `checkForValidFile($imgPath)`: Ellenőrzi, hogy egy érvényes képfájl létezik-e a megadott elérési úton, és a helyes MIME típusú fájlt szolgáltatja.

Használat:  
- Használja a `validateImage()` metódust a feltöltött képek validálására a fájl feltöltési folyamatok során.  
- Használja a `checkForValidFile()` metódust képfájlok kiszolgálására a szerverről, ha léteznek.

Függőségek:  
- `ApiResponse\Response`: HTTP hibás válaszok küldésére szolgál, amikor a validálás nem sikerül.

Megjegyzések:  
- A maximálisan engedélyezett kép mérete 70 MB.  
- Támogatott képformátumok: PNG, JPG, és JPEG.

## Books.php

Ez a fájl a `Books` osztályt definiálja, amely validálási funkciókat biztosít a könyvekkel kapcsolatos adatok kezelésére.  
Biztosítja a bemenetek integritását és helyességét, például könyvcímek, kiadási évek,  
kiadók és szerzők nevének formátumát.

Funkciók:  
- `validateFullName($fullName)`: Validálja egy szerző teljes nevének formátumát.  
- `validateTitle($title)`: Validálja egy könyv címének formátumát.  
- `validateYear($year)`: Validálja egy könyv kiadási évét, biztosítva, hogy az ne legyen a jövőben.  
- `validatePublisher($publisher)`: Validálja egy kiadó nevének formátumát.

Használat:  
- Használja ezeket a metódusokat könyvekkel kapcsolatos bemenetek validálására adatrögzítés vagy frissítés során.  
- Ezek a validálások biztosítják, hogy az adatok az elvárt formátumoknak és korlátozásoknak megfelelőek.

Függőségek:  
- `ApiResponse\Response`: HTTP hibás válaszok küldésére szolgál, amikor a validálás nem sikerül.

## Borrowing.php

Ez a fájl a `Borrowing` osztályt definiálja, amely validálási funkciókat biztosít a kölcsönzéssel kapcsolatos adatok kezelésére.  
Biztosítja a bemenetek helyességét, például a kölcsönzés dátumainak és a kölcsönzési limiteknek.

Funkciók:  
- `checkFormatDate($date)`: Validálja egy dátum formátumát (YYYY-MM-DD), és biztosítja, hogy érvényes naptári dátum legyen.  
- `checkLimit($limit)`: Validálja, hogy a kölcsönzési limit numerikus érték-e.

Használat:  
- Használja ezeket a metódusokat kölcsönzéssel kapcsolatos bemenetek validálására adatrögzítés vagy frissítés során.  
- Ezek a validálások biztosítják, hogy az adatok az elvárt formátumoknak és korlátozásoknak megfelelőek.

Függőségek:  
- `ApiResponse\Response`: HTTP hibás válaszok küldésére szolgál, amikor a validálás nem sikerül.

## ReservationController.php

Ez a fájl a `ReservationController` osztályt definiálja, amely a könyv foglalásokkal kapcsolatos HTTP kéréseket kezeli.  
Közvetítőként működik a kliens és a `ReservationTable` adatbázis lekérdezések között, elvégezve a bemenetek validálását  
és a megfelelő válaszokat visszaadva.

Funkciók:  
- `store($body, $userID)`: Új foglalást hoz létre egy könyv számára egy adott felhasználó által a bemenet validálása után.  
- `destroy($ISBN, $userID)`: Törli egy adott könyv foglalását egy adott felhasználó számára.  
- `show($body, $userID)`: Lekérdezi az összes foglalást, amit egy adott felhasználó tett.

Használat:  
- Használja ezt az kontrollert foglalásokkal kapcsolatos műveletekhez, mint például a foglalás létrehozása, törlése és listázása.  
- A bemeneti adatokat a `Helper` és `Model` validáló osztályok segítségével validálják, mielőtt adatbázis lekérdezés történik.

Függőségek:  
- `ApiResponse\Response`: HTTP válaszok küldésére szolgál.  
- `Database\Queries\ReservationTable`: Adatbázis lekérdezési módszerek biztosítása a foglalásokhoz.  
- `Database\Queries\BooksTable`: A könyv létezésének validálásához használatos.  
- `Database\Queries\BorrowingsTable`: A kölcsönzés állapotának ellenőrzésére szolgál.  
- `Helper\Helper`: A bemenetek validálásához használatos.  
- `App\Validations\Model`: További validálási módszereket biztosít.

## ImageController.php

Ez a fájl az `ImageController` osztályt definiálja, amely a képfeltöltésekkel és lekérésekkel kapcsolatos HTTP kéréseket kezeli.  
Validálja a képfájlokat, kezeli a feltöltéseket, és kiszolgálja a képeket a szerverről.

Funkciók:  
- `uploadImg()`: Kezeli egy kép fájl feltöltését, validálja azt, és áthelyezi a kijelölt könyvtárba.  
- `getImg($img)`: Lekéri egy kép fájlt az azonosítója alapján, validálja, és kiszolgálja a kliensnek. Ha a kép hiányzik, egy alapértelmezett képet szolgáltat.

Használat:  
- Használja az `uploadImg()`-t a képfeltöltések kezelésére a kliensektől.  
- Használja a `getImg()`-t képek lekérésére és kiszolgálására a klienseknek.

Függőségek:  
- `App\Validations\Image`: A képfájlok validálására szolgál.  
- `ApiResponse\Response`: HTTP válaszok küldésére használatos, ha a validálás vagy a fájl műveletek hibát okoznak.

Megjegyzések:  
- A feltöltött képek a `booksImages` könyvtárban kerülnek tárolásra.  
- Ha egy kép nem található, egy alapértelmezett "hiányzó" képet szolgáltat.

## BorrowingsController.php

Ez a fájl a `BorrowingsController` osztályt definiálja, amely a könyvek kölcsönzéseivel kapcsolatos HTTP kéréseket kezeli.  
A kliens és a `BorrowingsTable` adatbázis lekérdezések között hidat képez, végrehajtja a bemeneti validálást,  
és megfelelő válaszokat küld vissza.

Funkciók:  
- `allBorrowings($body)`: Lekéri az adatbázisból a kölcsönzések teljes számát.  
- `show($body, $userID)`: Lekéri egy adott felhasználó által kölcsönzött könyvek listáját.  
- `topBorrowedBooks($limit)`: Lekéri a leggyakrabban kölcsönzött könyveket, a megadott számú limit alapján.

Használat:  
- Használja ezt a kontrollert kölcsönzéssel kapcsolatos műveletekhez, mint például a kölcsönzési statisztikák lekérése vagy felhasználó-specifikus kölcsönzések.  
- A bemeneti adatokat a `Borrowing` validálási osztály segítségével validálják, mielőtt lekérdezéseket hajtanának végre az adatbázisban.

Függőségek:  
- `ApiResponse\Response`: A HTTP válaszok küldésére használatos.  
- `Database\Queries\BorrowingsTable`: Kölcsönzésekkel kapcsolatos adatbázis lekérdezési módszereket biztosít.  
- `App\Validations\Borrowing`: A kölcsönzéssel kapcsolatos bemeneti adatok validálásához használt osztály.

## Token.php

Ez a fájl a `Token` osztályt definiálja, amely a JSON Web Tokenek (JWT) létrehozására és érvényesítésére szolgáló módszereket biztosít.  
A felhasználói hitelesítéshez és autorizációhoz használatos az alkalmazásban.

Funkciók:  
- `makeToken($userid, $username, $expireTime = 3600)`: Generál egy JWT-t egy felhasználó számára a megadott lejárati idővel.  
- `verifyToken($token)`: Érvényesíti a JWT-t és kinyeri a felhasználói azonosítót belőle.

Használat:  
- Használja a `makeToken()` metódust az autentikált felhasználók számára történő tokenek generálásához.  
- Használja a `verifyToken()` metódust a tokenek validálásához és a felhasználói információk lekéréséhez védett útvonalakhoz.

Függőségek:  
- `Firebase\JWT\JWT` és `Firebase\JWT\Key`: A JWT-k kódolására és dekódolására használt osztályok.  
- `Config\Env`: Betölti a környezeti változókat, beleértve a JWT titkos kulcsot.  
- `ApiResponse\Response`: HTTP hibaválaszok küldése, ha a token érvényesítése nem sikerül.

Megjegyzések:  
- A tokeneket az HS256 algoritmus és egy titkos kulcs segítségével írják alá, amely a környezeti változókban tárolódik.  
- Az `iss` állítás az alkalmazás alap URL-jére van állítva, és az `exp` állítás meghatározza a token lejárati idejét.

## router.php

Ez a fájl egy egyszerű routerként szolgál, amely kezeli a beérkező HTTP kéréseket, és továbbítja azokat a megfelelő vezérlőkhöz.  
Meghatározza az alkalmazás API útvonalait, és összeköti azokat a megfelelő vezérlőmetódusokkal.

Funkciók:  
- Kezeli az API végpontok útvonalait, beleértve a felhasználókezelést, könyvkezelést, kölcsönzéseket, foglalásokat és képek kezelését.  
- Támogatja a többféle HTTP metódust (GET, POST, PUT, DELETE).  
- Központosított helyet biztosít az API útvonalak definiálására és kezelésére.  
- 404-es hibát ad vissza, ha a kért útvonal nem található.

Útvonalak:  
- `/api/login`: Felhasználói bejelentkezés kezelése.  
- `/api/register`: Felhasználói regisztráció kezelése.  
- `/api/verify-account`: Fiók megerősítésének kezelése.  
- `/api/forgot-password`: Jelszó-visszaállítási kérések kezelése.  
- `/api/user`: Felhasználói adatok lekérése és frissítése.  
- `/api/finalize-registration`: Regisztráció véglegesítése.  
- `/api/change-password`: Jelszó módosításának kezelése.  
- `/api/delete-user`: Felhasználó törlése.  
- `/api/top-borrowings/{limit}`: Leggyakrabban kölcsönzött könyvek lekérése.  
- `/api/borrowings`: Egy adott felhasználó kölcsönzéseinek lekérése.  
- `/api/available-books`: Elérhető könyvek lekérése.  
- `/api/upload-img`: Képfeltöltés kezelése.  
- `/img/{ISBN}`: Könyv borítóképének lekérése.  
- `/api/books`: Könyvek lekérése paraméterek alapján.  
- `/api/all-books`: Az összes könyv számának lekérése.  
- `/api/all-users`: Az összes felhasználó számának lekérése.  
- `/api/all-borrowings`: Az összes kölcsönzés számának lekérése.  
- `/api/reserve`: Könyvfoglalás kezelése.  
- `/api/test`: Teszt útvonal a kölcsönzési elérhetőség ellenőrzésére.

Függőségek:  
- `Router\Router`: Az útvonalak és kérések kezelése.  
- `ApiResponse\Response`: API válaszok és hibaformázás kezelése.  
- Vezérlők (Controllers): Az egyes funkciók kezelése (pl. `UserController`, `BooksController` stb.).