[1. Bevezetés 2](#_Toc133401765)

[1.1. A probléma és a megoldás 2](#_Toc133401766)

[1.2. Az ötletelés 3](#_Toc133401767)

[1.3. A jövő 3](#_Toc133401768)

[1.4. A csapatmunka 4](#_Toc133401769)

[1.5. A publikálás 6](#_Toc133401770)

[2. A program 7](#_Toc133401771)

[2.1. Technikai részletek 7](#_Toc133401772)

[2.2. Az adatbázis felépítése 8](#_Toc133401773)

[2.2.1. Táblák 8](#_Toc133401774)

[2.3. Felhasználói fiók 10](#_Toc133401775)

[2.3.1. Regisztráció 10](#_Toc133401776)

[2.4. Bejelentkezés 11](#_Toc133401777)

[2.5. Felhasználók számára elérhető funkciók 11](#_Toc133401778)

[2.6. Autentikáció a backenden 11](#_Toc133401779)

[2.7. API dokumentáció 12](#_Toc133401780)

[2.7.1. authentication 12](#_Toc133401781)

[2.7.2. Users 13](#_Toc133401782)

[2.7.3. Orders 14](#_Toc133401783)

[2.7.4. Products 15](#_Toc133401784)

[2.7.5. Orderdetails 17](#_Toc133401785)

[2.7.6. Payments 18](#_Toc133401786)

[2.8. Frontend dokumentáció 20](#_Toc133401787)

[2.8.1. Felépítése 20](#_Toc133401788)

[3. Tesztek 21](#_Toc133401789)

[3.1. Backend tesztek 21](#_Toc133401790)

1. Bevezetés
   1. A probléma és a megoldás

A shopking egy online webáruház ahol a ujjonnakn megjelnő könveket és régi kalsszikusokat egyaránt lehetőségunk van meg vásárolni. A webshopunk egyaránt lehetőséget fog nyujtani azoknak a fuggetlen íroknak akikknek nincs platformjuk árusitani a termékuket hogy rajtunk keresztül monetizálhasák a könyveiket.

A weboldal bárhonnan elérhető internetkapcsolat segítségével akár mobilról, tabletről

vagy számítógépről is. A regisztráció, a hirdetés, a böngészés, a jelentkezés mind online

felületen keresztül történik, ezért nem igényel semmiféle eszközön letöltést vagy telepítést.

* 1. Az ötletelés

ebben a fejezetben az ötletelés fázisban megfogalmazott terveinket részletezzük. A alkalmazást a letisztultság volt az elsődleges a dizájnolás közben és hogy a modern igényeknek megfelejen. Biztositjuk a könnyü és egyszerü bejelentkezést és regisztráciot, és a adminoknak lehetősége legyen a rendelt termékek gyors és egyértelmű kezelésére és ezek modositására, törlésére, hogy modositani tudják a adminok a rendelés státuszát és a ugyfeleknek meg lehetőséguk legyen lemondani vagy modositani a szálitási címet

* 1. A jövő

A project előzetes tervezésekor számalan ötlettel álltunk elő. A végleges tervezéskor azonban mérlegelnünk kellett, mi az, ami a megadott határidőn belül kivitelezhető. Annak érdekében, hogy az alkalmazás leadásakor használható legyen, bizonyos funkciók megvalositásába viszont nem kezdtünk bele. Ezeket, a jelenlegi vezzioból kimaradtak ötleteink azonban a jövőben mindenképpen szeretnénk véghez vinni. Fontosnak tartjuk megemliteni ezeket, hiszen megmutatják ambícóinkat és a fejlődés lehetőségeit, irányait a project szempontjából. Az egyik legjelentősebb tervünk

az online fizetés lehetősége. A jelentkezők bankkártyával előre kifizethetnék a rendelt termékeket. Bár a weboldal teljes mértékben reszponzív, elképzelhetőnek tartjuk egy esetleges mobilalkalmazás fejlesztését is. Ennek érdekében alakítottuk úgy a backendet, hogy az a frontendtől független legyen, arra az esetre, ha egy másik típusú alkalmazást is hozzá szeretnénk kapcsolni.

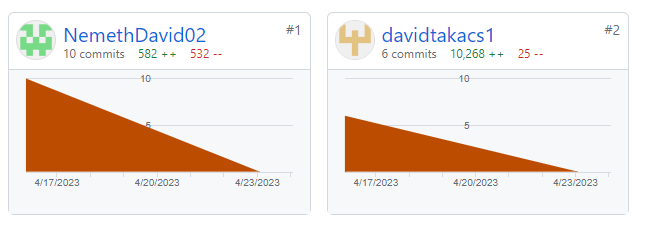
* 1. A csapatmunka

A programozás elkézdése előtt fontos volt, hogy megfelelően osszuk fel a feladatokat. mivel csak ketten voltunk annak megfelően kellet felosztauk ki mit csináljon ezért ugy gondoltuk az lenne a legjobb ha felosztanánk a feladatott bacend és frontendre igy gyorsabban tudtunk halladni, viszont nem kizárva azt hogy segitsünk egymásnak. A csapatmunka megkönnyítése végett és a verziókövetéshez a GitHubot használtuk. Ezen a felületen nyomon követhető, ki, melyik feladatot végezte.

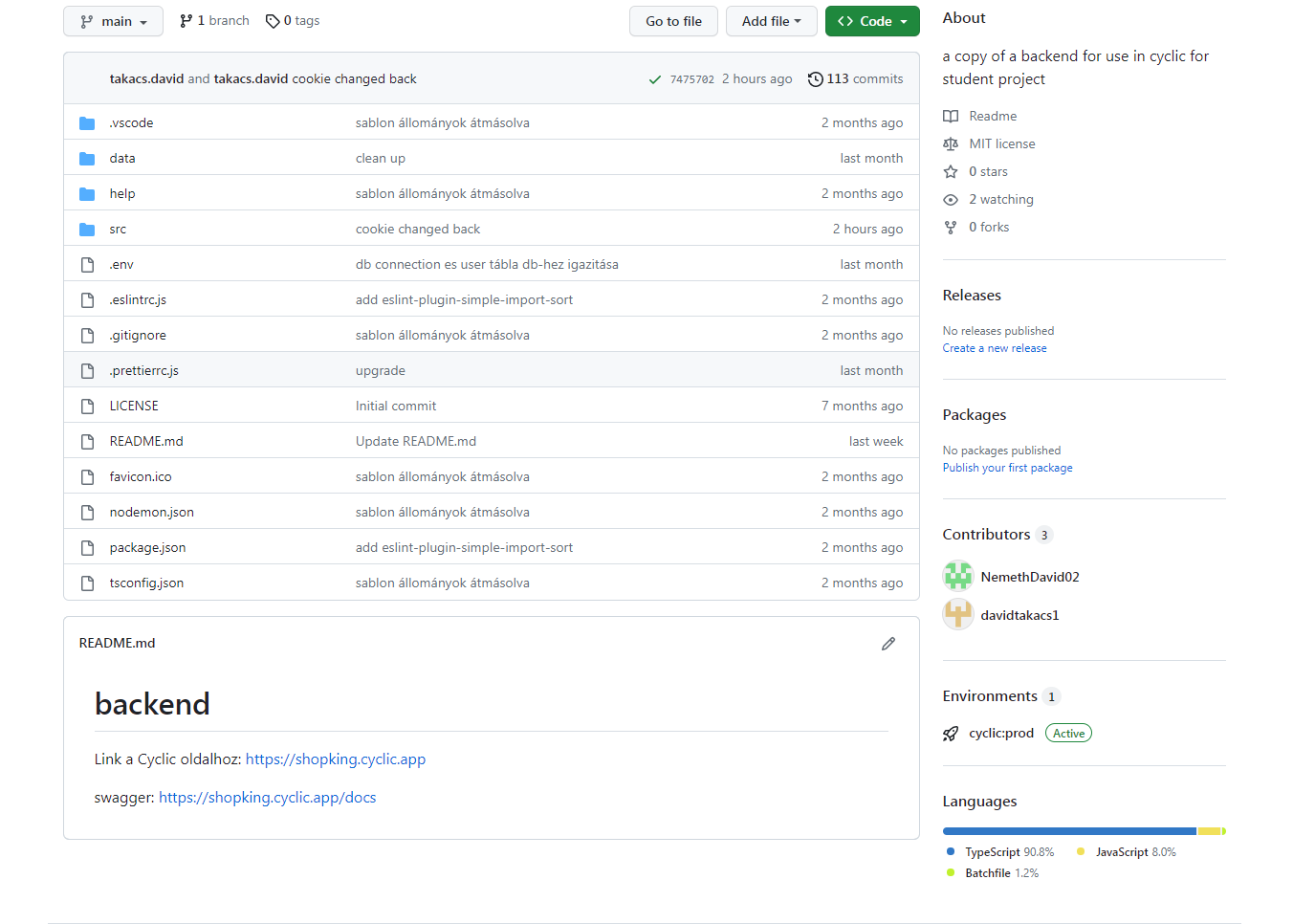
A GitHub repository-nkat két részre osztottuk ugyanis a hostunk nem támogatta hogy organisonsbol inditsunk projectett

A frontent megtalálható: https://github.com/davidtakacs1/BookStoreFrontend

A backend megtalálható: https://github.com/NemethDavid02/backend-cyclic



Itt a fontend látható ahol a megjelenést, dizajnt és a vue codot külön csináltuk.

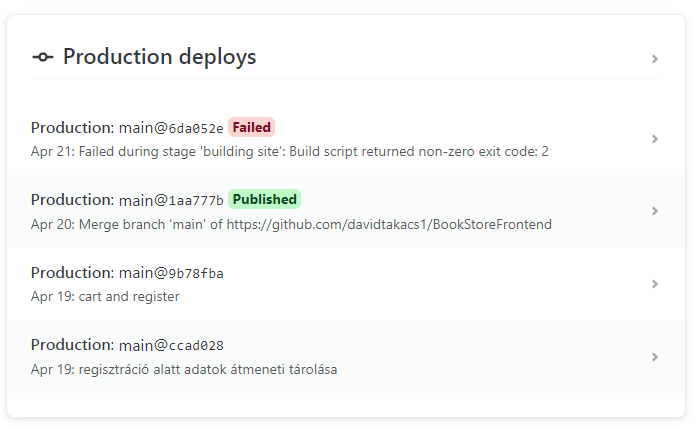
A csapatunk komunikácioját mikor személyesen nem tudtuk megoldani szünetben vagy hétvégén akkor Google Meet és mesengert használtunk , így valósult meg ilyenkor a csapatmunka. Minden feladat során külön branch-ben dolgoztunk, melyeket később, a helyes működés biztosítása után, Pull request segítségével töltöttünk fel a Master ágba. Ezzel elkerültük a már működő programkód hibással való felülírását, valamint így egyszerűen tudtunk egy időben dolgozni a programo

Erről a képről leolvasható, hogy milyen programnyelveket milyen arányban használtunk a fejlesztéskor. Az is látható, hogy a kép készítésekor 113 commitot hajtottunk végre. A mappastruktúra is megjelenik ezen az oldalon.

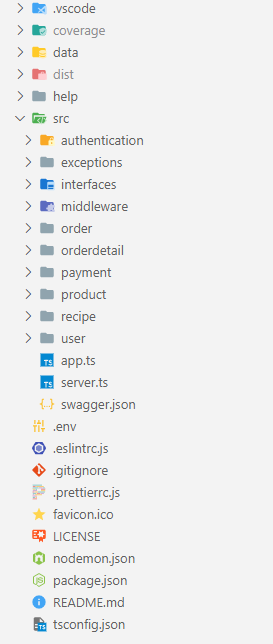
* 1. A publikálás

A publikálást a cyclik-en tőrténik ahol kőlön GitHub repositoryt kellet létrehoznunk hogy publikálni tudjuk a bacendunket .

A backend elérhető: https://shopking.cyclic.app/

Míg a frontendünket a netlify-al publikáltuk. A frontend elérhető: [https://shopk.netlify.app](https://shopk.netlify.app)

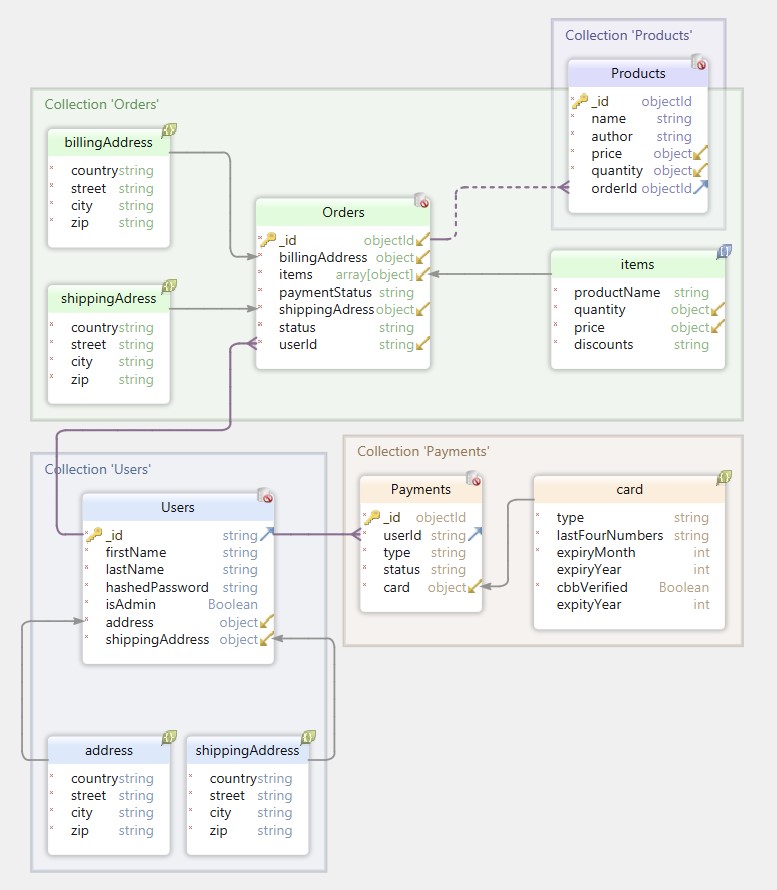
Illetve a Swagger dokumentáciot is megtalálható itt: <https://shopking.cyclic.app/docs>

1.  A program
   1. Technikai részletek

A frontend a Vue.js-el készült. Vue.js fokozatosan adaptálható architektúrával rendelkezik, amely a deklaratív megjelenítésre és az összetevők összeállítására összpontosít. Az alapkönyvtár csak a nézetrétegre fókuszál. Az összetett alkalmazásokhoz szükséges speciális szolgáltatásokat, például az útválasztást , az állapotkezelést és a build-eszközöket hivatalosan karbantartott támogató könyvtárakon és csomagokon keresztül kínálják.

A backend a node.js-el készült. A Node.js egy többplatformos , nyílt forráskódú kiszolgálókörnyezet, amely Windows , Linux , Unix , macOS és egyéb rendszereken futhat . A Node.js egy háttérben futó JavaScript futtatókörnyezet , a V8 JavaScript Engine -en fut , és a JavaScript-kódot webböngészőn kívül hajtja végre.

* 1. Az adatbázis felépítése



* + 1. Táblák

**Users** – felhasználok

* Id: elsődleges azonosító
* firstName: a felhasználó kereszt neve
* lastName: a felhasználó vezeték neve
* hashedPassword: regisztráció során a backenden kerül titkosításra a felhasználó által megadott jelszó
* isAdmin: jogosultságok kezelésére szolgáló mező
* address: objectum ahol a felhasználo lakcimét tárojuk
  + - country: a felhasználó nemzetiségét tárojuk
    - street: a felhasználó lakcimének utcáját tárojuk
    - city: a felhasználó városát tárojuk
    - zip: a felhasználó zip codját tárojuk

**Orders** - rendelések

* Id: elsődleges azonosító
  + userId: a felhasználó elsődleges azonosítója
  + paymentStatus: a felhasználó fizetett-e
  + status: a termék szálitásának állapota
  + billingAddress: a számlázási cím
    - country: a felhasználó számlázási címének nemzetiségét tárojuk
    - street: a felhasználó számlázási címének lakcimének utcáját tárojuk
    - city: a felhasználó számlázási címének városát tárojuk
    - zip: a felhasználó számlázási címének zip codját tárojuk

**Products –** termékek

* + Id: elsődleges azonosító
  + name: a termék neve
  + author: a termék írojának neve
  + price: a termék ára
  + quantity: a raktáron lévő mennyiség

**OrderDetails –** rendelés részletei

* + Id: elsődleges azonosító
  + productId: a termék elsődleges azonosítója
  + orderId: a rendelés elsődleges azonosítója
  + archived\_price: a rendelés ára leárazás előtt
  + discount: a leárazás mértéke
  + quantity: a rendelt mennyiség

**Payments -**  fuzetés

* + Id: elsődleges azonosító
  + userId: a felhasználó elsődleges azonosítója
  + type: a fűzetés modja
  + status: kifizetésre kerülte a termék
  + card: a kártya titkositott adatai
    - type: a kártya tipusa
    - lastFourNumbers: a kártya utolso 4 száma
    - expiryMonth: a lejárati hónap
    - expiryYear: a lejárati év
    - cbbVerified: a cbb kód elfogadásrakerulte

1. 1. Felhasználói fiók
      1. Regisztráció

A oldal szabadon használható azonban rendelés leadásához be kell jelentkeznie a felhasználónak. Ez a honlapon a /register útvonalon van a szűkséges adatok a regisztráciohoz

* Vezetéknév
* Keresztnév
* E-mail
* Jelszó

A regisztrációs és a bejelentkezési űrlapon is történik hibakezelés, így nem adható meg hibás adat.

1. 1. Bejelentkezés

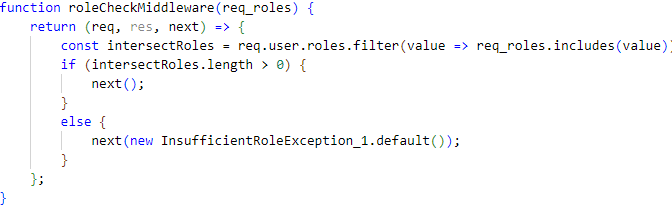
A bejelentkezési képernyő a honlapon a /login útvonalon a navbaron található menűponton Bejelentkezés csak a már regisztrált fiókokkal lehetséges.

* 1. Felhasználók számára elérhető funkciók

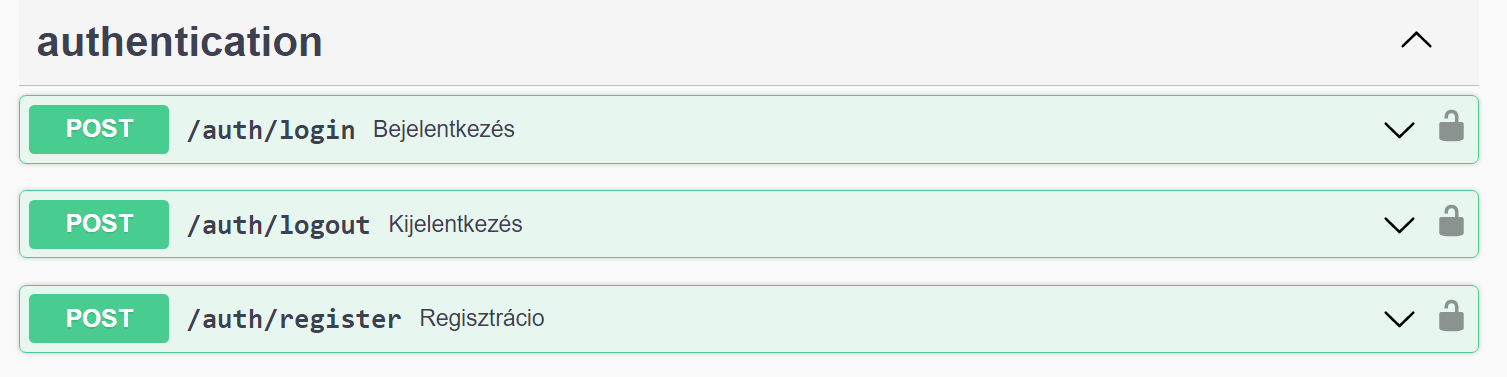
A bejelntkezés után a felhasználók számára elérhető lesz a rendelés lehetősége a felhasználó szabadon váltosztathatja a saját fiokját és hozzá férhet az átala rendel termékek adataihoz. Az admin felhasználóknak lehetőséguk van egy külön oldalon amihez csak ők férnek hozzá hogy küvetni lehesen kik rendeltek és ha egy felhsználó megkapta a rendelését vagy lemondta a admin felhasználok tőrőli tudják a rendelést igy késznek nyilvánitani azt

* 1. Autentikáció a backenden

A backenden a jelszót titkositjuk a bycript seggitségével igy a letárolt jelszó már nem fejthető vissza csak átalunk. a roleCheckMiddleware sgitségével elenőrizük a backenden hogy bizonyos controllerekhez csak a admin felhasználók férhesenek hozzá vagy minden felhasználó hozáférhessen



* 1. API dokumentáció
     1. authentication

****

Elérési útvónal: /auth

**Login()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: nincs korlátozva
* Feladat: bejelntkezés, token létrehozása
* Válasz: anonim osztály, mely tartalmazza a felhasználó adatait
* Elérési útvonal: /login

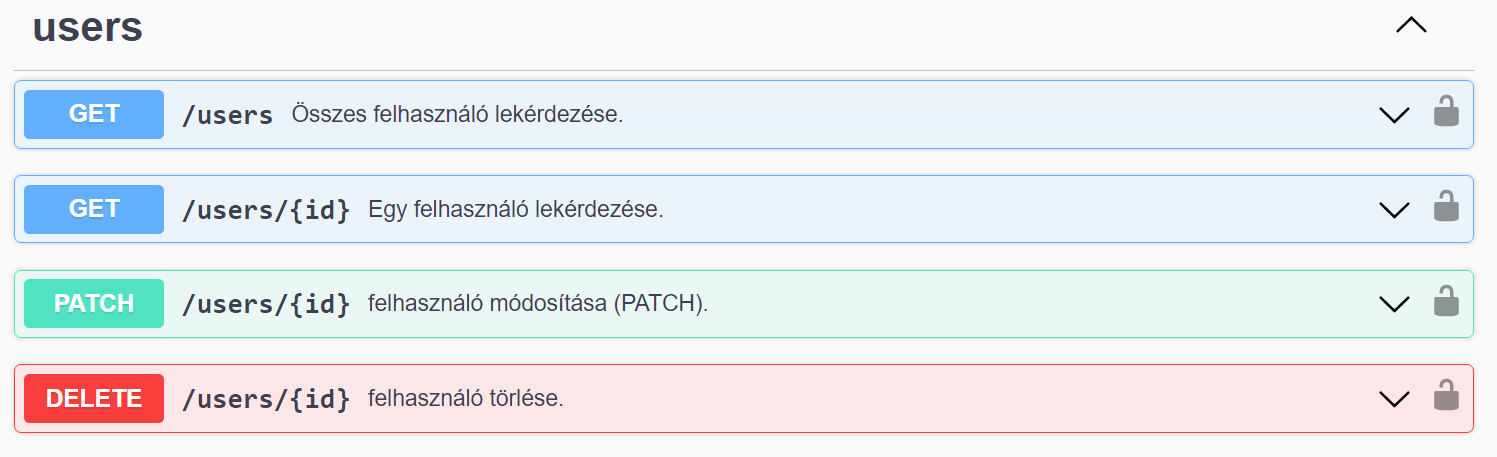
**Logout()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: User, Admin
* Feladat: a felhasználó kijelentkeztetése
* Válasz: viszadobja a logout sikereségét
* Elérési útvonal: /logout

**Register()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: nincs korlátozva
* Feladat: új felhasználó létrehozása
* Válasz: viszadobja a adatokat
* Elérési útvonal: /register

1. * 1. Users



Elérési útvónal: /users

**List()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Admin
* Feladat: Lekéri az ősszes felhasználló adatát
* Válasz: User typusú tömb
* Elérési útvonal: /

**GetById(int id)**

* Kérés típusa: HttpGet
* Paraméter: User azonositó
* Jogosultság: Admin
* Feladat: Azonosító alapján megkeresi és visszaadja az adott felhasználót, ha létezik
* Válasz: User típusú tömb
* Elérési útvonal: /id

**ModifyById(int id)**

* Kérés típusa: HttpPatch
* Paraméter: User azonositó
* Jogosultság: User, Admin
* Feladat: felhasználó adatainak módosítása
* Válasz: User típusú tömb
* Elérési útvonal: /id

**DeleteById(int id)**

* Kérés típusa: HttpDelete
* Paraméter: User azonositó
* Jogosultság: Admin
* Feladat: felhasználó tőrlése
* Válasz: User osztálypéldány
* Elérési útvonal: /id
  + 1. Orders



Elérési útvónal: /orders

**List()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Admin
* Feladat: Lekéri az ősszes order adatát
* Válasz: Order típusú tömb
* Elérési útvonal: /

**NewList()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: User, Admin
* Feladat: Lekéri az ősszes ordert
* Válasz: Order típusú tömb
* Elérési útvonal: /

**GetById(int id)**

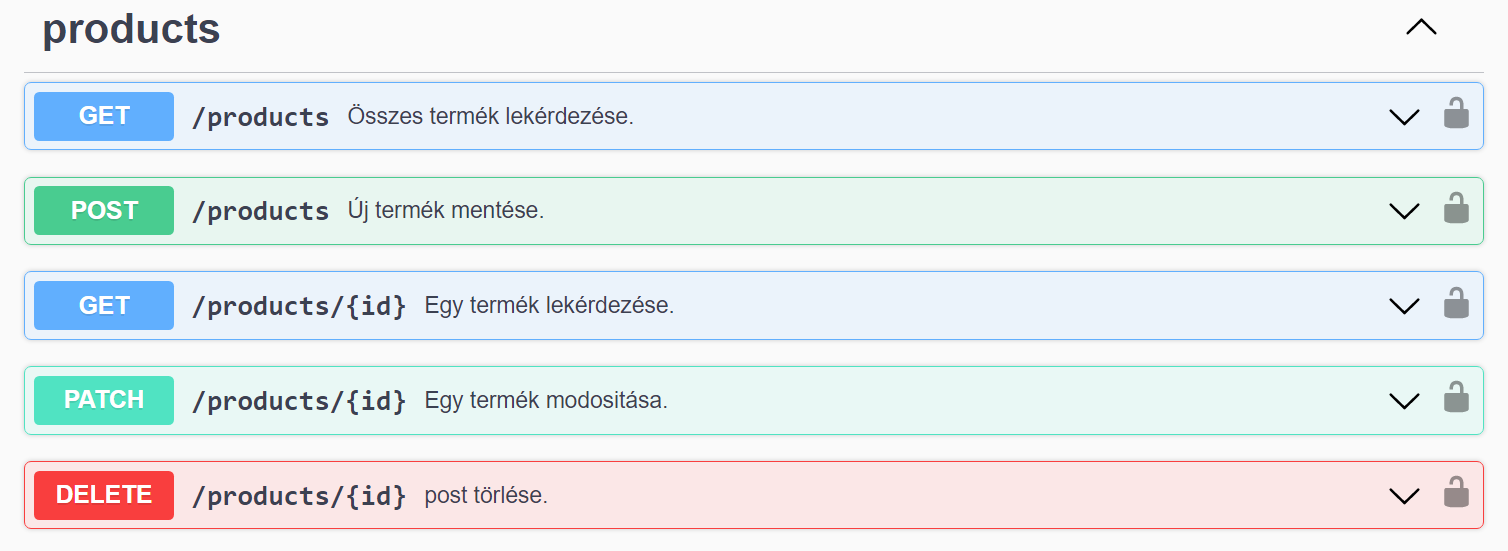
* Kérés típusa: HttpGet
* Paraméter: Order azonositó
* Jogosultság: Admin
* Feladat: Azonosító alapján megkeresi és visszaadja az adott ordert, ha létezik
* Válasz: Order típusú tömb
* Elérési útvonal: /id

**ModifyById(int id)**

* Kérés típusa: HttpPatch
* Paraméter: Order azonositó
* Jogosultság: User, Admin
* Feladat: order adatainak módosítása
* Válasz: Order típusú tömb
* Elérési útvonal: /id

**DeleteById(int id)**

* Kérés típusa: HttpDelete
* Paraméter: Order azonositó
* Jogosultság: Admin
* Feladat: order tőrlése
* Válasz: Order osztálypéldány
* Elérési útvonal: /id
  + 1. Products



Elérési útvónal: /products

**List()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Admin
* Feladat: Lekéri az ősszes products adatát
* Válasz: Products típusú tömb
* Elérési útvonal: /

**NewList()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: User, Admin
* Feladat: Lekéri az ősszes productst
* Válasz: Products típusú tömb
* Elérési útvonal: /

**GetById(int id)**

* Kérés típusa: HttpGet
* Paraméter: Products azonositó
* Jogosultság: Admin
* Feladat: Azonosító alapján megkeresi és visszaadja az adott products-ot, ha létezik
* Válasz: Products típusú tömb
* Elérési útvonal: /id

**ModifyById(int id)**

* Kérés típusa: HttpPatch
* Paraméter: Products azonositó
* Jogosultság: User, Admin
* Feladat: products adatainak módosítása
* Válasz: Products típusú tömb
* Elérési útvonal: /id

**DeleteById(int id)**

* Kérés típusa: HttpDelete
* Paraméter: Products azonositó
* Jogosultság: Admin
* Feladat: product tőrlése
* Válasz: Products osztálypéldány
* Elérési útvonal: /id
  + 1. Orderdetails

****

Elérési útvónal: /orderdetails

**List()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Admin
* Feladat: Lekéri az ősszes orderdetails adatát
* Válasz: Orderdetails típusú tömb
* Elérési útvonal: /

**NewList()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: User, Admin
* Feladat: Lekéri az ősszes orderdetailst
* Válasz: Orderdetails típusú tömb
* Elérési útvonal: /

**GetById(int id)**

* Kérés típusa: HttpGet
* Paraméter: Orderdetails azonositó
* Jogosultság: Admin
* Feladat: Azonosító alapján megkeresi és visszaadja az adott orderdetails-t, ha létezik
* Válasz: Orderdetails típusú tömb
* Elérési útvonal: /id

**ModifyById(int id)**

* Kérés típusa: HttpPatch
* Paraméter: Orderdetails azonositó
* Jogosultság: User, Admin
* Feladat: orderdetails adatainak módosítása
* Válasz: Orderdetails típusú tömb
* Elérési útvonal: /id

**DeleteById(int id)**

* Kérés típusa: HttpDelete
* Paraméter: Orderdetails azonositó
* Jogosultság: Admin
* Feladat: orderdetails tőrlése
* Válasz: Orderdetails osztálypéldány
* Elérési útvonal: /id
  + 1. Payments

****

Elérési útvónal: /payments

**List()**

* Kérés típusa: HttpGet
* Paraméter: nincs
* Jogosultság: Admin
* Feladat: Lekéri az ősszes payments adatát
* Válasz: Payments típusú tömb
* Elérési útvonal: /

**NewList()**

* Kérés típusa: HttpPost
* Paraméter: nincs
* Jogosultság: User, Admin
* Feladat: Lekéri az ősszes payments-t
* Válasz: Payments típusú tömb
* Elérési útvonal: /

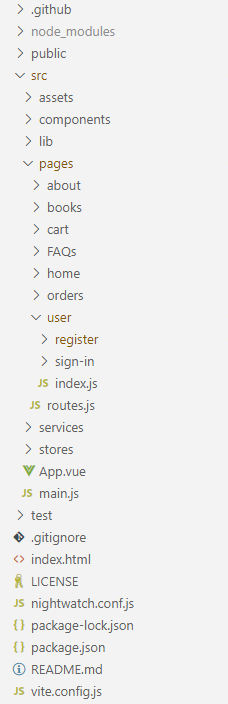
**GetById(int id)**

* Kérés típusa: HttpGet
* Paraméter: Payments azonositó
* Jogosultság: Admin
* Feladat: Azonosító alapján megkeresi és visszaadja az adott payments-et, ha létezik
* Válasz: Payments típusú tömb
* Elérési útvonal: /id

**ModifyById(int id)**

* Kérés típusa: HttpPatch
* Paraméter: Payments azonositó
* Jogosultság: User, Admin
* Feladat: payments adatainak módosítása
* Válasz: Payments típusú tömb
* Elérési útvonal: /id

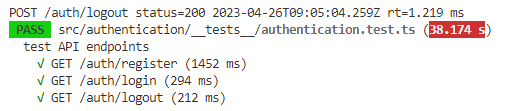
**DeleteById(int id)**

* Kérés típusa: HttpDelete
* Paraméter: Payments azonositó
* Jogosultság: Admin
* Feladat: payments tőrlése
* Válasz: Payments osztálypéldány
* Elérési útvonal: /id
  1. Frontend dokumentáció
     1. Felépítése

Frontendünk struktúrája főként a komponensekből, modellekből és servicekből áll. A modellek megegyeznek a backend service rétegének modelljeivel. A frontend servicei a backend azonos elnevezésű kontrollereinek RestAPI kéréseit és válaszait kezeli. A komponensek kezelik a webalkalmazás oldalainak megjelenítését. Elnevezésük beszédes, kifejezi, melyik oldal milyen funkciót biztosít. A bacendről az meghivásokat a Axios műkodteti

1. Tesztek
   1. Backend tesztek

Authentikáció teszt:

****

A backend teszteket Jest-el és szupertest segitségével végesztük el

Orders teszt:

