Vizsgaremek

Scriptum online könyvesbolt

Görtler Zsolt

Szaniszló Bálint

Vizsgaremek adatlap

A Vizsgaremek készítői:

Neve: Görtler Zsolt

E-mail címe: zsoltgortler@gmail.com

Neve: Szaniszló Balint

E-mail címe: [k955658@gmail.com](mailto:k955658@gmail.com)

A Vizsgaremek témája:

A projekt célja egy online könyvesbolt létrehozása, amely modern webes technológiák felhasználásával biztosítja a könyvek böngészését, értékelését és megvásárlását. A rendszer lehetőséget nyújt a felhasználóknak a teljes vásárlási folyamat lebonyolítására egy biztonságos, felhasználóbarát felületen keresztül.

A Vizsgaremek címe:

A „Scriptum online könyvesbolt” egy Laravel keretrendszerre épülő webalkalmazás, amely Vue.js frontenddel párosulva biztosít egy dinamikus, gyors és reszponzív felhasználói élményt. A rendszer lehetővé teszi könyvek keresését, kategorizálását, értékelését és megrendelését egy egyszerű és intuitív kezelőfelületen keresztül.

Kelt: Budapest, 2025.március.23.

A képen kézírás, kalligráfia, Betűtípus, kézzel írott látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen kézírás, kalligráfia, Betűtípus, kézzel írott látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

..................................................... ............................................. Görtler Zsolt Szaniszló Bálint

Eredetiség Nyilatkozat

Alulírott tanuló kijelentem, hogy a vizsgaremek saját és csapattársa(i)m munkájának eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült vizsgaremekrészét képező anyagokat az intézmény archiválhatja és felhasználhatja.

A képen kézírás, kalligráfia, tinta, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Kelt: Budapest, 2025.március.23.

.......................................

Görtler Zsolt

A képen kézírás, kalligráfia, Betűtípus, kézzel írott látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

.....................................

Szaniszló Bálint

Tartalom

[Tartalmi rész - 5 -](#_Toc196837272)

[1 Projekt ismertetése: - 5 -](#_Toc196837273)

[2 Felhasználói dokumentáció - 6 -](#_Toc196837274)

[2.2 Technikai követelmények: - 6 -](#_Toc196837275)

[2.2.1 Ajánlott böngészők: - 6 -](#_Toc196837276)

[2.2.2 Hibaelhárítás: - 6 -](#_Toc196837277)

[2.3 Weboldal bemutatása felhasználóként - 7 -](#_Toc196837278)

[2.4 Keresősáv bemutatása - 8 -](#_Toc196837279)

[2.5 Könyv részleteinek bemutatása - 9 -](#_Toc196837280)

[2.6 Kosár bemutatása - 10 -](#_Toc196837281)

[2.7 Kategóriák bemutatása - 11 -](#_Toc196837282)

[2.8 Regisztrációs felület bemutatása - 12 -](#_Toc196837283)

[2.9 Bejelentkezési felület és annak működésének bemutatása - 15 -](#_Toc196837284)

[2.10 Új jelszó igénylésének bemutatása - 16 -](#_Toc196837285)

[2.11 A felhasználói adatok szerkesztése - 17 -](#_Toc196837286)

[2.13 Vásárlás folyamatának bemutatása - 19 -](#_Toc196837287)

[2.13.1 Szállítási adatok megadása - 20 -](#_Toc196837288)

[2.13.2 Rendelés áttekintése - 21 -](#_Toc196837289)

[2.13.3 Rendelés véglegesítése - 22 -](#_Toc196837290)

[2.13.4 A rendelés véglegesítéséről kapott email - 22 -](#_Toc196837291)

[2.14 Rendeléseim menüpont bemutatása - 23 -](#_Toc196837292)

[2.15 Weboldal bemutatása Adminként - 24 -](#_Toc196837293)

[2.15.1 Admin Főoldal - 24 -](#_Toc196837294)

[3 Fejlesztői dokumentáció - 27 -](#_Toc196837295)

[3.1 Munkamegosztás: - 27 -](#_Toc196837296)

[3.2 Telepítési dokumentáció - 27 -](#_Toc196837297)

[3.3 Adatbázis - 28 -](#_Toc196837298)

[3.3.3 Mezők részletes ismertetése a táblák szerint - 29 -](#_Toc196837299)

[3.3.4 Kapcsolatok indoklása, ismertetése - 29 -](#_Toc196837300)

[3.4 Backend komponens/struktúra dokumentáció - 31 -](#_Toc196837301)

[3.4.1 Migráció - 31 -](#_Toc196837302)

[3.4.2 Rendszertáblák alapszintű bemutatása - 31 -](#_Toc196837303)

[3.4.8 Seederek - 36 -](#_Toc196837304)

[3.4.9 Seederek működése - 41 -](#_Toc196837305)

[3.4.11 Modell - 44 -](#_Toc196837306)

[3.4.12 Controllerek: - 47 -](#_Toc196837307)

[3.4.13 További resource, requestek, illetve controllerek: - 54 -](#_Toc196837308)

[3.4.14 Authentikáció - 55 -](#_Toc196837309)

[3.4.15 Router védelem - 57 -](#_Toc196837310)

[3.4.16 Authorization / Jogosultságok - 58 -](#_Toc196837311)

[3.4.17 Email - 59 -](#_Toc196837312)

[3.5 Komponens/struktúra dokumentáció (Frontend) - 68 -](#_Toc196837313)

[3.5.6 Router kapcsolatok - 82 -](#_Toc196837314)

[3.5.7 Router Konfiguráció - 83 -](#_Toc196837315)

[3.5.7 Üzleti logika dokumentáció - 95 -](#_Toc196837316)

[3.6 API Dokumentáció - 100 -](#_Toc196837317)

[3.6.1 Végpontok - 100 -](#_Toc196837318)

[3.7 Teszt dokumentáció - 120 -](#_Toc196837319)

[3.7.8 További API tesztek - 126 -](#_Toc196837320)

[4. További fejlesztési lehetőségek - 131 -](#_Toc196837321)

[5. Összefoglalás - 132 -](#_Toc196837322)

[6. Források - 133 -](#_Toc196837323)

# Tartalmi rész

# 1 Projekt ismertetése:

Ez a projekt egy modern, teljes értékű webalkalmazás, amely Laravel backend és Vue.js frontend technológiákat ötvöz az optimális felhasználói élmény és fejlesztői hatékonyság érdekében. A rendszer egy kétoldalú architektúrán alapul, ahol a backend RESTful API-n keresztül kommunikál a frontend réteggel, biztosítva a tiszta szétválasztást és a jobb karbantarthatóságot. A Laravel keretrendszer biztosítja a robusztus adatbázis-kezelést, biztonságos autentikációt és komplex üzleti logikát, míg a Vue.js a reaktív, dinamikus felhasználói felületet szolgáltatja. A projekt előnyben részesíti a modern fejlesztési gyakorlatokat, beleértve a komponens-alapú architektúrát, a dependency injection mintákat és az átfogó tesztelési stratégiákat. Az alkalmazás használja a Bootstrap keretrendszert a reszponzív design megvalósításához, valamint a FontAwesome ikonokat a vizuális elemek gazdagításához. A fejlesztési folyamat során a kód verziókezelése Git segítségével történik, ami lehetővé teszi a hatékony csapatmunkát és a változtatások nyomon követését. A rendszer skálázhatóságát a jól megtervezett adatbázis-struktúra és a hatékony lekérdezések biztosítják, míg a felhasználói élményt a Vue Router és Vuex integrációja teszi gördülékennyé. A projekt célja egy könnyen karbantartható, bővíthető és modern webalkalmazás létrehozása, amely mind a fejlesztők, mind a végfelhasználók számára kielégítő élményt nyújt.

# 2 Felhasználói dokumentáció

## 2.2 Technikai követelmények:

### 2.2.1 Ajánlott böngészők:

- Microsoft Edge

- Brave

- Google Chrome

Ismert kompatibilitási problémák:

A Firefoxban nem érdemes elindítani az oldalt, mert el van csúszva a kosár oldal kinézete, a továbbiakban kijavítás alatt lesz

Tesztelésre való belépési adatok:

- Email: vizsgaremek13@outlook.hu

- Jelszó: vizsga123

Tesztelői email belépés (outlook)

- Email: vizsgaremek13@outlook.hu

- Jelszó: vizsga123 vagy Vizsga123

### 2.2.2 Hibaelhárítás:

- Fejlesztés alatt áll a regisztrációkor elküldött email gyorsaságának javítása. Kérjük várjon pár másodpercet, míg az oldal visszajelzést nem ad arra, hogy elküldte az email megerősítést önnek.

A főoldal főrészében a következő tematikákban jelennek meg a könyvek:

⦁ Legnézettebb könyvek (legtöbbet látogatott könyv/könyvek)

⦁ Legkedveltebb könyvek (legjobb értékeléssel bíró könyv/könyvek)

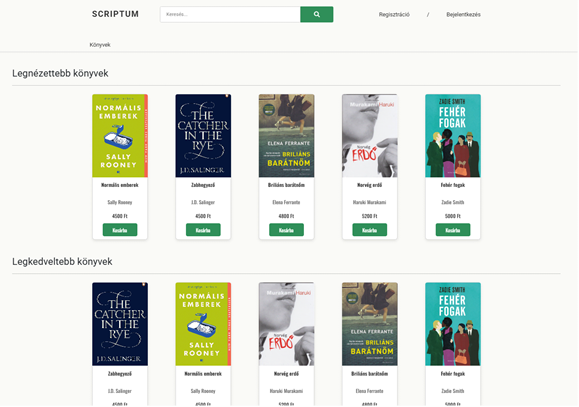
⦁ Legtöbet kommentelt könyvek (legtöbb megjegyzéssel rendelkező könyv/könyvek)

⦁ Legújabb könyvek

⦁ Legrégebben írt könyvek

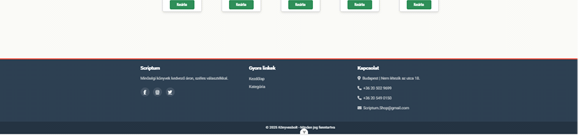
⦁ További könyvek

## 2.3 Weboldal bemutatása felhasználóként



1. ábra - Főoldal bemutatása

Amikor megnyitja az oldalt, a főoldalon találja magát a felhasználós. Az 1-es ábra a főoldalt mutatja be. Az oldal fejlécben van egy navigációs rész, amelyben helyezkedik el a kereső sáv, ahol könyveket lehet kikeresni a könyv címe, illetve a könyv írója alapján. A következő a Regisztrációs/ Bejelentkezés router link, amely két különböző alkomponensre küld át. A nevükből kiindulva a Regisztrációs részlegen lehet a még nem létező felhasználót létrehozni, a bejelentkezésbe pedig, a regisztrált fiókkal lehet bejelentkezni, annak érdekében, hogy teljes mértékben kitudja élvezni a Scriptum online könyvesoldal funkcionalitásait. A Könyvek lenyíló listában lehet a könyveket kategóriájuk alapján kikeresni.



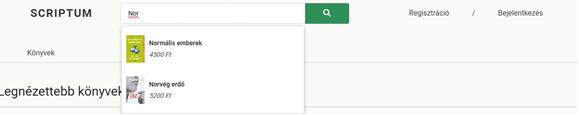
2. ábra - Lábjegyzék bemutatása

A lap legalján megtalálható lábjegyzékben található a weboldal emailcíme ahogyan a 2. ábrán is látható, social media platformok, mint pl. Facebook, Instagramm, Twitter vagy mai nevén X, gyors linkek (Kezdőlap, Kategória), illetve az elérési adatok (Cím, 2 tulajdonos telefonszám (Szaniszló Bálint, Görtler Zsolt).

## 2.4 Keresősáv bemutatása

A keresősávba minimum 3 karakter írása után megtalálja azokat a könyveket, amiknek a címében vagy a szerző nevében szerepel az a 3 karakter, amit begépelt.

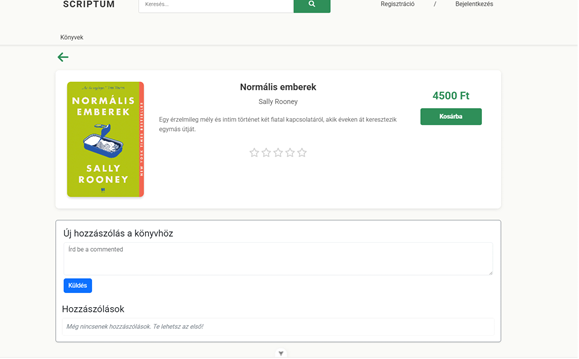
Amennyiben több azonos nevezetű vagy több azonos kezdőbetűkkel rendelkező könyv van a keresésben, úgy a keresés gombbal is lehet keresni.



3. ábra - Keresősáv bemutatása

## 2.5 Könyv részleteinek bemutatása

Akár a kezdőlapon levő könyvek rákattintásával, akár a kereső sávban rákeresett könyvekre kattintva megtekintheti részletesen egy könyv részleteit, mint pl. Könyv leírása, commenteket, értékelést és itt is elérhető a kosár funkció!



4. ábra - Egy Könyv részleteinek megtekintése

## 2.6 Kosár bemutatása

A kosárba rakott elemek egymás alatt jelennek meg, amelybe láthatja táblázat szerűen a könyvek adatait. Lehetséges egyesével hozzáadni, illetve levonni a könyvek mennyiségéből, a felhasználó kedve szerint, illetve lehet a sort, vagyis az adott könyvet/könyveket törölni a piros kuka ikon segítségével.

A fizetést akkor tudjuk megtenni, ha már bevagyunk jelentkezve, amennyiben nem bejelentkezett felhasználó szeretne vásárolni, automatikusan a bejelentkezés felületre küld át. A képen szöveg, képernyőkép, Weblap, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

5. ábra - Kosár működésének bemutatása

## 2.7 Kategóriák bemutatása

A könyveket kategóriák szerint is meglehet tekinteni. Ezt egy lenyíló listában találjuk a fejlécben



6. ábra - Kategória oldal működése

A lenyíló listában a következő kategóriákat találhatjuk

⦁ Modern irodalom

⦁ Klasszikus irodalom

⦁ Disztópia

⦁ Romantikus regény

⦁ Amerikai irodalom

⦁ Kalandregény

⦁ Fantasy

⦁ Történelmi regény

⦁ Pszichológiai regény

⦁ Epikus fantasy

⦁ Horror

⦁ Erotikus



7. ábra - Kategória oldal kinézete

A 7. ábra példán keresztül mutatja be, hogy néz ki egy Kategória kinézete. Jelen esetben ez a Fantasy részleg.

## 2.8 Regisztrációs felület bemutatása

Regisztrációs felületen látható egy űrlap, ahol kitudjuk tölteni az információnkkal, hogy létrehozzunk egy felhasználót.

A következő információk kellenek egy sikeres felhasználó regisztrációjához.

⦁ Felhasználónév

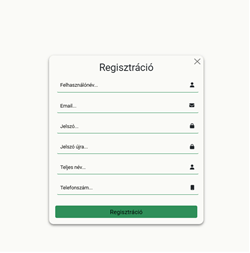
⦁ Email

⦁ Jelszó

⦁ Jelszó megerősítése

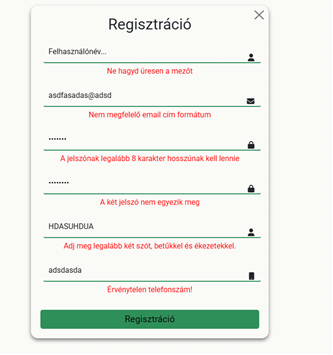
⦁ Teljes név

⦁ Telefonszám



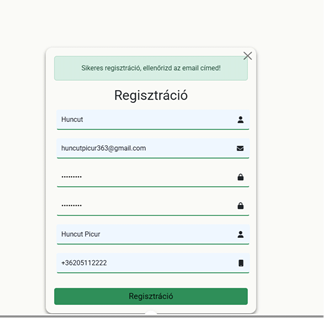
8. ábra - Regisztrációs folyamat kezdetes

Mindegyik beviteli mezője hitelesítve van, így nem tudunk pl. A telefonszámhoz random karaktereket, illetve szöveget írni, illetve megvan szabva a telefonszám hossza. A validáció megakadályozza, hogy érvénytelen információkat adjunk meg bizonyos beviteli mezőkbe.



9. ábra - Regisztráció űrlap validációkezelés bemutatása

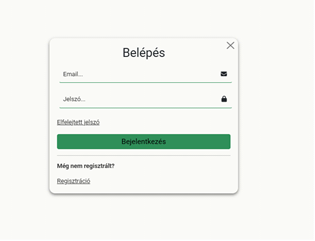
A sikeres regisztrációt követően küldünk egy hitelesítő emailt a sikeresen beregisztrált felhasználónak, amelyet érdemes megnyitni és rákattintani, hisz anélkül nincs meg a teljes regisztráció. FIGYELEM! A gomb lenyomása után a regisztráció során pár másodpercet kell várni, mert lassú válaszidő van az email rendszer és a weboldal között!



10. ábra - Sikeres regisztráció

## 2.9 Bejelentkezési felület és annak működésének bemutatása

Amint megnyitottuk az elküldött emailt és rákattintottunk a megerősítő linkre, már be is jelentkezhetünk a fiókunkba.



11. ábra - Belépés megkezdése

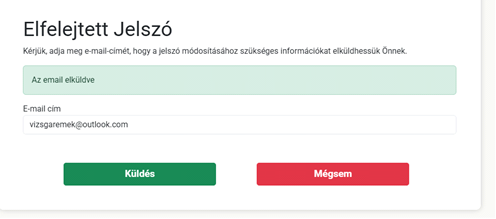
A bejelentkezéskor is hitelesítve vannak a beviteli mezők, így itt sem tudunk random adatokat megadni.



12. ábra - Bejelentkezési felület validációjának bemutatása

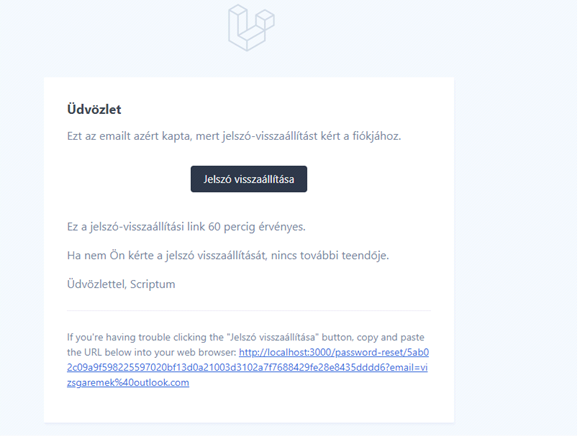
## 2.10 Új jelszó igénylésének bemutatása

Amennyiben elfelejtette a beregisztrált felhasználója jelszavát, erre gondolván, implementálva van egy új jelszó kérési lehetőség, amely az email megadásával tudunk új jelszót igényelni.



13. ábra - Elfelejtett Jelszó elküldése az emailre

Amennyiben megkapta az emailt, ezt kell, hogy lássa a felhasználó.



14. ábra - Elfelejtett Jelszó email tartalma

A gombra rákattintva, visszaküld a bejelentkezési felületre, ahol a következőket kell megadni

⦁ Email cím

⦁ Az új jelszó

⦁ Új jelszó újra



15. ábra - Jelszó visszaállítása

## 2.11 A felhasználói adatok szerkesztése

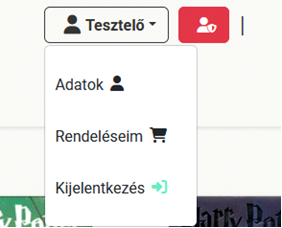
Amikor bejelentkezett a felhasználó, megjelenik egy felhasználói ikon és a neve a keresési sáv mellett. Ez egy lenyíló lista, amelyben három opció közül tud választani.

⦁ Adatok (saját adatait tudja szerkeszteni)

⦁ Rendeléseim (eddigi rendelések megtekintése)

⦁ Kijelentkezési lehetőség

A mellette lévő piros gomb az admin oldal. Ez csak az admin jogokkal rendelkező felhasználóknál jelenik meg.



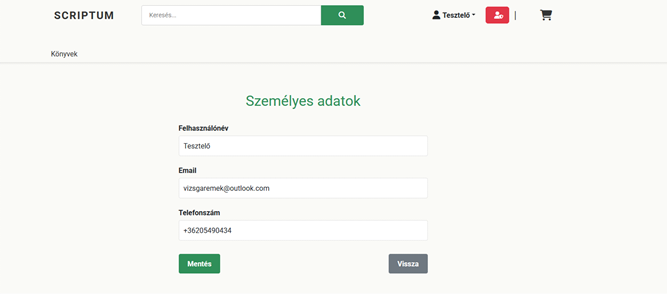
16. ábra - A felhasználói részletek

Az Személyes adatok kezelésénél bármely jogosultságú felhasználó átírhatja a következő adatait:

⦁ Felhasználónév

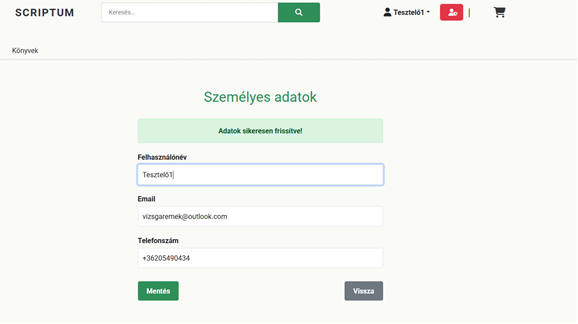
⦁ Email

⦁ Telefonszám



17. ábra - Személyes adatok kezelésének bemutatása

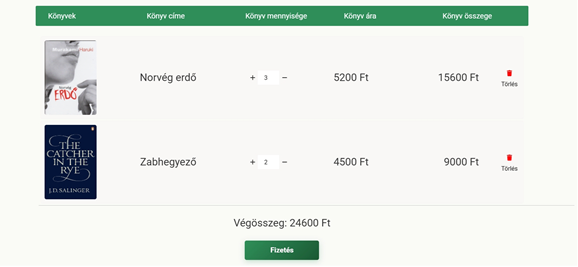
Ha átszeretné szerkeszteni a felhasználó az adatait esetleg, vagy megváltozott az email cím vagy telefonszáma, akkor itt megtudja tenni, szimplán csak átírja az adatait és a mentés gombbal már el is mentette a frissített adatokat. Amennyiben véletlen nyomott rá, vagy mégsem szeretné megváltoztatni az adatait, akkor a vissza gombbal a főoldalra fogja visszairányítani, illetve abban az esetben is, ha megváltoztatja az egyik régi adatát.



18. ábra - Sikeres adatszerkesztés

## 2.13 Vásárlás folyamatának bemutatása

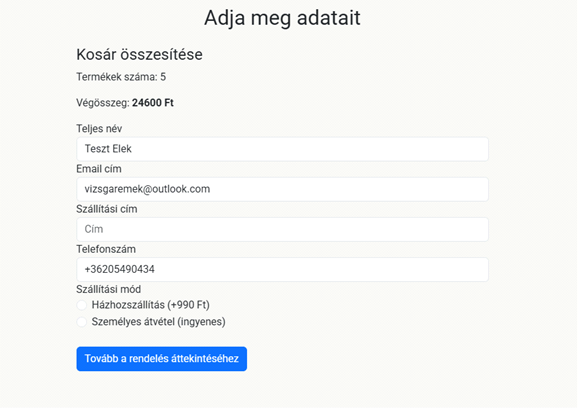
Kosárba tett könyvek megrendelése lehetséges, ha belepakolta a kosárba kellendő könyveket. Amennyiben megtörtént, nyomjon rá a ’’Fizetés” feliratú zöld gombra a fizetés megindításához.



19. ábra – Kosárban lévő könyvek vásárlása

### 2.13.1 Szállítási adatok megadása

Az első oldalon megkell adni az adatokat. A Szállítási címen és a Szállítási módon kívül az oldalon előre megadjuk az adatait, amennyiben ezek megvannak adva. Az első sorokban a kosár tartalmát jeleníti meg, pl. Termékek száma: 5, illetve az összes könyv összértéke (24600 FT). Majd jönnek a személyes adatok megadása, illetve a szállítási mód kiválasztása (még kezdetleges fázisban van, de további fejlesztés alatt ál). Amennyiben kitöltötte az összes személyes adatot, illetve a szállítási módot, nyomjon a “Tovább a rendelés áttekintéséhez” feliratú kék gombra.



20. ábra - Szállítási adatok megadása

A 20. ábra alapján látható a kötelező mezőket mint:

⦁ Név

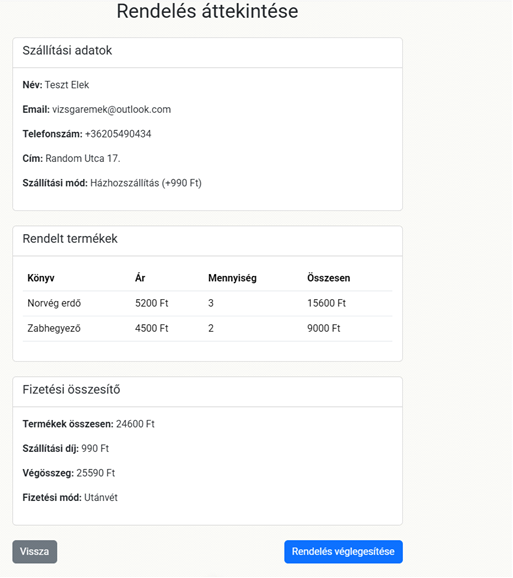
⦁ Email

⦁ Telefonszám

⦁ Cím

⦁ Szállítási mód (Amennyiben házhoz szállítási módot választotta az összárhoz hozzáadódik a plusz költség)

### 2.13.2 Rendelés áttekintése



21. ábra - Rendelési adatok áttekintése

A Rendelt termékek könyvek szerinti lebontását a 21. ábrán láthatja.

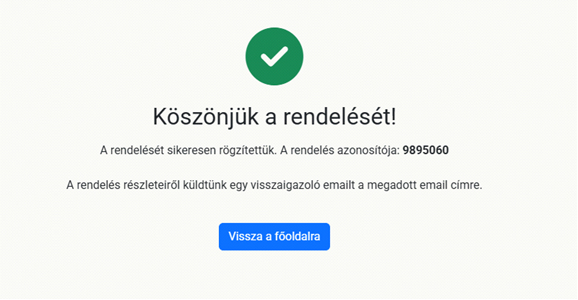
Pl. Norvég erdő könyv, 5200ft 1db könyv ára, amelyből 3 van a kosárba, így a végén a 3 könyv 15600ft

A Fizetési összesítőben pedig a tényleges összköltséget lehet megnézni.

Ha végzett mindezek átnézésével nyomja meg a “Rendelés véglegesítése” feliratú gombot

### 2.13.3 Rendelés véglegesítése

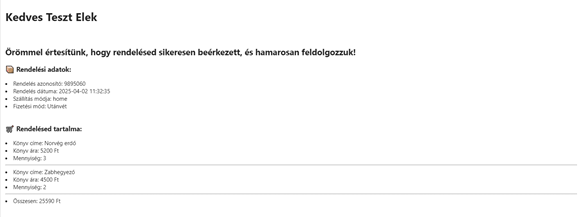
Ha megnyomta a gombot, akkor a következőt kell, hogy lássa. Elküldjük a rendelés visszaigazoló emailt az email címére. A rendelés azonosítóval tudja magát azonosítani



22. ábra - Rendelés elküldése

### 2.13.4 A rendelés véglegesítéséről kapott email

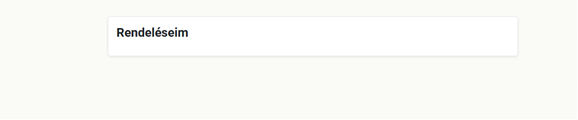
Az email:



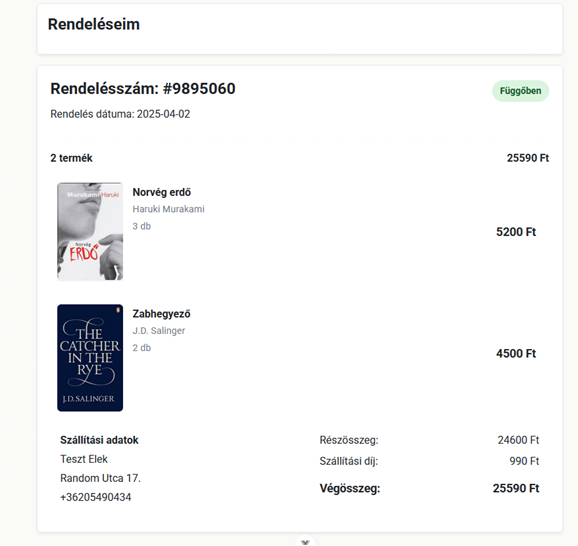
23. ábra - Rendelés visszaigazoló email

## 2.14 Rendeléseim menüpont bemutatása

A Rendeléseim részben csak akkor lesz tartalom, ha történt vásárlás a fiókon, ha nem történt még, akkor ez fogad



24. ábra - Rendeléseim nézet megtekintése, ha üres a rendelés

Amennyiben igen:

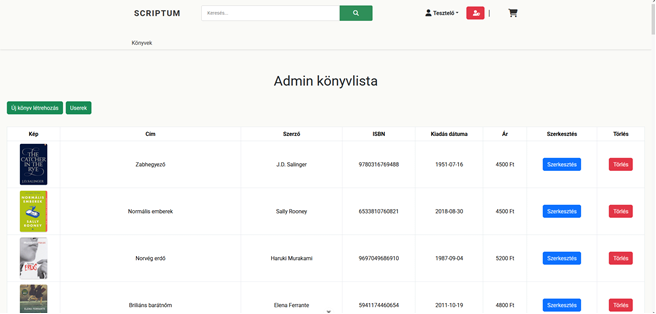
25. ábra - Rendelések áttekintése

## 2.15 Weboldal bemutatása Adminként

Az admin felületet csak admin jogosultsággal rendelkezők tekinthetik meg, ha esetleg valaki felhasználói szinten szeretne belépni pl. Beírja a keresőbe, hogy /admin, akkor a route guard segítségével az illetőt visszadobja a főoldalra.

### 2.15.1 Admin Főoldal

Ha van admin jogosultság, akkor a fentiekben bemutatott képen rajta van egy piros gomb, ha arra rányom, akkor ez az oldal fogadja:



26. ábra - Admin főoldal bemutatása

Itt tudja a könyveket szerkeszteni, törölni, illetve a táblázat bal felső sarkában van két zöld gomb. Az egyikben új könyveket tudunk létrehozni, a második pedig a felhasználókat jeleníti meg és a jogosultságukat tudjuk megváltoztatni. Alapból mindenki, aki beregisztrál, annak felhasználói jogosultsága van.

Felhasználók adatainak megtekintése

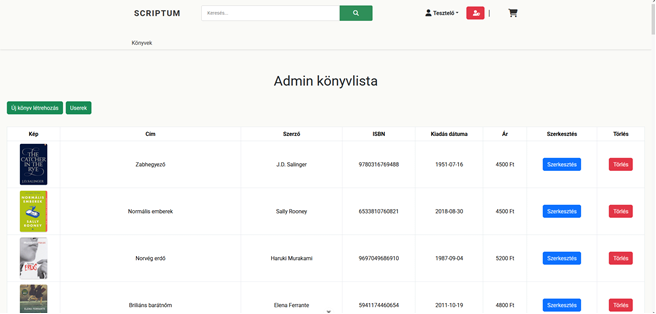


27. ábra - Felhasználók adatinak kezelése

A "Szerkesztés" nevű gombra rákattintva szerkesztheti a jogosultságát az adott felhasználónak. Admin és User között tud választani.

Új könyv létrehozása

Az új könyv létrehozás gombra kattintva tud új könyvet létrehozni



28. ábra - Új könyvek létrehozása Admin oldalon

Új könyvet a következőképpen lehet létrehozni:



29. ábra - Új könyv létrehozása

⦁ Cím - Írja be az új könyv nevét

⦁ Szerző - Írja be az új író nevét

⦁ ISBN – nyomja meg a generálás gombot, hogy egy 13 hosszú számsorozatot, amivel azonosítjuk a könyveket

⦁ Kiadási dátum: Kötőjellel írja le Év-hónap-nap

⦁ Ár: Adja meg a könyv árát

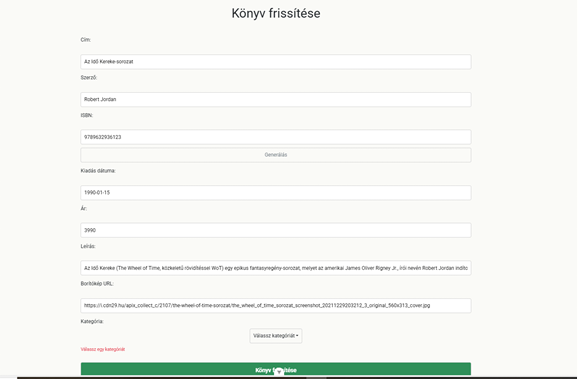
⦁ Leírás: Adja meg hogy mi a könyv leírása

⦁ Borítókép URL: adja meg szöveges formába a képet, ami a borítóképe lesz a könyvnek

⦁ Kategória: Egy lenyíló listában válassza ki, hogy a könyv milyen kategóriába kerüljön

Meglévő könyvek frissítése:

Ha frissíteni akarja az egyik könyvet, nyomjuk meg a szerkesztés gombot az adott könyvnél. A frissíteni kívánt könyv adatai megjelennek, ezeket átszerkesztve ugyanazt a könyvet vagy más könyvet frissített adatokkal tudja hozzáadni a könyvekhez.



30. ábra - Egy meglevő könyv adatinak frissítése

## 3 Fejlesztői dokumentáció

### 3.1 Munkamegosztás:

A munkamegosztás folyamata a Trello oldalán található meg.

### 3.2 Telepítési dokumentáció

Backend elindítása

Vizsga projektmappa > cd Backend

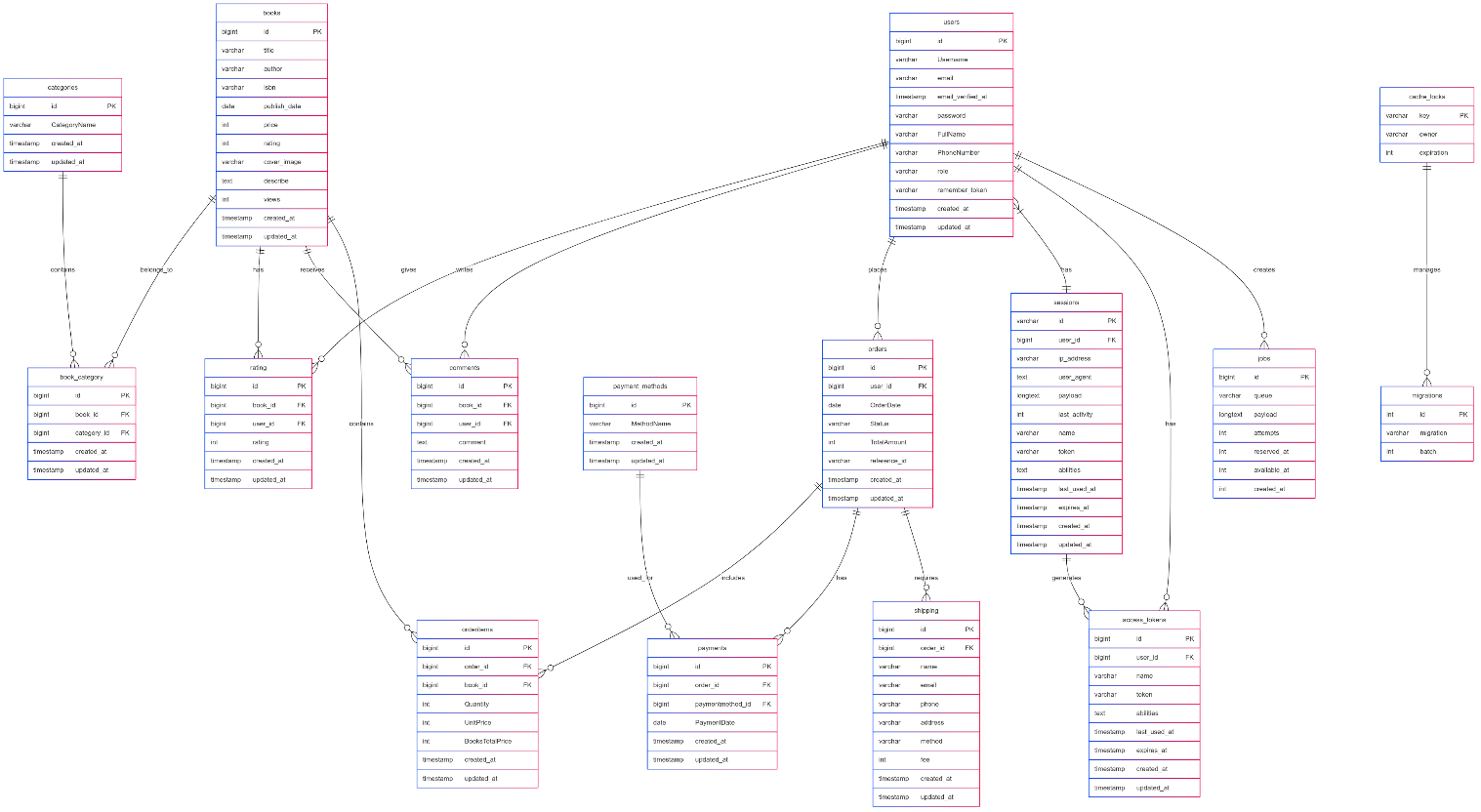
1. Composer install: Evvel létrejön a vendor mappa, amely szükséges a projekt elindításához és működéséhez.
2. Ha nincs elindítva, akkor el kell indítani a XAMPP-ot, hogy a következő lépések sikeren letudjanak futni, illetve élővé váljon a lokális szerverünk.
3. php artisan migrate: Létrejött az adatbázist, illetve a táblázatok és az oszlopok.
4. php artisan db:seed: A seederek tartalmát az adott táblákba betölti, innentől kezdve lesz adat az adatbázisban, amivel lehet dolgozni.
5. php artisan serve: Elindul a server, innentől kezdve már nem kell vele foglalkozni, következhet a frontend elindítása.

Frontend elindítása

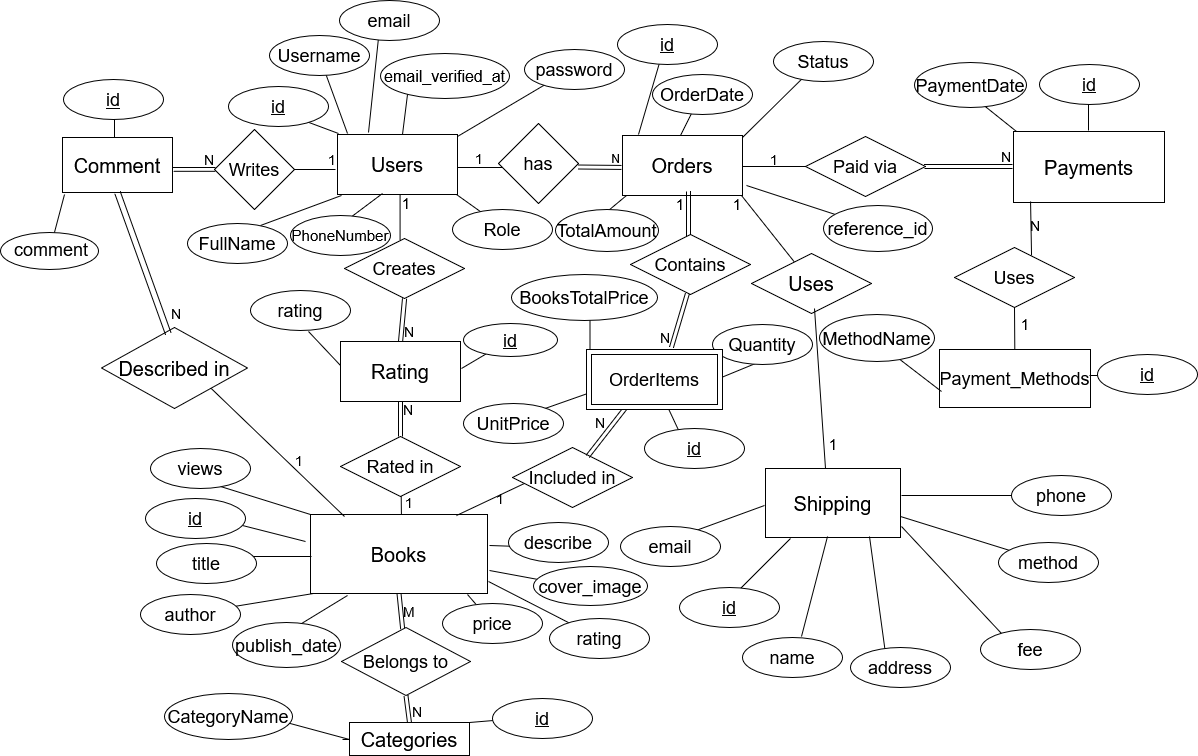
Vizsga projektmappa > cd Frontend

1. npm i: Létrehozza az npm mappát, amely szükséges a frontend helyes működéséhez.
2. npm run dev: Elindítja a tényleges oldalt, ha ctrl+ bal clickkel rákattint a http://localhost:3000 vagy ezt bírja a böngésző url sávjába, látni fogja az oldalt.

### 3.3 Adatbázis



31. ábra – EK diagram



32. Egyedi kapcsolatok diagramja

### 3.3.3 Mezők részletes ismertetése a táblák szerint

Books (id, title, author, isbn, publish\_date, price, rating, cover\_image, describe,views)

Categories (id, categoryname)

Book\_Category (pivot tábla) (id, book\_id, category\_id)

Comments (id, book\_id, user\_id, comment)

Orders (id, user\_id, orderdate, status, totalamount, reference\_id)

Shipping (id, order\_id, name, email, phone, address, method, fee)

Users (id, username, email, email\_verified\_at, password, fullname, phonenumber, role, remember\_token)

Payment\_methods (id, methodname)

Payments (id, order\_id, paymentmethod\_id, paymentdate)

Orderitems (id, order\_id, book\_id, quantity, unitprice, booktotalprice)

### 3.3.4 Kapcsolatok indoklása, ismertetése

User — Rating (1:N)

* Egy felhasználó több értékelést adhat
* Egy értékelés egy felhasználóhoz tartozik

User — Comment (1:N)

* Egy felhasználó több hozzászólást írhat
* Egy hozzászólás egy felhasználóhoz tartozik

User — Order (1:N)

* Egy felhasználónak több rendelése lehet
* Egy rendelés egy felhasználóhoz tartozik

User — Book (M:N)

* Egy felhasználó több könyvet értékelhet
* Egy könyvet több felhasználó értékelhet
* Ez egy M:N kapcsolat a Rating modellen keresztül

Book — Category (M:N)

* Egy könyv több kategóriába tartozhat
* Egy kategória több könyvet tartalmazhat

Book — Rating (1:N)

* Egy könyv több értékelést kaphat
* Egy értékelés egy könyvre vonatkozik

Book — Comment (1:N)

* Egy könyv több hozzászólást kaphat
* Egy hozzászólás egy könyvre vonatkozik

Book — OrderItem (1:N)

* Egy könyv több rendelési tételben szerepelhet
* Egy rendelési tétel egy könyvre vonatkozik

Order — OrderItem (1:N)

* Egy rendelés több tételből állhat
* Egy tétel egy rendeléshez tartozik

Order — Shipping (1:1)

* Egy rendeléshez egy szállítási adat tartozik
* Egy szállítás egy rendeléshez tartozik

Order — Payment (1:N)

* Egy rendeléshez több fizetés tartozhat
* Egy fizetés egy rendeléshez kapcsolódik

Payment — PaymentMethod (N:1)

* Több fizetés használhatja ugyanazt a fizetési módot
* Egy fizetés egy fizetési módot használ

## 3.4 Backend komponens/struktúra dokumentáció

### 3.4.1 Migráció

A BookShop alkalmazás adatbázisának szerkezetét Laravel migrációs fájlok definiálják, amelyek strukturáltan és verziókövethetően építik fel az adatbázis sémát. Az alábbiakban részletesen ismertetem a migráció legfontosabb elemeit és a kialakított adatbázis szerkezetét.

### 3.4.2 Rendszertáblák alapszintű bemutatása

Alapvető Laravel rendszertáblák

1. Users tábla (0001\_01\_01\_000000\_create\_users\_table.php)
2. Cache táblák (0001\_01\_01\_000001\_create\_cache\_table.php)
3. Queue rendszer táblák (0001\_01\_01\_000002\_create\_jobs\_table.php)
4. Personal Access Tokens (2025\_01\_27\_00003\_create\_personal\_access\_tokens\_table.php)

Üzleti logikához kapcsolódó táblák

1. Kategóriák (2025\_01\_27\_000003\_create\_categories\_table.php)
2. Könyvek (2025\_01\_27\_000004\_create\_books\_table.php)
3. Fizetési módok (2025\_01\_27\_000006\_create\_payment\_methods\_table.php)
4. Rendelések (2025\_01\_27\_000007\_create\_orders\_table.php)
5. Rendelési tételek (2025\_01\_27\_000008\_create\_orderitems\_table.php)
6. Fizetések (2025\_01\_27\_000009\_create\_payments\_table.php)
7. Könyv-kategória kapcsolótábla (2025\_01\_28\_103123\_create\_book\_category\_table.php)
8. Értékelések (2025\_02\_27\_112634\_create\_rating\_table.php)
9. Hozzászólások (2025\_03\_03\_110612\_create\_comments\_table.php)
10. Szállítás (2025\_03\_11\_082616\_create\_shipping\_table.php)

Módosító migrációk

1. Rendelési tételek összár (2025\_03\_13\_110643\_add\_total\_price\_to\_orderitems\_table.php)

3.4.3 Főbb táblák részletes leírása

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

33. ábra - Users tábla

A 33. ábra a Users táblát mutatja be, amelyben a következő oszlopok találhatóak meg. Megjegyzés! Az unique metódus arra szolgál, hogy ugyanolyan pl. felhasználónévvel ne rendelkezzen két felhasználó az adatbázisban.

* Id: egyedi azonosító, amely
* Username: Szöveg, a felhasználó nevét tároljuk el.
* email: Szöveg, a felhasználó email címét tároljuk el.
* email\_verified\_at: időbélyeg, email megerősítés, amely időpontként tárolja el, a beregisztrált emailt a hamis email cím regisztrációk ellen
* password: Szöveg, a felhasználó jelszava
* FullName: Szöveg, a felhasználó teljes neve
* PhoneNumber: Szöveg, a felhasználó telefonszáma
* role: Szöveg, a felhasználó jogosultsági szintje

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

34. ábra - Books tábla

A 34. ábra a Books tábla oszlopait mutatja:

* title: Szöveg, könyv címét tárolja
* author: Szöveg, könyv szerzőjének nevét tárolja
* isbn: Szöveg, könyvek azonosító száma, amely 13 karakter hosszú
* publish\_date: Dátum, a könyv kiadásának ideje (Év-Hónap-nap)
* price: Egész szám, könyv értékét tárolja
* rating: Egész szám, a könyv értékelését tárolja, kezdőértéke 1, és 1-5-ig lehet értékelni
* cover\_image: Szöveg, a könyv borítóképét tárolja, amely nullable vagyis nem kötelező megadni
* describe: Hosszú szöveg, a könyv leírását tárolja el
* views: Egész szám, a könyvek megtekintését számolja, amely kiindulópontként 0, mivel alapjáraton amikor a felhasználó megnyitja az oldalt, még nem kattintott rá a könyvre

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

35. ábra - Book-Category kapcsolótábla (Many-to-Many)

35. ábra a Book-Category kapcsolótáblát mutatja be. A kapcsolótáblát vagy Laravelben PIVOT táblaként nevezett táblát több a többhöz (M: N) kapcsolatkor használjuk a relációs adatbázisokban a következő előnyök miatt:

* Tiszta adatmodell: Minden entitás (könyv, kategória) csak a saját adatait tárolja
* Hatékony keresés: Könnyen megtalálhatók az összetartozó elemek
* Rugalmasság: Könnyen adhatunk hozzá vagy távolíthatunk el kapcsolatokat
* Bővíthetőség: A kapcsolótábla később kiegészíthető további mezőkkel
* Adatbázis-integritás: A külső kulcsok és CASCADE törlési szabályok biztosítják az adatok konzisztenciáját

3.4.4 Kapcsolatok és megszorítások

A migrációs fájlokban több kulcsfontosságú kapcsolat és megszorítás is definiálva van:

Idegen kulcs kapcsolatok



36. táblázat - OrderItems tábla idegenkulcsok

Értékelések → egyedi megszorítás



37. táblázat - Rating adatbázis

A könyv-kategória kapcsolótáblában:



38. táblázat - Kategória kapcsolótábla

3.4.5 Migrációk végrehajtása

A migrációk sorrendje fontos, mivel az idegen kulcs kapcsolatok miatt a hivatkozott tábláknak már léteznie kell. A Laravel migrációs rendszere automatikusan kezeli ezt a timestamp alapú névkonvenciókkal:

1. Először a rendszer alapvető táblái jönnek létre (0001\_01\_01\_\*)
2. Ezután a független entitás táblák (books, categories, payment\_methods)
3. Végül a függőségekkel rendelkező táblák (orders, orderitems, payments, stb.)

3.4.6 Indexek és teljesítmény optimalizáció

Az adatbázis szerkezet több kritikus indexet is tartalmaz:

* Minden id mező automatikus indexelése
* Egyedi indexek a Username, email, FullName és PhoneNumber mezőkön
* Idegen kulcs indexek a kapcsolt táblákon

3.4.7 További fejlesztési lehetőségek

Az adatbázis migrációjában még fejleszthető területek:

* Soft delete funkcionalitás bevezetése a kritikus táblákhoz
* Teljes szövegű keresés támogatása

### 3.4.8 Seederek

A seedereket az adatbázisban előre megadott, hard codolt adatokat adunk az adatbázisnak, úgymond teszt adatokra alkalmazzuk, hogy tesztelhető legyen az oldal.

DatabaseSeeder

Az alkalmazás fő seeder osztálya, amely az összes többi seedert megfelelő sorrendben futtatja. A Laravel keretrendszer automatikusan ezt hívja meg a php artisan db:seed parancs futtatásakor.

#### CategorieSeeder

A könyvkategóriák adatainak feltöltéséért felelős osztály. A [categories.json](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) fájlból tölti be az adatokat a categories táblába.

BookSeeder

A könyvek adatainak feltöltéséért felelős osztály. A [books.json](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) fájlból tölti be az adatokat a books táblába.

Book\_CategorySeeder

A könyv és kategória közötti kapcsolótábla (book\_category) feltöltéséért felelős. A [book\_category.json](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) fájlból tölti be a kapcsolatokat.

UserSeeder

A felhasználók adatainak feltöltéséért felelős osztály. A [users.json](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) fájlból tölti be az adatokat, miközben a jelszavakat titkosítja.

PaymentMethodSeeder

A fizetési módok feltöltéséért felelős osztály. Közvetlenül a kódban definiált fizetési módokat tölti fel.

Kapcsolatok

* Sorrendi kapcsolat: DatabaseSeeder biztosítja az adatfüggőség alapján kialakított betöltési sorrendet
* Adatfüggőség: Először a független entitások (kategóriák, könyvek, felhasználók) töltődnek be, majd a kapcsolótáblák adatai (book\_category)
* Osztály-JSON kapcsolat: Minden seeder (kivéve PaymentMethodSeeder) egy specifikus JSON adatfájlból tölti be az adatait

Főbb elem részletes ismertetése

DatabaseSeeder

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

39. ábra – Többi seeder meghívása a main seederbe

A DatabaseSeeder feladata az adatbetöltés koordinálása. Fontos a meghívás sorrendje, amely biztosítja, hogy az egymásra épülő adatok (például a book\_category kapcsolótábla) csak a megfelelő táblák feltöltése után kerüljenek betöltésre.

JSON fájlok struktúrája:

A JSON formátum mindig egy tömbbel kezdődik, amely objektumokat tárol/tárolnak. Az objektumnak kulcs-érték párból áll.

categories.json:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

40. ábra – Kép a kategória seederről JSON formátumban

A categories seederben 12 kategória név van eltárolva:

* Modern Irodalom
* Klasszikus Irodalom
* Disztópia
* Romantikus regény
* Amerikai regény
* Kalandregény
* Fantasy
* Történelmi regény
* Pszichológiai regény
* Epikus fantasy
* Horror
* Erotikus

books.json:

58 könyv adatait tartalmazza részletes információkkal

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

41. ábra – Zabhegyező című könyv adatainak bemutatása

A 41. ábrán látható módon tárolja az adatokat a books.json seeder a következőkkel

* title: A könyv címét tárolja
* author: A könyv írójának neve
* isbn: A könyv azonosítójának adatai. 13 számjegyű
* publish\_date: A könyv kiadásának ideje
* price: A könyv eladási értéke
* cover\_image: String formátumban a könyv borítójának képe
* describe: A könyv rövid leírása, bemutatása

books\_category.json:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

42. ábra – Könyvek és kategóriák összekapcsolása

Az összes könyv összekapcsolása az adott kategóriával/kategóriákkal

users.json:

A users seederben vannak tárolva az előre megadott felhasználók

A képen szöveg, képernyőkép, Betűtípus, képernyő látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

43. ábra – Kép a users seederben tárolt előre megadott tesztelő fiók adatiról

* Username: A felhasználó nevét tárolja
* email: A felhasználói email
* password: A felhasználó jelszava
* FullName: A felhasználó igazi, teljes neve
* PhoneNumber: A felhasználó telefonszáma
* email\_verified\_at: Az email validációjának időpontját tárolja
* role: A felhasználó jogiszintje

### 3.4.9 Seederek működése

CategorieSeeder

A kategóriák alapvető adatait tölti be. Minden kategória rendelkezik egy egyedi azonosítóval és egy névvel. Ezek a kategóriák alkotják a könyvek rendszerezési alapját.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

44. ábra – Kategória json adatok betöltése a Kategória seederbe

BookSeeder

A könyvek részletes adatait tölti be, beleértve a címet, szerzőt, ISBN számot, kiadási dátumot, árat, borítóképet és a leírást.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

45. ábra – Könyvek json adatok betöltése a Könyvek seederbe

Book\_CategorySeeder

A könyvek és kategóriák közötti many-to-many kapcsolatot hozza létre, lehetővé téve a könyvek kategóriák szerinti rendszerezését és szűrését.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

46. ábra – Könyvek és a Kategóriák közötti kapcsolótáblába való adatok feltöltése

UserSeeder

Létrehozza az alapértelmezett felhasználói fiókokat, különös tekintettel az adminisztrátorokra. A jelszavakat megfelelően titkosítja a Hash::make() függvény segítségével.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

47. ábra – Felhasználók adatainak beszúrása a táblázatba, a jelszó utólagos hasheléssel

PaymentMethodSeeder

Az elérhető fizetési módokat tölti be, amelyek később a megrendelések fizetési módjának kiválasztásához használhatók.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

48. ábra – Fizetési módok megadása

3.4.10 Adatbetöltés folyamata

* A php artisan db:seed parancs meghívja a DatabaseSeeder osztály run() metódusát
* A DatabaseSeeder sorban meghívja a többi seedert a definiált sorrendben
* Minden seeder beolvassa a hozzá tartozó JSON fájl tartalmát (kivéve PaymentMethodSeeder)
* A beolvasott adatokat a seeder a megfelelő adatbázis táblába tölti
* A UserSeeder extra műveletet végez a jelszavak titkosítására
* Ez a seeder struktúra biztosítja, hogy:
* Az alkalmazás alapadatai minden telepítésnél konzisztensek legyenek
* A tesztadatok rendelkezésre álljanak fejlesztési és demonstrációs célokra
* Az összetett adatszerkezetek (kapcsolótáblák) helyesen épüljenek fel
* A seedert a php artisan db:seed paranccsal lehet végrehajtani, vagy egy friss migráció részeként a php artisan migrate:fresh --seed paranccsal.

### 3.4.11 Modell

Laravelben a Modellek fontos részét képezik az MVC architektúrájának. Figyeli, kezeli, hogy az adatok miként használja az applikáció. A Laravel modellek az adatbázisban lévő táblákat jelenítik meg, lehetővé téve az alapvető műveletek végrehajtását – például lekérdezést, frissítést és adatok törlését – kifejező, ember által olvasható kódon keresztül. Az alapvető adatbázis-interakciókon túl a Laravel modellek lehetővé teszik, hogy üzleti logikát közvetlenül a modellekhez adjon, így a kód modulárisabbá és rendszerezettebbé válik.

A Modellek célja: A modellek egy erőműként a háttérben kezelik és rendezik az alkalmazás adatait, hogy könnyen interaktálhassunk az adatbázisunkkal bármi erőfeszítés nélkül.

A modellek hidat képeznek az adatbázis táblákhoz, mivel azokat reprezentálják. Például: van egy User táblánk, aminek van egy User modellje, a Modell fogja reprezentálni a táblát, ezáltal könnyen írhatunk SQL sorokat pl. CRUD műveletek.

Kapcsolatok kiépítése a táblák között: A leggyakoribb kapcsolatok, amelyek előfordulnak egy adatbázisban.

1:1 --> egy az egyhez kapcsolat (ez a legegyszerűbb). Rendelés és Kiszállítás: 1 rendeléshez 1 kiszállítási cím tartozhat, és 1 kiszállítási címhez 1 rendelés tartozhat

1: N --> egy a többhöz kapcsolat: Rendelés és Felhasználók között. 1 felhasználóhoz több rendelés is hozzátartozhat, viszont 1 rendelés 1 felhasználóhoz tartozhat.

M: N --> több a többhöz kapcsolat: Könyvek és a kategóriák. 1 könyvhöz több kategória is tartozhat, és egy kategóriához több könyv is tartozhat. Ehhez szükséges PIVOT táblát (kapcsolótábla) létrehozni relációs adatbázis esetén.

A modelleknek meglehet adni, hogy milyen néven adja meg a tábla nevet, ha esetleg az, amit a Laravel automatikusan ad nem lenne megfelelő, átírhatjuk.



49. ábra – Tábla név beállítás

Még a modellek tartalmaznak úgymond beépített adatbiztonsággal. A modellek olyan funkciókat tartalmaznak, mint a tömeges hozzárendelés védelme és az attribútumok átadása, amelyek megkönnyítik az adatok biztonságos kezelését és megakadályozzák a gyakori sebezhetőségeket. Mit takar ez a gyakorlatban?

Ha például szeretnénk a táblázatunkból az oszlopneveket majd megjeleníteni valamilyen formában, azt megkell adni.



50. ábra – Használható oszlopok megadása

A következő modellek vannak használva a projektben:

Book.php

Category.php

Comment.php

Order.php

OrderItem.php

Payment.php

PaymentMethod.php

Rating.php

Shipping.php

User.php

Főbb kapcsolatok

A fentiekben említett tábla kapcsolatok alapján a következő kapcsolatok vannak:

Könyv kapcsolatai:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Category.php | M: N |
| Book.php | OrderItem.php | 1: N |
| [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Rating.php | 1: N |
| [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Comment.php | 1: N |

1. táblázat Könyv kapcsolatok rendszerezése

Felhasználó kapcsolatai:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| User.php | Order.php | 1: N |
| User.php | Rating.php | 1: N |
| User.php | Comment.php | 1: N |

2. táblázat Felhasználói kapcsolatok rendszerezése

Rendelés kapcsolatai:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| Order.php | OrderItem.php | 1: N |
| Order.php | Payment.php | 1: N |
| Order.php | Shipping.php | 1:1 |
| Order.php | User.php | N:1 (fordítva nézve) |

3. táblázat Rendelés kapcsolatok megjelenítése

Fizetés kapcsolatai:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| Payment.php | PaymentMethod.php | N:1 |
| Payment.php | Order.php | N:1 |

4. táblázat Fizetés kapcsolatok megjelenítése

Egyéb kapcsolatok:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| Rating.php | User.php | N:1 |
| Rating.php | [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | N:1 |
| Comment.php | User.php | N:1 |
| Comment.php | [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | N:1 |
| OrderItem.php | [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | N:1 |
| OrderItem.php | Order.php | N:1 |
| Shipping.php | Order.php | 1:1 |
| PaymentMethod.php | Payment.php | 1: N |

5. táblázat Egyéb kapcsolatok megjelenítése

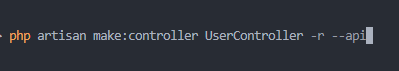
Speciális kapcsolat:

|  |  |  |
| --- | --- | --- |
| Model | Model | Kapcsolat |
| [Book.php](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | User.php | |Távoli elérés a Rating modellen keresztül (hasManyThrough) |

6. táblázat Speciális kapcsolat megjelenítése

### 3.4.12 Controllerek:

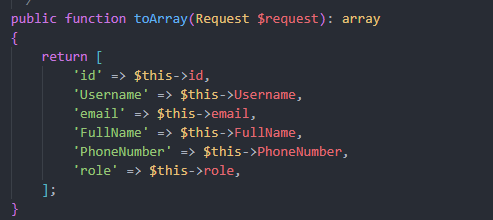
Az MVC arhitektúrának másik része. A Controllerek elkülönítik a kéréseket, a route-ok és a nézetek közt. A Controllerek HTTP kéréseket kezelnek le. Az átlagos controller metódusok a következők: index(), show(), store(), update(), és a destroy(), ezek felelnek meg a CRUD műveleteknek. Ezeken felül lehet még pl a create() is. Ezeket a következő artisan console paranccsal lehet a leggyorsabban létrehozni:



51. ábra - API Controller artisan parancsal való létrehozása

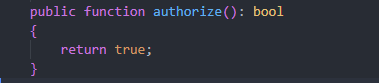
A Controllereknek van middleware-jei ami egy köztes hídként értelmezhető a kérések és a reakciók között. A middlewarek egy úgymond szűrőrendszerként viselkedik, amely azt figyeli, hogy valami ellenőrizve van vagy sem. Az adatok kezelésére hasznos alkamazni az API resource-ket,illetve requesteket a tisztább, szervezettebb és biztonságosabb kódbázis eredményéhez. Mit mire használunk?

Az API Resource-ok segítenek az adatok strukturált formázásában és a válaszok egységesítésében. Előnyei közé tartozik a könnyen formázható adatok, az egységes API válaszok, illetve teljesítmény javítás is. A resourceket collection osztályokba csoportosítjuk, hogy logikus adatszerkezet jöjjön létre, következetes URL-struktúra, jól illeszkedik a HTTP metódusokhoz, egymásba ágyazhatók pl. /users/{id}/orders. Illetve a skálázhatóság. A resource osztály reprezentál egy modellt, amelyet JSON formátummá kell átalakítani.



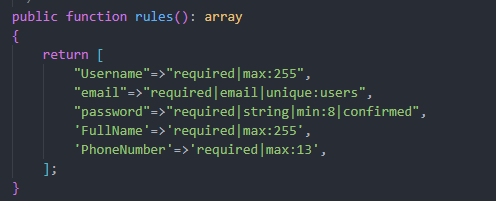
52. ábra - API Resource

A Requesteket validációra és adatok ellenőrzésének egységes módját biztosítják. A tisztább controllerekért, az automatikus jogosultság kezelésért, illetve a code újrahasználhatóságáért szükséges használni. Két alap metódusa van, az egyik az authorize, amelynek feladata eldönteni, hogy a kérés feldolgozható-e vagy sem.



53. ábra - Request authorize

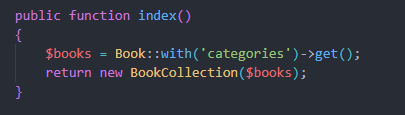
Az 53. ábrán látható módon, Ha true értéket ad vissza, akkor a kérés sikeres volt, folytatódhat a validáció. Ellenben, ha false, akkor a Laravel automatikusan 403 Forbidden hibával fog visszatérni. Lehetséges különböző speciálisabb kéréseket is írni bele.



54. ábra - Request validációs szabályok

A másik metódus, pedig a rules, amely a tényleges validációs szabályokat jelenti. Az 54. ábrán látható módon megírjuk a validációs szabályokat, amelyek vonatkozni fognak a Controllerekben behivatkozott metódusokra.

3.4.13 Ezeket pedig a Controllerben feltudjuk használni. Bemutatom a használatukat a BookControlleren keresztül:

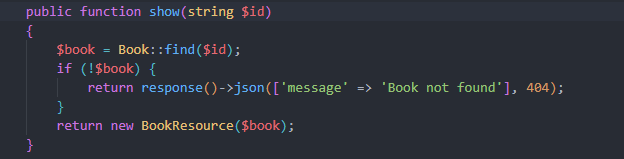


55. ábra – Minden könyv lekérése kategóriákkal együtt

Az index metódusban lekérjük a könyveket, nem csak egy könyvet, ezért visszatérünk egy BookCollectionnel, amely a kategóriákat is magába foglalja

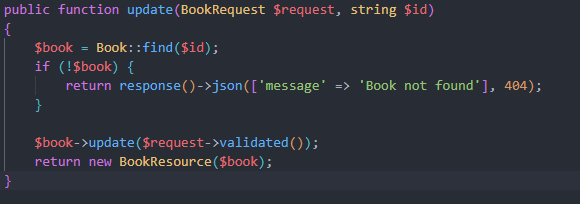
56. ábra – Új könyv eltárolása/létrehozása

A store metódus POST-ot küld a pl egy új könyv létrehozásakor, hogy elmentse az új adatokat az adatbázisba. A metódusba használjuk a megírt request osztályt validálásra, létrehozzuk az új könyvet, és visszaadjuk a json formátumba a választ

57. ábra – Egy könyv adatainak megjelenítése

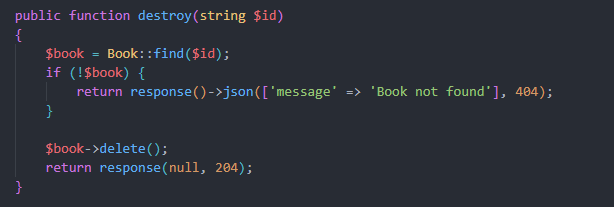
A 57. ábrán is látható módon jelenítjük meg egy könyv adatait. Így most resource-ot használunk, nem pedig collection-t.

Id alapján megkeressük az adatot könyvet, ha van, akkor visszadjuk, ha nincs akkor pedig error message-et dob a server.

58. ábra – Egy létező könyv adatainak szerkesztése

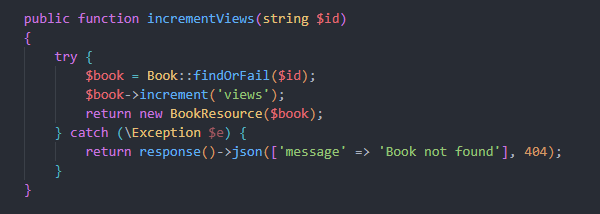
Update metódus egy eddig meglévő adat frissítésért felel, jelen esetben egy eddigi könyv adatainak frissítésére.

Először megkeressük a frissíteni kívánt könyvet id alapján, ha nem találjuk errort dobunk, ha létezik, akkor updateljük, validáljuk és a visszakapjuk a BookResource-t

59. ábra – Egy könyv törlése

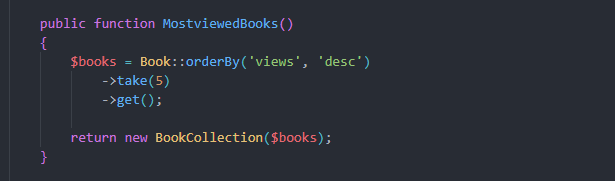
Destroy metódus, pedig a törlésért felel. 59. ábra esetében, megkeressük, mint eddig ID alapján a törölni kívánt könyvet, majd ugyanúgy megnézzük, hogy ha nem találjuk az adott könyvet, akkor errort dobunk, más különben töröljük az adott könyvet, majd egy választ visszaadunk, hogy sikeres volt.

Általában vannak még specifikus metódusok, amelyeket mi magunk definiálunk, pl. A könyvek esetén megszámoljuk, hogy hányszor nézték meg, vagyis hányszor kattintottak az adott könyvre, ezt a következőképpen lehet megtenni.

60. ábra – Megtekintett könyvek számlálása

Megkeressük ID alapján a könyvet, a books táblán van egy views oszlop, amelybe számoljuk, hogy hányszor kattintottak rá, az increment-el hozzáadjuk a views oszlophoz, majd ezt visszadjuk az adott könyvnek.

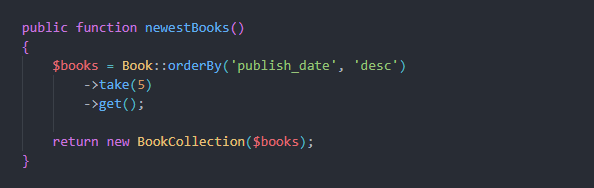
A többi metódusok a BookControllerben:

61. ábra – Legtöbbet megtekintett könyvek metódus

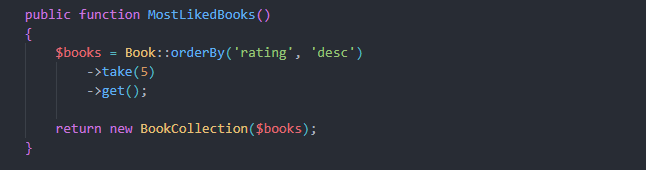
Könyveket a megtekintésük alapján rendezzük a views oszlop számlálásának köszönhetően, csökkenő sorrendben. Kiveszünk 5 könyvet, majd ezeket a BookCollectionbe visszaadjuk.

62. ábra – Legrégebbi könyvek metódus

Itt a legrégebben megjelent könyveket keressük rendezés alapján, amely a publish\_date oszlopban növekvő sorrendbe nézi, ugyanúgy 5 könyvet kiválasztunk és visszakapja a collection.

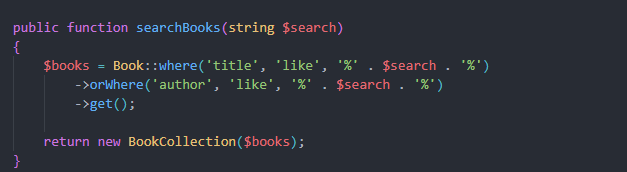
63. ábra – Legújabb könyvek metódus

Legújabban kiadott könyveket keressük a metódussal, amely csökkenő sorrendbe rakja a kiválasztott 5 könyvet a publish\_date oszlop szerint.

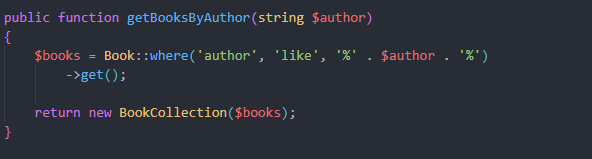
64. ábra - Legjobban értékelt könyvek keresése

Rendezés alapján a Book modellből, amely kapcsolatban van a Books táblával, annak a rating oszlopát csökkenő sorrendbe rendezzük, abból 5-öt kiveszünk és azt megkapjuk.

65. ábra - Legtöbb kommenttel rendelkező könyvek

66. ábra - Keresés funkció implementálása

Keresés implementálása backenden, hogy a könyveket keresni lehessen a címük, és az írójuk által.

67. ábra - Keresés a könyv írója szerint is

Külön keresés implementálása azoknak a könyveknek a megjelenítésére, amelyek egy íróhoz tartoznak.

### 3.4.13 További resource, requestek, illetve controllerek:

Resource:

AuthCollection

AuthResource

BookCollection

BookResource

CategoryCollection

CategoryResource

CommentCollection

CommentResource

OrderCollection

OrderResource

RatingCollection

RatingResource

Requestek:

AuthRequest

BookRequest

CategoryRequest

CommentRequest

LoginRequest

OrdersRequest

RatingRequest

UserUpdateRequest

Controllerek:

AuthController

BookController

CategoryController

CommentController

Controller

OrdersController

RatingController

### 3.4.14 Authentikáció

A mai világban már elengedhetetlen a felhasználó lecsekkolása a weboldalakon, hisz így nagyobb felhasználói élményt tudunk nyújtani, plusz nem utolsó, hogy adatokat tudunk az adott felhasználóról pl., hogy nem hamis adatokkal regisztrált az oldalra. Az authentikáció folyamat implementálása bonyolult és biztonsági kockázatokat rejthet, erre

biztosít a Laravel egy biztonságos eszközt, hogy gyorsan, biztonságosan és könnyen implementálhassuk az oldalunkon bármi probléma nélkül.

Az authentikáció működése Laravelben:

Laravelben alapjáraton az adatbázisában a users táblában van egy remember\_token nevezetű oszlop, amely 100 karakter hosszú szöveget képes befogadni. Ez az oszlop felel a felhasználó regisztrált adatainak eltárolására token formában.

Az authentikáció működése pedig, hogy amikor használunk egy weboldalt, a felhasználó megadja a felhasználónevét, illetve a jelszavát, amelyet úgynevezett sessionökben, vagyis munkamenetekben tárol az oldal. A böngészőnek kiadott cookie tartalmazza a session ID-t így az alkalmazáshoz intézett későbbi kérések a felhasználót a megfelelő sessionhoz társíthatják. Miután a session cookie beérkezett, az alkalmazás a session ID alapján lekéri a munkamenet adatait, amely adatokat eltáról a sessionben és így be is van jelentkeztetve a felhasználó. Azonban a session és cookiekat nem igen használunkk API használatokor. Ehelyett a API tokenekkel fogunk API-kal küldözgetni.

Laravelben két opció közül lehet választanim, ha API token alapú authentikációt szeretnénk. Passport és a Sanctum. A projektben Sanctum van használatban, mert mivel könnyebb implementálni és használni, mint másik társát, amely egy két faktoriális authentikáció, amely komplexebb és biztonságilag kockázatosabb implementálni, mint a Sanctumot.

Laravel Sanctum használata a weboldalon:

Az API tokeneket arra találták ki olyan oldalak, mint pl. Github és társai, hogy elkerüljék az authentikációkkal járó problémákat. Az API tokenek sokkal hosszabb ideig, akár évekig is tarthat a lejárati idejük, de a felhasználó manuálisan bármikor visszavonhatja. A Laravel Sanctum ezt a szolgáltatást úgy kínálja, hogy egyetlen adatbázistáblában tárolja a felhasználói API-tokeneket, és hitelesíti a bejövő HTTP-kéréseket az Authorization fejlécen keresztül, amelynek tartalmaznia kell egy érvényes API tokent. Az SPA vagyis single-page applications, mint például ez a weboldal is használ a sessionök védelmének érdekében egy úgynevezett CSRF védelmet, amelyet az XSS(Cross Site Scripting) támadások ellen találtak ki.

Laravel Santum letöltéséhez a következő artisan parancsot kell beírni:

A képen képernyőkép, szöveg, Betűtípus, fekete látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

68. ábra – Laravel artisan command api csomag letöltésére

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

69. ábra – API beállítása a User modellben

Miután letöltöttük az API csomagot, az ábrán lévő 3 dolgot kell implementálni a User modellben, hogy sikeresen működjön az authentikáció.

* HasFactory
* Notifiable
* HasApiTokens

Ezt a 3-at kell, hogy használja az use kulcsszóval.

Ezek után létre kell hoznunk a tokent a kívánt helyen, jelen esetben a AuthControllerben a megfelelő metódusban, ami jelenesetben a register.

A képen képernyőkép, Betűtípus, Grafika, szöveg látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

70. ábra – Authentikációs token létrehozása

A createToken metódus visszaadja a Laravel\Sanctum\NewAccessToken példányt.

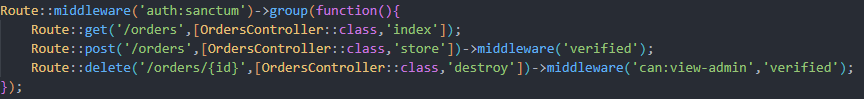
Az API tokenek hashelt értékek, amelyek az SHA-256 hashelés típust használják, mielőtt eltárolják az adatbázisban. Ezt elérhetjük szövegesen is a planTextToken hozzáadásával.  
A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

71. ábra – Authentikációs token létrehozása a login metódusban is

### 3.4.15 Router védelem

Mikor letöltöttük az API csomagot, létrejött a routes mappában az api.php route file-ok.

72. ábra – API route létrehozása

Létrehozzuk a Route facadel, majd a middleware segítségével (amelyről a Controllerek részben volt szó) kiszűri a sanctum authentikációval jelen esetben, hogy az /orders-hez kapcsolódó dolgokat csak regisztrált felhasználó férhessen hozzá.

### 3.4.16 Authorization / Jogosultságok

Laravelben bevan építve a jogosultságkezelési lehetőség, például olyan esetekben, amikor adott felhasználóknak nem szeretnénk, hogy olyan jogaik legyenek a weboldalon, amely veszélyforrásként is tekinthető. Ennek korlátozására két opció létezik a Laravelben:

* Gates
* Policies

A Gatek egyszerű megközeletést alkalmaznak, vagyis nem lehet túlzott logikai dolgokat írni bele, mert erre van a Policies. Mindkettőt egyszerre lehet használni, nem zárja ki egyik sem a másik használatát.

A Gateseket általában a App\Providers\AppServiceProvider osztályban lévő boot metódusban definiáljuk Gate facade használatával definiálunk egy gate-et

A projekt esetében egy jogosultság kezelés történt:

A képen képernyőkép, szöveg, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.73. ábra – Gates megírása az AppServiceProvidersben

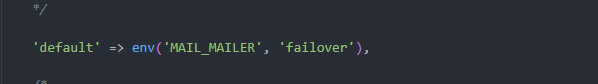
A 73. ábrán látható Gate működése: definiáltuk, hogy a User modellből kinyert felhasználói adatok alapján, ha a felhasználónak a jogosultsági szintje egyenlő adminnal, akkor engedélyezzük, ha nem akkor kiírjuk az elutasítási szöveget, amelyet az ábrán is lehet látni.

A 73. ábrán látható módon ugyanúgy használhatjuk, mint a sanctum authentikációt, hogy az’’view-admin” felhasználjuk a middlewareben, így, már figyelni fogja a jogosultság kezelést a Laravel.

### 3.4.17 Email

Laravelben az email küldéséhez többféle email szolgáltatást lehet igénybe venni. A Laravel egy egyszerű, és tiszta email API-t nyújt a Symfony Mailer komponens által.

Laravel és a Symfony Mailer vezérlőket nyújt az e-mailek küldésére, mint pl. SMTP, Mailgun, Postmark, Amazon SES, és sendmail. Néhány ingyenes, de a legtöbb fizetős szolgáltatás. A projektben SMTP szolgáltatást választottuk, mert az SMTP egy ingyenes szolgáltatás, illetve könnyű konfigurálni, viszont sokkal lassabb, mint bármely másik email szolgáltatás. Mi az SMTP? Az SMTP egy levelezőrendszer, amely a feladó által küldött e-maileket, egy vagy több címzettnek továbbítja a hálózati protokollnak megfelelően az interneten keresztül. Főbb funkciója, a beépített spam elleni hitelesítési mechanizmusok. Laravelben lehetséges egyszerre több email szolgáltatást használni, mint pl. Postmarkot tranzakciós e-mailek küldésére, míg Amazonas SES-t tömeges e-mailek küldésére egy azon applikáción belül.

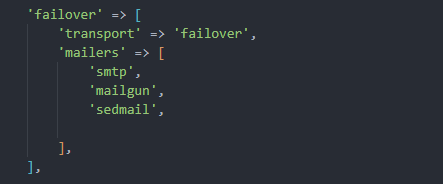


74. ábra – Email beállítása

Az email beállításának elkezdéséhez először a config/mail.php-ban kell beállítani. Többféle beállítás van, attól függően, hogy melyik szolgáltatást vesszük igénybe. Jelen esetben az SMTP beállítása lesz.

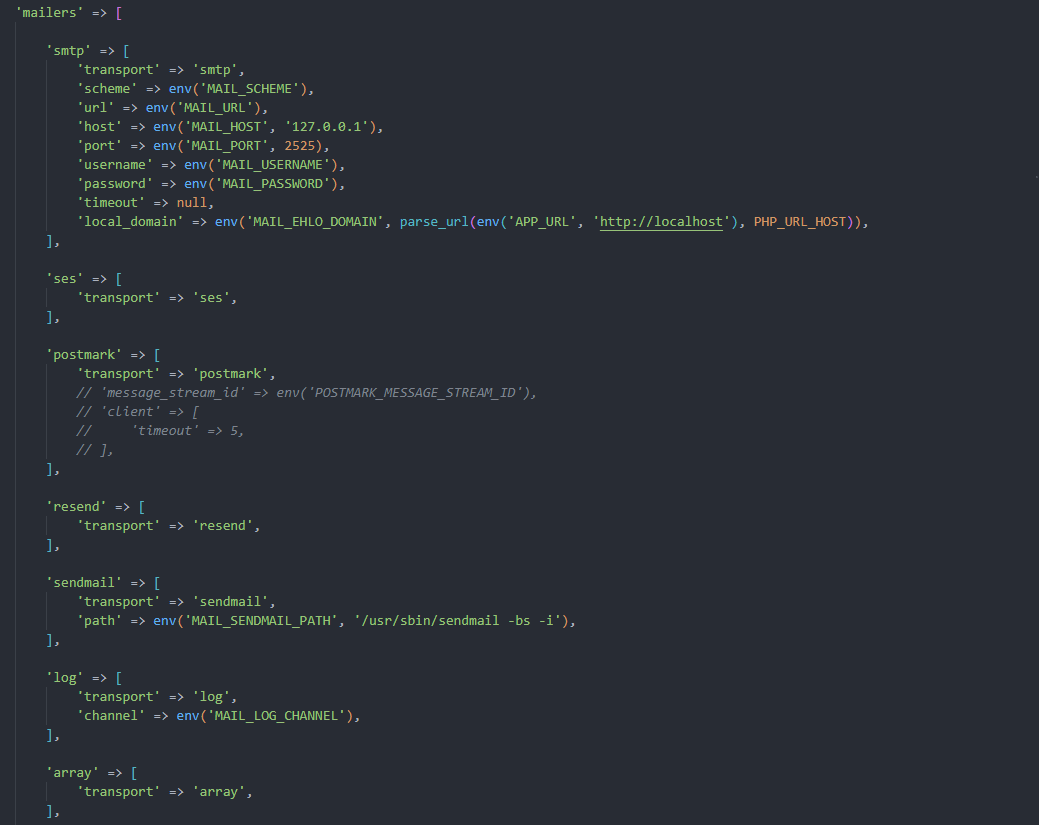
Először bekell állítani, hogy milyen szolgáltatást akarunk alapból, mint ahogy a 74. ábra is mutatja.

A failover annyit tesz, hogy ha esetleg az alkalmazásunkban az egyik email szolgáltatás nem működne, akkor a failoverben pluszban beírt email szolgáltató/szolgáltatók foknak működni.



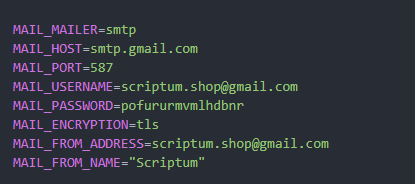
75. ábra – Failover beállítása

Jelen esetben az SMTP szolgáltatás van legfelül, így azt fogja alkalmazni.



76. ábra – Képernyőfotó a teljes config/mail.php-ról

Majd miután a felsorolt lépésekkel megvagyunk, következhet az .env fájlban az SMTP beállítása.

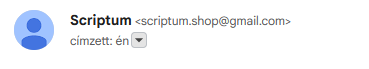


77. ábra -. env email beállítás

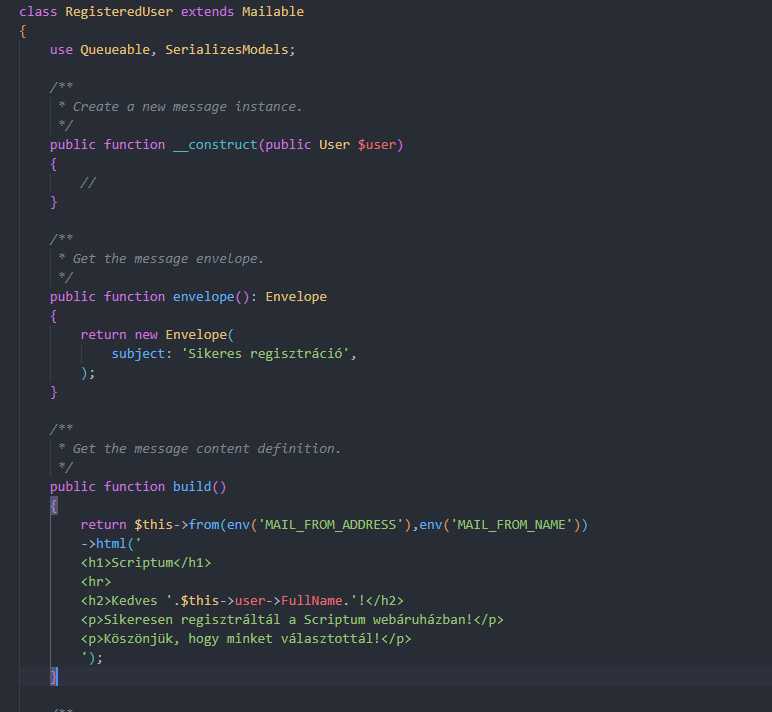
Ezekből a lépésekből áll a beállítása:

Ahogy a 77.-es ábra is mutatja, először magát az email szolgáltatás típusát kell beírni, ami jelen esetben az SMTP. Majd az email hostolás-t. Több email portot beírhatunk az SMTP-el kapcsolatban pl. 25,465, de a leggyakoribb és legjavasolttabb az a 587-es port. Majd beállítjuk, hogy mi legyen a username, majd egy a gmail által adott alkalmazás jelszót állítunk be, amelyet csak akkor tudunk elkérni, ha kétfaktoros hitelesítés bevan állítva. Ez a Google 2022 májusától beiktatott “kevésbé biztonságos alkalmazások” nem engedélyezése miatt kell, illetve fokozza a fiók védelmét.

Majd a SMTP és más email szolgáltatások által használt TLS titkosítást kell beállítani. Ezek után azt az email-t írjuk be, amiről szeretnénk küldeni az emaileket, jelen esetben a Scriptum online könyvesbolt dedikált email fiókjáról. És persze hogy milyen néven jelenjen majd meg az emailben.



78. ábra – Az email küldő oldal email profilja

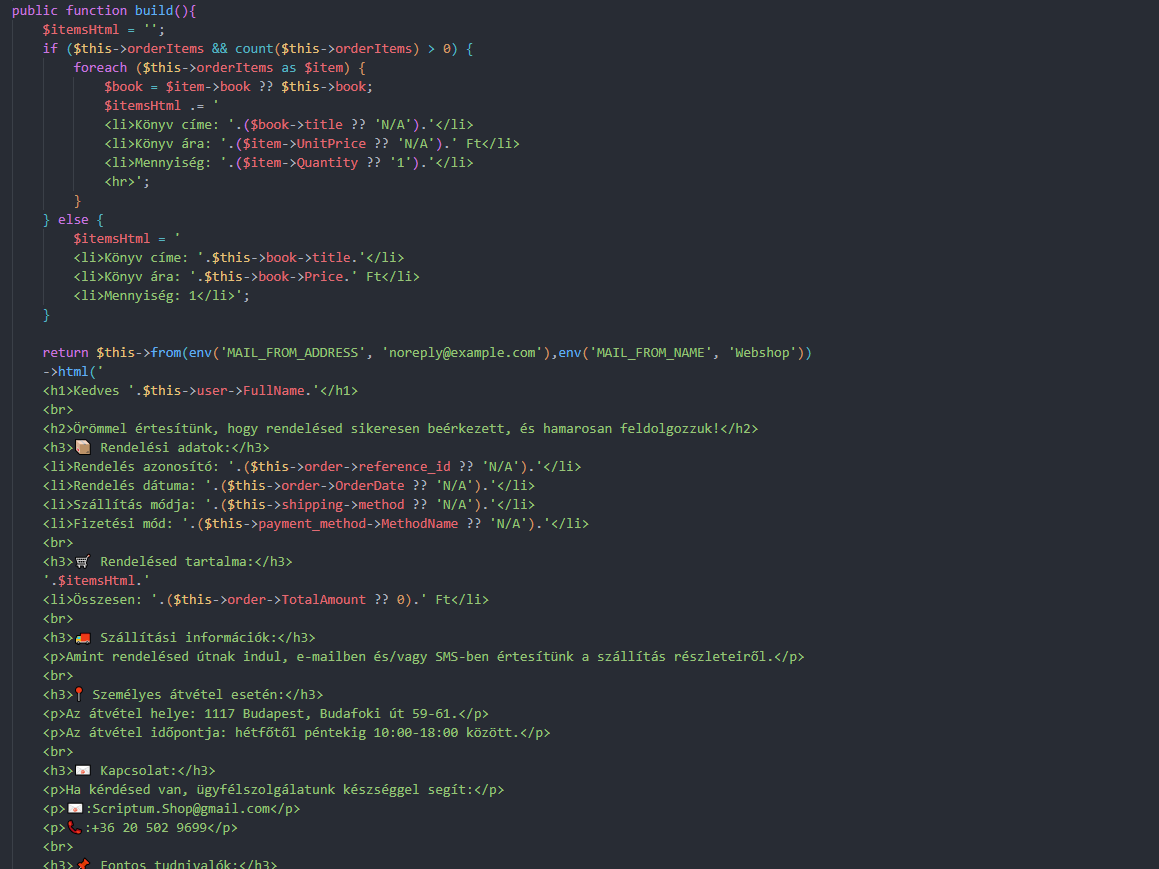


79. ábra – Mail osztály létrehozása és tartalma

Egyszerű email elkészítése:

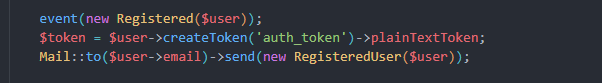
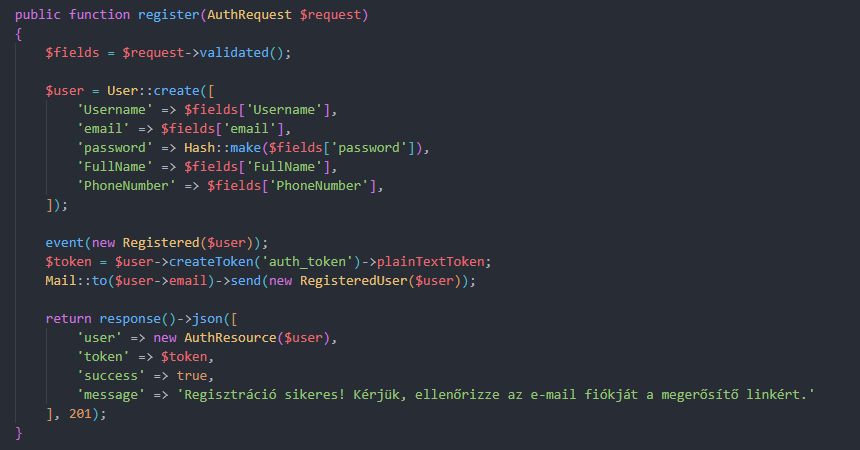
Php artisan make:mail azMailNeve paranccsal létrehozzuk az email-t. Van egy \_\_construct metódusa, amelybe public User $user-ként meghívjuk a User model-t, annak érdekében, hogy felhasználhassam a User modelben lévő adatokat. Pl. mint, ahogy a 79. ábrán is látható, a felhasználó teljes nevét hivatkozom be az emailbe a User model behívásával.

A képen képernyőkép, szöveg látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

80. ábra – Teljes beumatatása a rendelés emailnek

Az envelope metódus az email tárgyát reprezentálja, míg, ha nem server side oldalt írunk, akkor kell nekünk egy build metódus, amelyben megadjuk, hogy kinek akarjuk elküldeni, és html formában megformázzuk a levelet.



81. ábra – Sikeres regisztrációs email elküldése

Már csak annyi a dolgunk, hogy megadjuk, hogy hol és mikor adjuk át az adott email-t.

A 81. ábra a sikeres regisztrációt követő, regisztrációs email elküldését mutatja be.

létre kell hoznunk egy eventet, amely azt figyeli, hogy regisztrált a felhasználó, amennyiben igen, a Mail::to metódussal elküldjük az emailt a felhasználó megadott email címére.

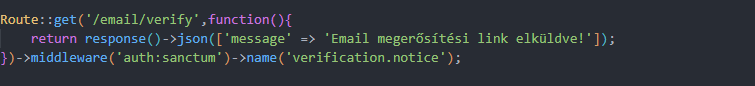
Az email megerősítéséhez, illetve az elfelejtett jelszó visszaállítás funkciót is, külön API routingban kell létrehozni.



82. ábra – Verifikációs email

A route a verifikációs emailt küldi újra, ha az esetleg nem érkezett volna meg, vagy lejárt volna a token, mielott megnyitotta volna.

* Megkeresi a felhasználót az email alapján
* Ellenőrzi, hogy létezik-e ilyen felhasználó
* Ellenőrzi, hogy az email még nincs megerősítve
* Újra elküldi az email verifikációs értesítést
* JSON választ ad a művelet eredményéről



83. ábra – Email megerősítés

Cél az, hogy visszajelezzünk az email verifikációs folyamatról.

* Működése az, hogy egy egyszerű JSON üzenetet ad vissza, ami tájékoztatja a felhasználó.
* A middleware által csak bejelentkezett felhasználó férhetnek hozzá API tokennel



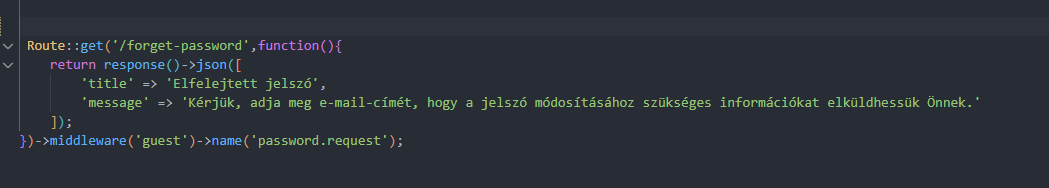
84. ábra – Email küldése

84. ábra módon kezeljük az email verifikációs linket.

Működése:

* Megkeresi a felhasználót az ID alapján
* Ellenőrzi a hash érvényességét a felhasználó email címével
* Ha a hash hibás, 400-as hibát ad vissza
* Ha az email már korábban meg lett erősítve, átirányít a login oldalra "már megerősítve" paraméterrel
* Ha még nem volt megerősítve, beállítja az email-t ellenőrzöttként és átirányít a login oldalra

Elfelejtett jelszó implementálása:



85. ábra – Frontend elfelejtett jelszó oldal router beállítása

A /forget-password route teszi lehetővé, hogy majd a frontenden megírt componensen elkért emailen keresztül küldjön egy jelszó megváltoztató emailt.

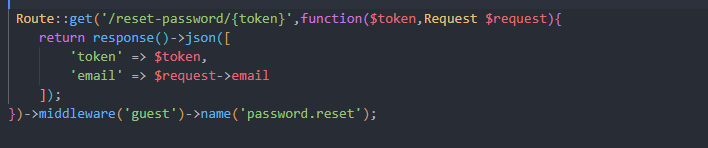
* A middleware által figyelve van, hogy csak be nem jelentkezett felhasználó férhessen hozzá



86. ábra - Jelszó-visszaállítási link küldése

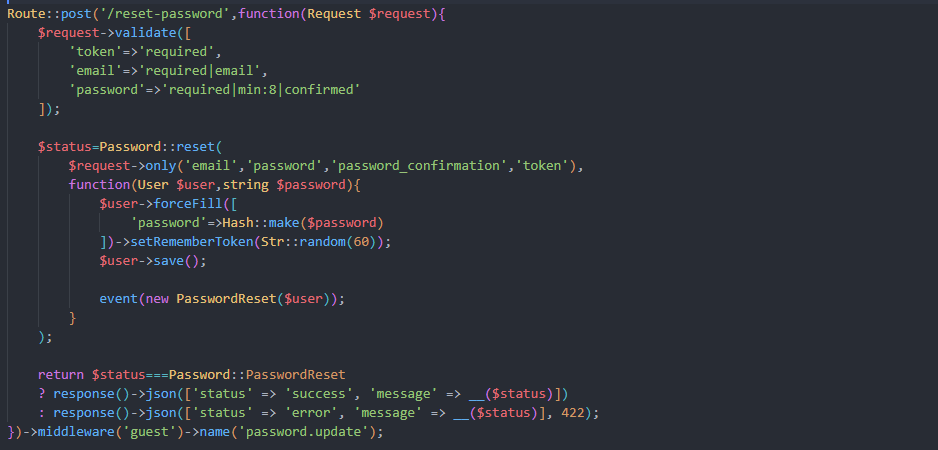
Ez a route küldi el a jelszó-visszaállítási linket az email-re. Ellenőrzi, hogy a kérés tartalmaz-e érvényes email címet.

* A Password facade segítségével küld egy visszaállítási linket a megadott címre.
* Sikeres/sikertelen választ küld JSON formátumban.



87. ábra – Token visszaadása az email kattintása esetén

* Amikor a felhasználó rákattint az emailben kapott jelszó-visszaállítási linkre, visszaadja a tokent és az email címet JSON formátumban.
* A {token} az a visszaállításhoz szükséges egyedi tokent képzi.



88. ábra – Új jelszó beállítása

Ez a route azt kezeli, amikor a felhasználó elküldi az új jelszavát a visszaállítási oldalon.

* Működése annyiban merül ki, hogy validálja a beérkező adatokat (token, email, jelszó, jelszó megerősítés) és Password facade segítségével végrehajtja a jelszó visszaállítását.
* A belső callback függvény frissíti a felhasználó jelszavát (titkosítva). Egy új “emlékezz rám” tokent állít be.
* A PasswordReset egy eseményt küld el, majd sikeres vagy sikertelen választ küld el JSON formátumban.

### 3.5 Frontend Komponens/struktúra dokumentáció

3.5.1 Frontend Komponensek

Könyv megjelenítési komponensek

* Main.vue - Főoldal komponense, az összes könyvlista konténere
* Item.vue - Általános könyvlista megjelenítő komponens
* ItemDetails.vue - Könyv részletes adatait megjelenítő komponens
* AuthorOtherBooks.vue - Egy szerző további könyveit megjelenítő komponens
* MostViewedItem.vue - Legnézettebb könyvek megjelenítése
* MostLikedBooks.vue - Legkedveltebb könyvek megjelenítése
* MostCommentedBooks.vue - Legtöbbet kommentelt könyvek megjelenítése
* NewestBooks.vue - Legújabb könyvek megjelenítése
* LatestBooks.vue - Legrégebbi könyvek megjelenítése
* Rating.vue - Könyvek értékelését kezelő komponens
* Comment.vue - Könyvhöz tartozó hozzászólások kezelése

Felhasználói komponensek

* Login.vue - Bejelentkezési űrlap
* Register.vue - Regisztrációs űrlap
* Email.vue - Jelszó-visszaállítási email kérése
* ResetPassword.vue - Új jelszó megadása
* UserDataSettings.vue - Felhasználói adatok módosítása
* UserOrderDatas.vue - Felhasználó rendeléseinek megtekintése

Vásárlási komponensek

* Cart.vue - Kosár tartalmának megjelenítése
* CartButton.vue - Kosár oldalra navigáló gomb
* UserDatas.vue - Vásárlási folyamat (rendelési adatok, szállítás, fizetés)

Admin komponensek

* AdminBooks.vue - Könyvek adminisztrálása (CRUD műveletek)
* CreateBooks.vue - Új könyv létrehozása
* UpdateBooks.vue - Könyv adatainak szerkesztése
* UserSettings.vue - Felhasználók listázása adminoknak
* RoleSettings.vue - Felhasználói jogosultságok módosítása

Kategória komponensek

CategoryDetails.vue - Kategóriához tartozó könyvek megjelenítése

Elrendezési komponensek

* App.vue - Az alkalmazás gyökérkomponense
* TheHeader.vue - Az alkalmazás fejléce
* Footer.vue - Az alkalmazás lábléce
* TheDropdown.vue - Könyv kategóriák lenyíló menüje
* Card.vue - Újrafelhasználható könyvkártya komponens
* ReturnButton.vue - Előző oldalra navigáló gomb
* NotFound.vue - 404-es hiba oldal

3.5.2 Kapcsolatok

Komponens hierarchia

* App.vue → TheHeader.vue, router-view, Footer.vue
* Main.vue → Item.vue, MostViewedItem.vue, MostLikedBooks.vue, MostCommentedBooks.vue, NewestBooks.vue, LatestBooks.vue
* ItemDetails.vue → Rating.vue, Comment.vue, AuthorOtherBooks.vue, ReturnButton.vue
* AdminBooks.vue → router-view (UpdateBooks.vue, CreateBooks.vue)
* Card.vue → felhasználja: Item.vue, AuthorOtherBooks.vue, LatestBooks.vue, stb.

3.5.3 Vuex ismertetése

A Vuex a Vue.js alkalmazások állapotkezelő könyvtára, amely központosított tárolóként (store) szolgál az alkalmazás összes komponense számára. Biztosítja az adatok egységes kezelését és a komponensek közötti kommunikációt.

Miért használjunk Vuex-et?

* Központosított állapotkezelés: Az alkalmazás állapotát egy helyen tárolja
* Kiszámítható állapot változások: Az állapotváltozások explicit módon történnek
* Komponensek közötti kommunikáció egyszerűsítése
* Fejlesztői eszközökkel való integráció: Vue DevTools támogatás az állapotváltozások követésére

Vuex Store modulok

* book - Könyvekkel kapcsolatos állapotkezelés
* user - Felhasználókkal és autentikációval kapcsolatos állapotkezelés
* cart - Kosár tartalmának kezelése
* category - Kategóriák kezelése
* comment - Hozzászólások kezelése
* rating - Értékelések kezelése
* search - Keresési eredmények kezelése
* order - Rendelési folyamat kezelése

State (Állapot)

A state tartalmazza az alkalmazás összes megosztott adatát.

A képen szöveg, képernyőkép, képernyő, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

89 ábra – BooksVuex index.js

3.5.4 Osztályok/Modulok

A Scriptum könyvesbolt alkalmazás modularizált Vuex állapotkezelő rendszert használ, amely nyolc különálló funkcióorientált modulra van felosztva, amelyekre hivatkozva tudjuk az állapottereinket használni

A képen szöveg, képernyőkép, képernyő, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

90. ábra - Központi Vuex Store

Modul struktúra

Minden modul azonos szerkezetű, a következő fájlokkal:

* index.js: Kezdeti állapot, modulkonfiguráció
* actions.js: Aszinkron API műveletek és üzleti logika
* mutations.js: Állapotmódosító függvények
* getters.js: Számított tulajdonságok és lekérdezések

Modulok közötti kapcsolatok

Kereszt-modul hívások: A modulok közötti kommunikáció a {root: true} paraméterrel történik



91. ábra - RatingVuex/actions.js

Adatmegosztás: Bizonyos modulok más modulok állapotára támaszkodnak



92. ábra - RatingVuex/actions.js

3.5.5 Főbb modulok részletes ismertetése

User Modul (Felhasználókezelés)

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

93. ábra – UserVuex index.js

Kiemelt művelet | Felhasználó hitelesítés

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

94. ábra – UserVuex actions.js login metódus

* JWT token kezelés: A bejelentkezés után a token localStorage-ban tárolódik
* Szerepkör kezelés: Admin és felhasználói jogosultságok különválasztása
* Jelszó-visszaállítás: Email-alapú jelszó-visszaállítási folyamat

Book Modul (Könyv adatkezelés)

A képen szöveg, képernyőkép, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

95. ábra – BooksVuex index.js bemutatása

Kiemelt művelet | Könyvek lekérése különböző szűrők alapján

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

96. ábra – BooksVuex actions.js, a legtöbbet megtekintett könyv megjelenítése

* Nézettségi számláló: Könyvek megtekintéseinek követése
* Kategória szerinti szűrés: Könyvek kategóriák szerinti leválogatása
* Értékelés frissítése: Könyvek értékeléseinek frissítése a kapcsolódó modulokon keresztül

Cart Modul (Kosár funkciók)

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.97. ábra - CartVuex/index.js

Kiemelt művelet

Kosárba helyezés:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.98. ábra - CartVuex/mutations.js kosár tartalom hozzáadás

* Mennyiség kezelés: Elemek hozzáadása/törlése egyenként
* Teljes kosár törlése: Vásárlás befejezése után a kosár kiürítése
* Lokális adatmegőrzés: Kosár adatainak perzisztálása localStorage-ben

Order Modul (Rendeléskezelés)

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

99. ábra – OrderVuex/index.js állapotkezelés

Kiemelt művelet

Rendelés feldolgozása:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

100. ábra – OrderVuex/actions.js rendelés leadás kezelése

* Rendelés létrehozása: Kosár tartalmának és felhasználói adatoknak az egyesítése
* Rendelés követése: Lépésenkénti folyamat követése állapotváltozókkal
* Szállítási díj kalkuláció: Kiszállítási módtól függő díjszámítás

Rating Modul (Értékelések)

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

101. ábra - RatingVuex/index.js állapotkezelés

Kiemelt művelet

Értékelések kezelése:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

102. ábra - RatingVuex/actions.js értékelés hozzáadás a könyvhöz

* Felhasználói értékelések nyilvántartása: Felhasználónként és könyvenként
* Átlagértékelés számítás: Computed property könyv értékelésekhez
* Felhasználói interakciók: Új értékelések, módosítások, törlések kezelése

Általános tervezési elvek és minták

Modul szerkezet egységesítése

Minden modul konzisztensen követi a Vuex javasolt felépítését:

export default {

namespaced: true,

state() { */\* kezdeti állapot \*/* },

mutations: { */\* állapotmódosító függvények \*/* },

actions: { */\* aszinkron műveletek \*/* },

getters: { */\* számított tulajdonságok \*/* }

}

103. ábra – Vuex felépítése

Hibakezelési stratégiák. A modulok egységes hibakezelési mintát követnek:

try {

*// API művelet végrehajtása*

*// sikeres válasz feldolgozása*

return true; *// sikeres művelet jelzése*

} catch (error) {

console.error('Hibatípus leírása:', error);

if (error.response) {

console.error("Válasz adat:", error.response.data);

console.error("Válasz státuszkód:", error.response.status);

}

return false; *// sikertelen művelet jelzése*

}

104. ábra – Hibakezelés

Token-alapú autentikáció:

localStorage.setItem("token", response.data.token);

105. ábra - Token-alapú autentikáció

Kosár adatok megőrzése:

localStorage.setItem('cart', JSON.stringify(state.cart));

106. ábra – Localstorage használata adatok megtartására

Modulok közötti kommunikáció:

dispatch('book/updateBookRating', ratingData.book\_id, {root: true});

107. ábra - Dispatch más modulba

Globális getter használata:

const user = this.getters["user/user"];

108. ábra - Teljesítmény-optimalizálás és biztonság

Hatékony állapotlekérdezések

getAverageRating: (*state*) => (*bookId*) => {

const bookRatings = *state*.ratings.filter(*r* => *r*.book\_id == *bookId*);

if (bookRatings.length === 0) return 0;

const sum = bookRatings.reduce((*total*, *rating*) => *total* + *rating*.rating, 0);

return (sum / bookRatings.length).toFixed(1);

}

109. ábra - A getterek hatékony újraszámítási logikát implementálnak

Biztonsági szempontok

* CSRF védelem: Sanctum cookie lekérés API kérések előtt
* Token-kezelés: JWT token tárolás és HTTP fejlécek automatikus beállítása
* Szerepkör ellenőrzés: Admin és felhasználói jogosultságok kezelése

async loginUser({ commit, dispatch }, { email, password }) {

try {

await http.get("/sanctum/csrf-cookie"); *// CSRF védelem*

const response = await http.post("/login", { email, password });

*// ...*

} catch (error) {

*// ...*

}

}

110. ábra LoginUser metódus

Felhasználói élmény optimalizálása

* Betöltési állapotok kezelése: Indikátorok a hosszú műveletek során
* Hibakezelés és hibaüzenetek: Felhasználóbarát hibavisszajelzések

### 3.5.6 Router kapcsolatok

* Oldalak közötti navigáció
* URL paraméterek használata (pl. /books/:id, /categories/:id)
* Bejelentkezéshez kötött oldalak védelme

Main.vue

A könyvesbolti alkalmazás fő oldala, amely a különböző könyvlistákat jeleníti meg (legnézettebb, legkedveltebb, legújabb, stb.).

* Útvonal: / (alapértelmezett kezdőlap)
* Betöltés: Lazy-loading technikával, csak akkor töltődik be, amikor szükséges

Login.vue

Bejelentkezési űrlapot megjelenítő komponens, amely a felhasználók hitelesítését kezeli.

* Útvonal: /login
* Betöltés: Lazy-loading technikával

Register.vue

Regisztrációs űrlapot megjelenítő komponens, amely új felhasználók regisztrálását teszi lehetővé.

* Útvonal: /register
* Betöltés: Lazy-loading technikával

ItemDetails.vue

Egy konkrét könyv részletes adatait megjelenítő komponens.

* Útvonal: /book/:id (ahol az: id a könyv egyedi azonosítója)
* Betöltés: Lazy-loading technikával

Kapcsolatok

* Navigációs folyamat: A komponensek közötti navigáció a Vue Router által kezelve
* Adatáramlás: A router paramétereken keresztül (pl. könyv azonosító az ItemDetails komponensnél)
* Állapotkezelés: Feltehetően Vuex vagy Pinia store-on keresztül együttműködve a routerrel
* Jogosultságkezelés: Router guard-ok segítségével (bár ez nem látható közvetlenül a kódrészletben)

### 3.5.7 Router Konfiguráció

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

111. ábra - Vue router

Lazy Loading Mechanizmus

A router.js fájlban minden komponens dinamikus importálással (lazy loading) van konfigurálva:



112. ábra - Main router

Ez a megvalósítás számos előnnyel jár:

* Jobb kezdeti betöltési idő: Csak a kezdőlaphoz szükséges komponensek töltődnek be azonnal
* Kisebb bundle méret: A komponensek külön chunk-okban kerülnek összeállításra
* Optimalizált erőforrás-használat: Komponensek csak akkor töltődnek be, amikor ténylegesen szükség van rájuk

Navigációs folyamat

A felhasználói navigáció tipikus folyamata:

* Kezdőlap betöltése (Main.vue): Különböző kategóriákba rendezett könyvek megjelenítése
* Könyvrészletek megtekintése (ItemDetails.vue): Egy kiválasztott könyv részleteinek megtekintése a könyv azonosítója alapján
* Felhasználói hitelesítés (Login.vue, Register.vue): Bejelentkezés meglévő fiókkal vagy új felhasználó regisztrálása

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

113. ábra Route beállítások

Router Guard Rendszer

A 113. ábrán látható módon a router guard-ok a következő funkciókat látják el:

* Jogosultság-ellenőrzés: Csak bejelentkezett felhasználók férhetnek hozzá bizonyos útvonalakhoz (pl. profil, kosár)
* Admin ellenőrzés: Admin funkciókat csak admin szerepkörrel rendelkező felhasználók érhetnek el
* Bejelentkezési állapot átirányítás: Bejelentkezett felhasználók átirányítása a bejelentkezési oldalról
* URL paraméterek validálása: Érvényes könyv/kategória azonosítók ellenőrzése

A router guardon felül az adott oldal címét, ahol vagyunk, megtudjuk adni annak megfelelően, amit a router beállításakor megadtunk, mint metatag.



114. ábra - Oldal cím megadása



115. ábra - metatag megadása

Ennek megfelelően, ha a főoldalra lépünk felül megjelenik címként a’’Kezdőlap”

Technológiai megfontolások

A router implementáció a Vue Router modern jellemzőit használja:

HTML5 History Mode: A createWebHistory() használata szebb URL-ek érdekében (nincs # a címben)

* Named Routes: Minden útvonal névvel rendelkezik a programozott navigáció megkönnyítése érdekében
* Route Params: Dinamikus paraméterek (pl.: id) használata
* Meta mezők: Útvonalspecifikus metaadatok definiálása (pl. jogosultsági követelmények)
* Code Splitting: Alkalmazás részekre bontása a jobb teljesítmény érdekében

A Vue Router implementációban különleges figyelmet érdemel a webpack optimalizációs direktívák használata:



116. ábra- Prefetch

* Prefetch: Alacsony prioritású erőforrás-letöltést jelez, amelyet a böngésző tétlenségi idejében végez
* Alkalmazás: Kritikus navigációs célpontokhoz (főoldal, bejelentkezés), amelyekre valószínűleg szükség lesz

Preload Stratégia





117. ábra – Preload

* Preload: Magasabb prioritású betöltés, a jelenlegi navigáció során azonnal szükséges
* Alkalmazás: Részletes adatokat megjelenítő komponensekhez és admin funkcionalitáshoz

Alap Lazy Loading (optimalizáció nélkül)

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

118. ábra - Alap Lazy Loading

* Alapértelmezett: Egyszerű dinamikus importálás webpack direktíva nélkül
* Alkalmazás: Kevésbé gyakran használt komponensekhez

Teljesítményoptimalizálás jelentősége

Webpack direktívák előnyei

1. Kisebb kezdeti bundle méret: Csak a feltétlenül szükséges komponensek töltődnek be kezdetben
2. Párhuzamos letöltések: Több kisebb chunk párhuzamos letöltése gyorsabb, mint egy nagy bundle letöltése
3. Intelligens erőforrás-kezelés:

* Prefetch: Kihasználja a böngésző tétlen idejét
* Preload: Prioritizálja a kritikus erőforrásokat

1. Optimalizált felhasználói élmény:

* Gyorsabb kezdeti betöltés
* Gyorsabb navigáció az alkalmazáson belül
* Minimális várakozási idők a felhasználó számára

Főbb elemek részletes ismertetése

Autentikációs rendszer

A felhasználói azonosítás kezelése a [Uservuex](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) modulban történik:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

119. ábra - UserVuex Login

Az autentikációs token kezelése:

* A token a [localStorage](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)-ban tárolódik
* Az alkalmazás indításakor a token ellenőrzése ([App.vue](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) created hook-ban)
* Kérésekhez automatikus hozzáadás ([http.js](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)-ben)

Könyv megjelenítési rendszer

A könyvek megjelenítését több komponens együttműködése teszi lehetővé:

1. Main.vue: A főoldal különböző könyvlistákat jelenít meg, az azonosításuk a Vuex modulban tárolt adatok alapján történik
2. ItemDetails.vue: Egy könyv részletes adatainak megjelenítése, az adatok lekérése a [book/getDataId](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) action-nel történik:



120. ábra - Könyv adatok lekérése

3. Card.vue: Újrafelhasználható komponens, amely egy könyv alapvető adatait jeleníti meg kártya formátumban

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

121. ábra - Card.vue/Layout

Kosár kezelő rendszer

A kosár kezelése a CartVuex modulban történik, amely a localStorage-t használja a kosár adatainak perzisztens tárolására:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

122. ábra - Item hozzáadás

A [Cart.vue](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) komponens felelős a kosár tartalmának megjelenítéséért és a mennyiségek módosításáért:

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

123. ábra - Cart componens

Rendelési folyamat

A rendelési folyamat 3 lépésből áll, amit az OrderVuex modul orderStep változója követ:

1. Felhasználói adatok megadása
2. Rendelés áttekintése
3. Rendelés visszaigazolása
4. Összegyűjti a szükséges adatokat (kosár, felhasználói adatok)
5. Ellenőrzi az adatok helyességét
6. Elküldi a rendelést az API-nak
7. Beállítja a kész állapotot

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

124. ábra - Rendelés leadás

Értékelési rendszer

A Rating komponens és RatingVuex modul kezeli a könyvek értékelését:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

125. ábra - Értékelés componens

Az értékelés elküldése:

* Ellenőrzi a bejelentkezést
* Ellenőrzi, hogy már értékelte-e a könyvet
* Elküldi az értékelést

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

126. ábra - Rating.vue értékelés elküldése

Keresési rendszer

A keresési folyamat a SearchVuex modulban történik, a TheHeader.vue-ban található keresőmezőhöz kapcsolva:

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

127. ábra - keresés implementálása

Admin terület

Az [AdminBooks.vue](vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) lehetővé teszi a könyvek kezelését adminisztrátorok számára:

* Könyvek listázása
* Új könyv létrehozása
* Könyv szerkesztése
* Könyv törlése

HTTP kommunikáció

Az API-kal való kommunikáció az Axios-on alapuló [http.js](https://vscode-file://vscode-app/c:/Users/Asus/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) segítségével történik.

Az npm által letöltött axiossal, egy create metódusban a baseURL-t beállítjuk, ha kell, akkor a headerst is. Majd a tokeneket is eltároljuk.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

128. ábra - Backend - Frontend kommunikáció

### 3.5.7 Üzleti logika dokumentáció

Állapotkezelés (Vuex)

A Vuex moduláris felépítése lehetővé teszi a különböző üzleti területek elkülönített kezelését:

BookVuex

Felelősség: Könyvek adatainak lekérése, rendezése különböző szempontok szerint (legnépszerűbb, legújabb stb.)

Főbb funkciók:

* Összes könyv lekérése
* Egy könyv adatainak lekérése
* Könyvek szűrése kategória szerint
* Szerző szerinti könyvek lekérése
* Megtekintések számolása

UserVuex

Felelősség: Felhasználói adatok és hitelesítés kezelése

Főbb funkciók:

* Bejelentkezés/Regisztráció
* Felhasználói adatok lekérése
* Jelszó-visszaállítás
* Jogosultságkezelés

CartVuex

Felelősség: Kosár tartalmának és műveleteinek kezelése

Főbb funkciók:

* Tétel hozzáadása a kosárhoz
* Tétel eltávolítása a kosárból
* Tétel mennyiségének módosítása
* Kosár összegeinek számítása
* Kosár tartalmának mentése localStorage-ba

OrderVuex

Felelősség: Rendelési folyamat kezelése

Főbb funkciók:

* Rendelési adatok tárolása
* Rendelési folyamat lépéseinek nyomon követése
* Rendelés véglegesítése és küldése a szervernek
* Szállítási díj számítása

Validáció

A form validáció a vee-validate könyvtárat használja. Példa egy validációra:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

129. ábra - Regisztráció email validáció

Reaktív adatmodell

A Vue.js reaktív rendszerét kihasználva az applikáció hatékonyan reagál az adatváltozásokra:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

130. ábra - Item.vue

Reszponzív design

A frontend kódbázis gondosan ügyel a mobilbarát megjelenítésre. Példa egy reszponzív componensre:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

131. ábra - Egy példa a reszponzivitásra

A reszponzivitással tudjuk elérni, hogy az oldal minden pixel méreten megfelelően jelenjen meg. pl. mobil és asztali gép között más lesz a pixelek mennyisége egyértelműen, hisz a telefonnak kisebb a kijelzője, ebből fakadóan kevesebb dolog jelenik meg rajta, így módosítani kell a megjelenésén.

Felhasználói élmény optimalizálás

* Lazy loading képeknél
* Kondicionális komponens renderelés
* Animációk és átmenetek
* Hibaüzenetek és visszajelzések
* Állapotok visszajelzése (loading, success, error)

## 3.6 API Dokumentáció

### 3.6.1 Végpontok

Könyvek API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/books | Összes könyv lekérdezése | Publikus |
| GET | /api/books/{id} | Egy könyv részleteinek lekérdezése | Publikus |
| POST | /api/books | Új könyv létrehozása | Admin |
| PUT | /api/books/{id} | Könyv adatainak módosítása | Admin |
| DELETE | /api/books/{id} | Könyv törlése | Admin |
| GET | /api/books/most-viewed | Legnézettebb 5 könyv lekérdezése | Publikus |
| GET | /api/books/oldest-books | Legrégebbi 5 könyv lekérdezése | Publikus |
| GET | /api/books/newest-books | Legújabb 5 könyv lekérdezése | Publikus |
| GET | /api/books/most-liked | Legjobb értékeléssel rendelkező 5 könyv | Publikus |
| GET | /api/books/most-commented | Legtöbb kommenttel rendelkező 5 könyv | Publikus |
| GET | /api/books/search/{search} | Könyvek keresése cím vagy szerző alapján | Publikus |
| POST | /api/books/{id}/view | Könyv megtekintésének növelése | Publikus |
| GET | /api/books/author/{author} | Könyvek szűrése szerző alapján | Publikus |

7. táblázat - Könyvek elérési útjai

Felhasználók API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| POST | /api/register | Felhasználó regisztrálása | Publikus |
| POST | /api/login | Bejelentkezés | Publikus |
| POST | /api/logout | Kijelentkezés | Bejelentkezett felhasználó |
| GET | /api/user | Bejelentkezett felhasználó adatainak lekérdezése | Bejelentkezett felhasználó |
| GET | /api/users | Összes felhasználó lekérdezése | Admin |
| PUT | /api/users/{id}/role | Felhasználói szerepkör módosítása | Admin |
| PUT | /api/users/{id} | Felhasználói adatok módosítása | Tulajdonos/Admin |

8. táblázat -Felhasználók elérési útjai

Kategóriák API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/categories | Összes kategória lekérdezése | Publikus |
| GET | /api/categories/{id} | Egy kategória részleteinek lekérdezése | Publikus |
| POST | /api/categories | Új kategória létrehozása | Admin |
| PUT | /api/categories/{id} | Kategória módosítása | Admin |
| DELETE | /api/categories/{id} | Kategória törlése | Admin |

9. táblázat - Kategóriák elérési útjai

Rendelések API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/orders | Felhasználó rendeléseinek lekérdezése | Bejelentkezett felhasználó |
| POST | /api/orders | Új rendelés létrehozása | Email-ellenőrzött felhasználó |
| DELETE | /api/orders/{id} | Rendelés törlése | Admin |

10. táblázat - Rendelések elérési útjai

Értékelések API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/rating | Összes értékelés lekérdezése | Publikus |
| POST | /api/rating | Új értékelés létrehozása | Bejelentkezett felhasználó |
| GET | /api/rating/{id} | Egy értékelés részleteinek lekérdezése | Publikus |
| PUT | /api/rating/{id} | Értékelés módosítása | Tulajdonos/Admin |
| DELETE | /api/rating/{id} | Értékelés törlése | Tulajdonos/Admin |

11. táblázat - Értékelések elérési útjai

Hozzászólások API

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/comments | Összes hozzászólás lekérdezése | Publikus |
| POST | /api/comments | Új hozzászólás létrehozása | Bejelentkezett felhasználó |
| GET | /api/comments/{id} | Egy hozzászólás részleteinek lekérdezése | Publikus |
| PUT | /api/comments/{id} | Hozzászólás módosítása | Tulajdonos/Admin |
| DELETE | /api/comments/{id} | Hozzászólás törlése | Tulajdonos/Admin |

12. táblázat - Hozzászólások elérési útjai

Email verifikáció

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/email/verify | Verifikációs értesítés lekérése | Bejelentkezett felhasználó |
| GET | /api/email/verify/{id}/{hash} | Email cím ellenőrzése | Specifikus felhasználó |
| POST | /api/email/verification-notification | Verifikációs email újraküldése | Bejelentkezett felhasználó |

13. táblázat – Email verifikációs elérési útjai

Jelszó visszaállítás

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Metódus | Végpont | Leírás | Jogosultság |
| GET | /api/forget-password | Elfelejtett jelszó oldal információk | Publikus |
| POST | /api/forget-password | Jelszó visszaállítási link küldése | Publikus |
| GET | /api/reset-password/{token} | Jelszó visszaállítási oldal adatok | Specifikus token |

14. táblázat – Jelszó visszaállítási elérési útjai

3.6.2 Paraméteres kérések lehetőségei

Könyvek API

POST /api/books (Új könyv létrehozása)

{

"title": "A könyv címe",

"author": "Szerző neve",

"isbn": "1234567890123",

"publish\_date": "2024-03-15",

"price": 3990,

"describe": "Könyv leírása",

"cover\_image": "url/to/image.jpg",

"category\_id": 1

}

Validációs szabályok:

* title: kötelező, string, max 60 karakter, egyedi
* author: kötelező, string, max 255 karakter
* isbn: kötelező, string, egyedi, pontosan 13 karakter
* publish\_date: kötelező, dátum (YYYY-MM-DD), nem későbbi mint a mai nap
* price: kötelező, numerikus, min 0
* describe: kötelező, string
* cover\_image: kötelező, string, max 255 karakter
* category\_id: kötelező, létező kategória ID

PUT /api/books/{id} (Könyv módosítása) Ugyanazok a paraméterek, mint a létrehozásnál.

Felhasználók API

POST /api/register (Regisztráció)

{

"Username": "felhasznalo",

"email": "[user@example.com](mailto:user@example.com)",

"password": "password123",

"password\_confirmation": "password123",

"FullName": "Teljes Név",

"PhoneNumber": "+36301234567"

}

Validációs szabályok:

* Username: kötelező, max 255 karakter
* email: kötelező, érvényes email formátum, egyedi
* password: kötelező, string, min 8 karakter, confirmed
* FullName: kötelező, max 255 karakter
* PhoneNumber: kötelező, max 13 karakter

POST /api/login (Bejelentkezés)

{

"email": "[user@example.com](mailto:user@example.com)",

"password": "password123"

}

Validációs szabályok:

* email: kötelező, érvényes email formátum, létező felhasználó
* password: kötelező

PUT /api/users/{id} (Felhasználói adatok módosítása)

{

"username": "újfelhasznalonév",

"email": "[user@example.com](mailto:user@example.com)",

"PhoneNumber": "+36301234567"

}

Validációs szabályok:

* username: kötelező, string, max 255 karakter
* email: kötelező, érvényes email formátum, egyedi (kivéve az aktuális felhasználó)
* PhoneNumber: kötelező, string, max 20 karakter

Rendelések API

POST /api/orders (Rendelés létrehozása)

{

"items": [

{

"book\_id": 1,

"quantity": 2,

"price": 3990,

"books\_total\_price": 7980

}

],

"shipping": {

"name": "Vásárló Neve",

"email": "[user@example.com](mailto:user@example.com)",

"phone": "+36301234567",

"address": "1234 Budapest, Példa utca 1.",

"method": "home",

"fee": 990

},

"payment\_method": "bank\_transfer",

"total\_amount": 8970,

"reference\_id": "REF123456"

}

Validációs szabályok:

* items: kötelező, tömb
* items.\*.book\_id: kötelező, létező könyv ID
* items.\*.quantity: kötelező, egész szám, min 1
* items.\*.price: kötelező, numerikus, min 0
* items.\*.books\_total\_price: kötelező, numerikus, min 0
* shipping.name: kötelező, string, min 3
* shipping.email: kötelező, érvényes email
* shipping.phone: kötelező, string
* shipping.address: kötelező, string
* shipping.method: kötelező, értéke 'home' vagy 'pickup'
* shipping.fee: kötelező, numerikus
* payment\_method: kötelező, string
* total\_amount: kötelező, numerikus
* reference\_id: opcionális, string

Értékelések API

POST /api/rating (Értékelés létrehozása)

{

"user\_id": 1,

"book\_id": 2,

"rating": 5

}

Validációs szabályok:

* user\_id: kötelező, létező felhasználó ID
* book\_id: kötelező, létező könyv ID
* rating: kötelező, egész szám, 1-5 közötti érték

Hozzászólások API

POST /api/comments (Hozzászólás létrehozása)

{

"comment": "Ez egy nagyszerű könyv!",

"book\_id": 2,

"user\_id": 1

}

Validációs szabályok:

* comment: kötelező, string, max 255 karakter
* book\_id: kötelező, létező könyv ID
* user\_id: kötelező, létező felhasználó ID

3.6.3 Válasz

Könyvek API

Sikeres válasz (GET /api/books):

{

"data": [

{

"id": 1,

"title": "Könyv címe",

"author": "Szerző neve",

"isbn": "1234567890123",

"publish\_date": "2024-03-15",

"price": 3990,

"describe": "Könyv leírása",

"cover\_image": "url/to/image.jpg",

"categories": [

“id”:1,

“CategoryName”:” Modern Irodalom”

]

}

],

"success": true,

"message": " "

}

Sikeres válasz (GET /api/books/{id}):

{

"id": 1,

"title": "Könyv címe",

"author": "Szerző neve",

"isbn": "1234567890123",

"publish\_date": "2024-03-15",

"price": 3990,

"describe": "Könyv leírása",

"cover\_image": "url/to/image.jpg",

"categories": [

“id”:1,

“CategoryName”:” Modern Irodalom”

]

}

Hiba válasz:

{

"message": "Book not found"

}

Felhasználók API

Sikeres regisztrációs válasz:

{

"user": {

"id": 1,

"Username": "felhasznalo",

"email": "[user@example.com](mailto:user@example.com)",

"FullName": "Teljes Név",

"PhoneNumber": "+36301234567",

"role": "user"

},

"token": "1|iKnGQThLDJfzAJAu1GvRv0WsrJC8JYqr3TKi6QKj",

"success": true,

"message": "Regisztráció sikeres! Kérjük, ellenőrizze az e-mail fiókját a megerősítő linkért."

}

Sikeres bejelentkezési válasz:

{

"user": {

"id": 1,

"Username": "felhasznalo",

"email": "[user@example.com](mailto:user@example.com)",

"FullName": "Teljes Név",

"PhoneNumber": "+36301234567",

"role": "user"

},

"token": "1|iKnGQThLDJfzAJAu1GvRv0WsrJC8JYqr3TKi6QKj"

Hiba válasz bejelentkezéskor:

{

"message": "Rossz email vagy jelszó"

}

Rendelések API

Sikeres rendelés válasz:

{

"success": true,

"order\_id": "REF123456",

"message": "A rendelés sikeresen feldolgozva",

"order": {

"id": 1,

"user\_id": 1,

"OrderDate": "2024-03-31",

"Status": "Függőben",

"TotalAmount": 8970,

"reference\_id": "REF123456",

"created\_at": "2024-03-31T12:00:00.000000Z",

"updated\_at": "2024-03-31T12:00:00.000000Z",

"order\_items": [...],

"payments": [...],

"shipping": {...}

}

}

Hiba válasz rendelésnél:

{

"error": "Hiba történt a rendelés feldolgozásakor: [hibaüzenet]"

}

Kategóriák API

Sikeres válasz (GET /api/categories):

{

"data": [

{

"id": 1,

"CategoryName": "Fantasy",

"created\_at": "2024-02-15T10:20:30.000000Z",

"updated\_at": "2024-02-15T10:20:30.000000Z"

},

{

"id": 2,

"CategoryName": "Thriller",

"created\_at": "2024-02-15T10:20:30.000000Z",

"updated\_at": "2024-02-15T10:20:30.000000Z"

}

]

}

Értékelések API

Sikeres válasz (GET /api/rating):

{

"data": [

{

"id": 1,

"book\_id": 2,

"user\_id": 1,

"rating": 5,

"created\_at": "2024-03-20T09:15:30.000000Z",

"updated\_at": "2024-03-20T09:15:30.000000Z",

"user": {

"id": 1,

"Username": "felhasznalo"

},

"book": {

"id": 2,

"title": "Könyv címe"

}

}

]

}

Sikeres értékelés létrehozása (POST /api/rating):

{

"message": "Rating created successfully",

"rating": {

"user\_id": 1,

"book\_id": 2,

"rating": 5,

"updated\_at": "2024-03-31T15:10:22.000000Z",

"created\_at": "2024-03-31T15:10:22.000000Z",

"id": 3

}

}

Hozzászólások API

Sikeres válasz (GET /api/comments):

{

"data": [

{

"id": 1,

"comment": "Ez egy nagyszerű könyv!",

"book\_id": 2,

"user\_id": 1,

"created\_at": "2024-03-25T11:30:45.000000Z",

"updated\_at": "2024-03-25T11:30:45.000000Z",

"user": {

"id": 1,

"Username": "felhasznalo"

}

}

]

}

Sikeres hozzászólás létrehozása (POST /api/comments):

{

"message": "Comment added successfully",

"comment": {

"comment": "Ez egy nagyszerű könyv!",

"book\_id": 2,

"user\_id": 1,

"updated\_at": "2024-03-31T16:05:10.000000Z",

"created\_at": "2024-03-31T16:05:10.000000Z",

"id": 2

}

}

Email verifikáció

Email verifikációs értesítés (GET /api/email/verify):

{

"message": "Email megerősítési link elküldve!"

}

Sikeres email újraküldés (POST /api/email/verification-notification):

{

"message": "Email megerősítés elküldve!"

}

Hiba email újraküldésnél:

{

"message": "Felhasználó nem található"

}

Jelszó visszaállítás

Elfelejtett jelszó információ (GET /api/forget-password):

{

"title": "Elfelejtett jelszó",

"message": "Kérjük, adja meg e-mail-címét, hogy a jelszó módosításához szükséges információkat elküldhessük Önnek."

}

Sikeres jelszó visszaállítási link küldés (POST /api/forget-password):

{

"status": "success",

"message": "We have emailed your password reset link!"

}

Jelszó visszaállítási token információ (GET /api/reset-password/{token}):

{

"token": "abcdef1234567890",

"email": "[user@example.com](mailto:user@example.com)"

}

Sikeres jelszó visszaállítás (POST /api/reset-password):

{

"status": "success",

"message": "Your password has been reset!"

}

Hiba jelszó visszaállításkor:

{

"status": "error",

"message": "This password reset token is invalid."

}

HTTP státuszkódok

* 200 OK: Sikeres GET, PUT, DELETE művelet
* 201 Created: Sikeres létrehozás (POST)
* 204 No Content: Sikeres törlés, válasz nélkül
* 400 Bad Request: Érvénytelen kérés
* 401 Unauthorized: Nincs bejelentkezve
* 403 Forbidden: Jogosultság hiánya
* 404 Not Found: Nem található erőforrás
* 422 Unprocessable Entity: Validációs hiba
* 500 Internal Server Error: Szerver hiba

### 3.7 Teszt dokumentáció

3.7.1 Bevezetés

Ez a tesztdokumentáció részletesen bemutatja a Scriptum Könyvesbolt webalkalmazás tesztelési folyamatát, a tesztesetek típusait és a teszteredményeket. A dokumentáció célja, hogy átfogó képet adjon a rendszer minőségbiztosítási eljárásairól.

3.7.2 A rendszer áttekintése

A Scriptum Könyvesbolt egy online könyváruház, amely lehetővé teszi a felhasználók számára könyvek böngészését, keresését, értékelését, kosárba helyezését és megvásárlását. Az alkalmazás frontend és backend komponensekből áll:

* Frontend: Vue.js 3 alapú single-page application
* Backend: Laravel 11 RESTful API

3.7.3 A tesztdokumentáció hatóköre

Ez a dokumentáció kiterjed:

* Frontend komponensek egység- és integrációs tesztjeire
* Backend API végpontok funkcionalitási tesztjeire
* Adatbázis modellek egységtesztjeire
* Felhasználói jogosultságok tesztjeire

3.7.4 Tesztelési megközelítés

Tesztstratégia

* Frontend: Komponens-alapú tesztelés Vue Test Utils és Vitest használatával
* Backend: Feature és Unit tesztek PHPUnit segítségével
* Tesztkörnyezet: Külön fejlesztői, tesztelési és éles környezetek

Automatizációs szint

* Frontend automatizáció: 85%
* Backend automatizáció: 95%
* Regressziós tesztek automatizációja: 90%

3.7.5 Tesztkörnyezet

Frontend tesztkörnyezet

* Node verzió: v18.16.0
* Vue verzió: 3.2.47
* Tesztelési keretrendszer: Vitest 0.30.1
* Komponens tesztelő: @vue/test-utils 2.3.0
* Böngésző emuláció: JSDOM 21.1.1

Backend tesztkörnyezet

* PHP verzió: 8.2.0
* Laravel verzió: 11.0
* Testing Framework: PHPUnit 11.0.19
* Adatbázis: MySQL 8.0 (tesztelési példány)
* Autentikáció: Laravel Sanctum

3.7.6 Frontend komponens tesztek

Cart.test.ts

Tesztek összefoglalása

* A kosár komponens terméklista megjelenítés tesztelése
* Üres kosár esetén megfelelő üzenet megjelenítésének ellenőrzése
* Tétel hozzáadás, módosítás és törlés funkciók tesztelése
* Kosár összesítő helyes számításának ellenőrzése
* Rendelés folyamat kezdeményezésének tesztelése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| CART-001 | Üres kosár esetén megfelelő üzenet megjelenítése | "A kosár üres" üzenet látható | Sikeres |
| CART-002 | Termékek helyes megjelenítése a kosárban | Minden kosárelem címe, ára és mennyisége látható | Sikeres |
| CART-003 | Mennyiség módosítása | Az ár és a végösszeg helyesen frissül | Sikeres |
| CART-004 | Tétel törlése a kosárból | A tétel eltűnik és a végösszeg frissül | Sikeres |
| CART-005 | "Rendelés" gomb csak bejelentkezett felhasználóknak aktív | Nem bejelentkezett felhasználó esetén inaktív gomb | Sikeres |

15. táblázat - Tesztesetek részletezése

Footer.test.ts

Tesztek összefoglalása

* Lábléc megfelelő megjelenítésének ellenőrzése
* Kapcsolati linkek és copyright információk tesztelése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| FOOTER-001 | A lábléc megjelenése | A lábléc látható és tartalmazza a cég nevét | Sikeres |
| FOOTER-002 | Copyright szöveg ellenőrzése | A copyright felirat tartalmazza az aktuális évet | Sikeres |
| FOOTER-003 | Kapcsolati linkek működése | Minden link megfelelő URL-re mutat | Sikeres |

16. táblázat - Tesztesetek részletezése

Item.test.ts

Tesztek összefoglalása

* Könyv kártya helyes megjelenítésének ellenőrzése
* Ár formázás ellenőrzése
* "Kosárba" gomb működésének tesztelése
* "Részletek" gomb és navigáció tesztelése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| ITEM-001 | Könyv adatok helyes megjelenítése | Cím, szerző, ár látható | Sikeres |
| ITEM-002 | Borítókép helyes betöltése | A kép alt szövege és src attribútuma helyes | Sikeres |
| ITEM-003 | "Kosárba" gomb kattintás | add-to-cart esemény kiváltása helyes adatokkal | Sikeres |
| ITEM-004 | "Részletek" gomb kattintás | Navigálás a könyv részletek oldalra | Sikeres |

17. táblázat - Tesztesetek részletezése

AdminBooks.test.ts

Tesztek összefoglalása

* Admin jogosultság ellenőrzése
* Könyvek listázásának tesztelése admin felületen
* Új könyv létrehozás űrlap validálása
* Könyv szerkesztés és törlés funkciók tesztelése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| ADMIN-BOOKS-001 | Nem admin felhasználó átirányítása | Átirányítás a főoldalra nem admin felhasználó esetén | Sikeres |
| ADMIN-BOOKS-002 | Könyvek listázása | Az összes könyv megjelenik táblázatos formában | Sikeres |
| ADMIN-BOOKS-003 | Új könyv űrlap validáció | Hibaüzenetek megjelennek hiányzó adatok esetén | Sikeres |
| ADMIN-BOOKS-004 | Könyv szerkesztése | A módosítások sikeresen mentésre kerülnek | Sikertelen |
| ADMIN-BOOKS-005 | Könyv törlése | A könyv sikeresen törlődik megerősítés után | Sikeres |

18. táblázat - Tesztesetek részletezése

További frontend tesztek

A dokumentáció további frontend teszteket is tartalmaz, melyek hasonló struktúrában részletezik a Rating, NotFound, ReturnButton, ItemDetails, Comment és CartButton komponensek teszteseteit.

3.7.7 Backend API tesztek

BookTest.php

Tesztek összefoglalása

* Könyvek lekérdezési API végpontjainak tesztelése
* Könyv létrehozási, módosítási és törlési műveletekhez tartozó jogosultságok ellenőrzése
* Speciális könyv lekérdezések tesztelése (legnépszerűbb, legújabb stb.)

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| API-BOOK-001 | Összes könyv lekérdezése | 200 OK válasz és könyvek listája | Sikeres |
| API-BOOK-002 | Egy könyv lekérdezése | 200 OK válasz és könyv adatok | Sikeres |
| API-BOOK-003 | Nem létező könyv lekérdezése | 404 Not Found válasz | Sikeres |
| API-BOOK-004 | Könyv létrehozása admin jogosultsággal | 201 Created válasz és az új könyv adatai | Sikeres |
| API-BOOK-005 | Könyv létrehozása jogosultság nélkül | 403 Forbidden válasz | Sikeres |
| API-BOOK-006 | Könyv módosítása | 200 OK válasz és frissített adatok | Sikeres |
| API-BOOK-007 | Könyv törlése | 204 No Content válasz | Sikeres |
| API-BOOK-008 | Legnézettebb könyvek | 200 OK válasz és rendezett lista | Sikeres |
| API-BOOK-009 | Legújabb könyvek | 200 OK és dátum szerint rendezett lista | Sikeres |
| API-BOOK-010 | Szerző szerinti keresés | 200 OK és a megfelelő könyvek | Sikeres |

19. táblázat - Tesztesetek részletezése

CategoryTest.php

Tesztek összefoglalása

* Kategóriák lekérdezési API végpontjainak tesztelése
* Kategória létrehozási, módosítási és törlési műveletek tesztelése
* Kategóriához tartozó könyvek lekérdezésének tesztelése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| API-CAT-001 | Összes kategória lekérdezése | 200 OK válasz és kategóriák listája | Sikeres |
| API-CAT-002 | Egy kategória lekérdezése | 200 OK válasz és kategória adatok | Sikeres |
| API-CAT-003 | Kategóriához tartozó könyvek | 200 OK válasz és könyvek listája | Sikeres |
| API-CAT-004 | Kategória létrehozása admin jogosultsággal | 201 Created válasz | Sikeres |
| API-CAT-005 | Kategória módosítása | 200 OK válasz és frissített adatok | Sikeres |
| API-CAT-006 | Kategória törlése | 204 No Content válasz | Sikeres |

20. táblázat - Tesztesetek részletezése

### 3.7.8 További API tesztek

A dokumentáció további API teszteket is tartalmaz, melyek hasonló struktúrában részletezik a Comment, Orders, Rating és User API végpontok teszteseteit.

3.7.9 Adatbázis modell unit tesztek

BookModelTest.php

Tesztek összefoglalása

* Book modell attribútumainak ellenőrzése
* Book-Category kapcsolat tesztelése
* Book-Rating és Book-Comment kapcsolatok ellenőrzése

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt azonosító | Leírás | Elvárt eredmény | Státusz |
| UNIT-BOOK-001 | Book modell létrehozhatósága | Az objektum sikeresen létrejön | Sikeres |
| UNIT-BOOK-002 | Fillable attribútumok ellenőrzése | Minden mező beállítható | Sikeres |
| UNIT-BOOK-003 | Könyv-kategória kapcsolat | A categories() metódus belongsToMany kapcsolatot ad vissza | Sikeres |
| UNIT-BOOK-004 | Könyv-értékelés kapcsolat | A ratings() metódus hasMany kapcsolatot ad vissza | Sikeres |
| UNIT-BOOK-005 | Könyv-hozzászólás kapcsolat | A comments() metódus hasMany kapcsolatot ad vissza | Sikeres |
| UNIT-BOOK-006 | Legnézettebbek scope | A mostViewed() scope helyes lekérdezést ad | Sikeres |

21. táblázat - Tesztesetek részletezése

3.7.10 További modell tesztek

A dokumentáció további modell teszteket is tartalmaz a Category, Comment, Order, Payment, PaymentMethod, Rating, Shipping és User modellek részletes tesztelésével.

3.7.11 Tesztfuttatási eljárások

Frontend tesztek futtatása

# Inicializálás

cd Vizsga/Frontend

npm install

# Összes teszt futtatása

npm run test

# Egyedi komponens tesztelése

npm run test -- tests/components/Cart.test.ts

# Lefedettségi riport generálása

npm run test:coverage

3.7.12 Backend tesztek futtatása

# Inicializálás

cd Vizsga/Backend/BookShop

composer install

php artisan config:cache --env=testing

php artisan migrate:fresh --env=testing

# Összes teszt futtatása

php artisan test

# Feature tesztek futtatása

php artisan test --testsuite=Feature

# Unit tesztek futtatása

php artisan test --testsuite=Unit

# Egyedi tesztfájl futtatása

php artisan test tests/Feature/BookTest.php

3.7.13 Teszteredmények összesítése

Frontend tesztek eredménye

* Összes teszt: 78
* Sikeres tesztek: 75
* Sikertelen tesztek: 3
* Skippelt tesztek: 0
* Kódlefedettség: 87%

Frontend teszt hibák összefoglalása

|  |  |  |  |
| --- | --- | --- | --- |
| Hiba azonosító | Érintett teszt | Hiba leírása | Prioritás |
| FE-BUG-001 | ADMIN-BOOKS-004 | Könyv adatok nem frissülnek szerkesztés után | Magas |
| FE-BUG-002 | CART-003 | A kosár nem frissíti helyesen az árat mennyiség csökkentésekor | Közepes |
| FE-BUG-003 | RATING-002 | Az értékelési csillagok nem kattinthatók mobilnézetben | Alacsony |

22. táblázat - Tesztesetek részletezése

3.7.14 Backend tesztek eredménye

Utolsó futtatás időpontja: 2025.03.31

* Összes teszt: 124
* Sikeres tesztek: 120
* Sikertelen tesztek: 4
* Skippelt tesztek: 0
* Kódlefedettség: 92%

Backend teszt hibák összefoglalása

|  |  |  |  |
| --- | --- | --- | --- |
| Hiba azonosító | Érintett teszt | Hiba leírása | Prioritás |
| BE-BUG-001 | API-BOOK-008 | A legnézettebb könyvek nem a helyes sorrendben jelennek meg | Közepes |
| BE-BUG-002 | API-ORDER-003 | A rendelés létrehozása nem ellenőrzi a készletet | Magas |
| BE-BUG-003 | API-USER-004 | A jelszó-visszaállítás nem küld emailt | Magas |

23. táblázat - Tesztesetek részletezése

### 4. További fejlesztési lehetőségek

4.1. Technológiai fejlesztések:

* A közeljövőben teljes és életszerű fizetési rendszer implementálása Paypal vagy Stripe-al.

4.2. Felhasználói élmény fejlesztések:

* Az adatok dedikált szerverre vagy CDM-re való átrakása, a jobb felhasználói élményért pl. Gyorsabb adat betöltés az oldalon.
* Email küldési rendszer felgyorsítása a regisztráció során.
* Firefox böngészőben a kosár oldal megjelenítése hibás lehet, ez javítás alatt áll.

### 5. Összefoglalás

A jelen tesztdokumentáció célja a Scriptum Könyvesbolt webalkalmazás minőségbiztosítási folyamatának és eredményeinek átfogó bemutatása. A frontend Vue.js és backend (Laravel API) komponensek tesztelése során összesen 202 automatizált tesztet futtattunk, amelyek kiemelkedő, 90% körüli sikerességi arányt mutattak. A tesztfolyamat 87%-os frontend és 92%-os backend kódlefedettséget ért el, ami megfelel az iparági standardoknak. A felhasználói felület komponensei és az API végpontok tesztelése egyaránt részét képezte a vizsgálatnak, különös tekintettel a könyv keresési, kosárkezelési és rendelési funkcionalitásokra.

### 6. Források

1. Laravel (2024. márc. 12.). Database: Migrations. <https://laravel.com/docs/11.x/migrations>. Utolsó látogatás: 2025.04.12.
2. Laravel (2024. márc. 12.). Database: Seeders. <https://laravel.com/docs/11.x/seeding>. Utolsó látogatás: 2025.04.12.
3. Laravel (2024. márc. 12.). Digging Deeper: Mail. <https://laravel.com/docs/11.x/mail>. Utolsó látogatás: 2025.04.12.
4. Laravel (2024. márc. 12.). Eloquent ORM: API Resource. <https://laravel.com/docs/11.x/eloquent-resources>. Utolsó látogatás: 2025.04.12.
5. Laravel (2024. márc. 12.). Security: Authentication. <https://laravel.com/docs/11.x/authentication>. Utolsó látogatás: 2025.04.12.
6. Laravel (2024. márc. 12.). Security: Authorization. <https://laravel.com/docs/11.x/authorization>. Utolsó látogatás: 2025.04.12.
7. Laravel (2024. márc. 12.). Security: Email Verification. <https://laravel.com/docs/11.x/verification>. Utolsó látogatás: 2025.04.12.
8. Laravel (2024. márc. 12.). Security: Password Reset. <https://laravel.com/docs/11.x/passwords>. Utolsó látogatás: 2025.04.12.
9. Laravel (2024. márc. 12.). Testing. <https://laravel.com/docs/11.x/testing>. Utolsó látogatás: 2025.04.12.
10. Laravel (2024. márc. 12.). The Basics: Middleware. <https://laravel.com/docs/11.x/middleware>. Utolsó látogatás: 2025.04.12.
11. Laravel (2024. márc. 12.). The Basics: Requests. <https://laravel.com/docs/11.x/requests>. Utolsó látogatás: 2025.04.12.
12. Vee-validate (2025. jan.). Vee-validate. <https://vee-validate.logaretm.com/v4/>. Utolsó látogatás: 2025.04.12.
13. Vuejs.org (2021. febr. 2.). Vuex. <https://vuex.vuejs.org>. Utolsó látogatás: 2025.04.12.
14. Vuejs.org (2024. nov. 25.). Vue Router. <https://router.vuejs.org/guide/>. Utolsó látogatás: 2025.04.12.