BGéSZC Kálmán Technikum és Kollégium

Szakképesítés neve: Szoftverfejlesztő

OKJ száma: 5-0613-12-03

**Vizsgaremek dokumentáció**

GrossKidz webáruház

Török Eszter Gébele Krisztián Dénes,Verdes Barna, Erdei Zoltán

Témavezető 2/14ACE

Budapest ,2023

**Tartalom jegyzék**

[Bevezető 3](#_Toc1402717274)

[Felhasználói rész a GrossKidz webáruházhoz: 4](#_Toc1411713497)

[Mi az a token amit neked biztosítunk? 5](#_Toc22782460)

[Kosarad és használata, vásárlás működése: 6](#_Toc508957273)

[Fejlesztői rész, Backend és arról röviden: 9](#_Toc1415651276)

[GrossKidz backend 11](#_Toc486850728)

[Frontend és alkalmazása: 24](#_Toc613101848)

## Bevezető

Napjainkban már szinte mindenki elgondolkozott legalább egyszer hogy legyen egy webáruháza, akár kisebb akár nagyobb. Újat manapság már kifejezetten nehéz alkotni de probléma mindig van mit megoldhatnak a fejlesztők:

* Hogy küzdjük le a **túlterhelést**: Adódhat hogy nem megfelelő kiépítés vagy technikai hátrányból megvannak a szerverek határai. Főleg ha belegondolunk egy időben mennyi ember vásárolhat szinte egyszerre, arról nem beszélve hogy a netes vásárlást biztonságossá tegyék közben.
* Hogy legyen a közeg minél egyszerűbben **alkalmazható és felhasználó barát**: Dizájn részen vagy funkcionalitásban is könnyen elvetheti bárki a sulykot és valami olyan végeredménnyel találkozhat amivel szemben az egyszerűbb megoldás célszerűbb lett volna.

**Témaválasztásunk indoklása**

Közös megegyezés alapján eldőlt, hogy a GrossKidz lemezkiadónak készült webáruházat alkossuk meg. Zenét manapság alig akad aki nem hallgat, kedvenc előadóink márkázott temérkeit is örömmel hordjuk amivel őket támogatjuk, ebben az esetben kényelmes lehet ha erre az igényre megtaláljuk azt a helyet ahol egyszerűen és biztonságosan megkapjuk a kívánt termékeinket.

## Felhasználói rész a GrossKidz webáruházhoz:

Főoldalunkon megtekintheted a jelenlegi tagok képét hisz nélkülük nem lenne az érintett lemezkiadó az ami. Ha kedved tartja gombnyomásra hallgathatod zenéjiket az oldalon látható spotify ablakból. Esetleg ha kíváncsi lennél mélyebb információkra a termékekkel kapcsolatban a Termék infók ablaknál megtalálod. Nem utolsó sorban a navigációs sáv is megtalálható kifejezetten könnyen hisz az oldal tetején helyezkedik, valamint az eltéveszthetetlen logó mindenképp nem durván de jól látható formában is jelen van. Igyekeztünk törekedni arra hogy a felhasználói élményt ne rontsa ha esetleg kisebb eszközről tekintenél meg mindent és ebből amennyire tudtunk nem engedtünk a többi oldalon se. A termékek megtekintéséhez csupán átkell térned a kívánt terméked részére legyen az pulóver vagy nadrág, ahol az összes rendelkezésre álló méretben lévő termék fogad.

Megtekintheted bármelyik terméket amelyik megtetszik, ebben az esetben a termék megtekintése gomb nyomása ezt intézi neked. Ha a terméket már el is helyeznéd a kosaradba csupán bejelentkezve kell lenned egy már regisztrált felhasználói fiókkal. [Boostrap](https://getbootstrap.com) 5 alkalmazásával tettük minél jobban reszponzívvá, valamint alertek és üzenetekhez Frontend részen a [Sweetalert2](https://sweetalert2.github.io/#examples)-t használtuk

**Bootstrap:**

A Bootstrap egy nyílt forráskódú CSS keretrendszer aminek célja a reszponzív, többnyire telefonra keszült weboldalak készítésére alkalmazzuk. Részei HTML, CSS és (tetszőlegesen) JavaScript-alapú dizájnra szánt template-ek,formok,gombok,navigation, és más interfész komponensek. Lényege hogy a kijelzőnk méretéhez igazítva tudjuk a weboldalak megjelnítését alakítani, garantálva azt hogy mindenhol megfelelő felhasználói élményt biztosítson.

Biztosít továbbá 5 előre megszabott képernyőméretet amiket használhatunk akár a “container”-eknél.

* Kisebb mint 576 pixel
* 576–768 pixel
* 768–992 pixel
* 992–1200 pixel
* Nagyobb mint 1200 pixel

Telepítése mi esetünkben: npm i bootstrap@5.3.0-alpha3

**SweetAlert:**

Lényegében olyan mint egy modál, remekül használható az alap Js pop-up boxok helyetesítésére. Jellemzői hogy responszív, letisztult, remek dizján társul hozzá és sokrétű. Használata is egyszerű és rengeteg alfajtájának köszönhetően sokrétűen használható frontend részen.

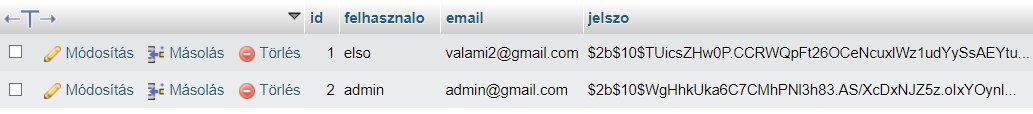
Telepítése: npm install sweetalert2, vagy script tag-ben:

<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

Telepítés után azon az oldalon ahol használni szeretnénk beimportáljuk és a megfelelő szintaktikát alkalmazva használjuk.

**Regisztráció és Bejelentkezés:**

Ha már regisztráltál weboldalunkra akár az oldalra való fellépéskor, vagy a kosár megtekintésekor bejelentkezhetsz. Add meg csupán a jelszavad és a felhasználói nevedet és már bent is vagy, a regisztrációt a belépés alatt lévő linkkel kezdheted meg. Csupán adj meg egy érvényes email fiókot, tetszőleges felhasználó fiókot és egy jelszót melyet a továbbiakban is használhatsz. Felhasználó nevedet, emailedet eltároljuk hisz ezekre szükség lehet jelszavadat adatbázison belül titkosítva tároljuk melyet itt lent láthatsz, fontos hogy jelszavaddal és más adataiddal sem élünk vissza. Minden továbbiakban a rendszer a fiókodat kezeli a bejelentkezéskor számodra generált Tokennel.

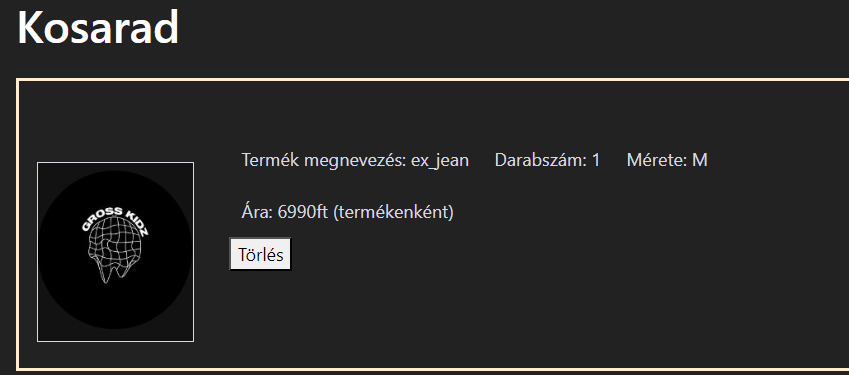
*Titkosított jelszavak az adatbázisban.*

## Mi az a token amit neked biztosítunk?

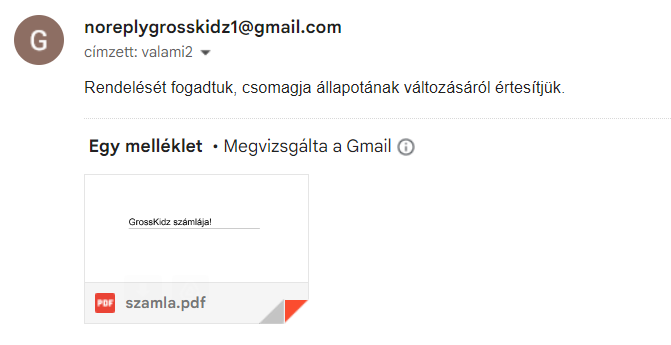
Minden alkalommal mikor bejelentkezel a rendszerünk biztosít neked egy Tokent, mely teljesen egyedi és nincs másik belőle. Tokened bejelentkezéstől egy órán át el, amint letelik az időd újra kell jelentkezned hogy vásárolhass. Mivel a vásárláshoz szükséges csupán ez a kis token de termék megtekintéséhez nem ezért a kiszabott egy órát tartjuk, természetesen ezekkel sem élünk vissza, kitérünk arra is használat után hogyan kezeljük őket. Azt hogy a tokened egy óra után lejárjon szerverünk egy modulja biztosítja, mely figyeli az időtartamot. Amint letelik az 1 óra és tokened “élete” a weboldalon lejár bekerül egy feketelistára az adatbázisunkba. Minden token kiosztásnál ellenőrizzük a feketelistánkat és az esetlegesen ott szereplő tokeneket, elkerülve azt hogy a te korábban használt tokened más ne megkapja.

## Kosarad és használata, vásárlás működése:

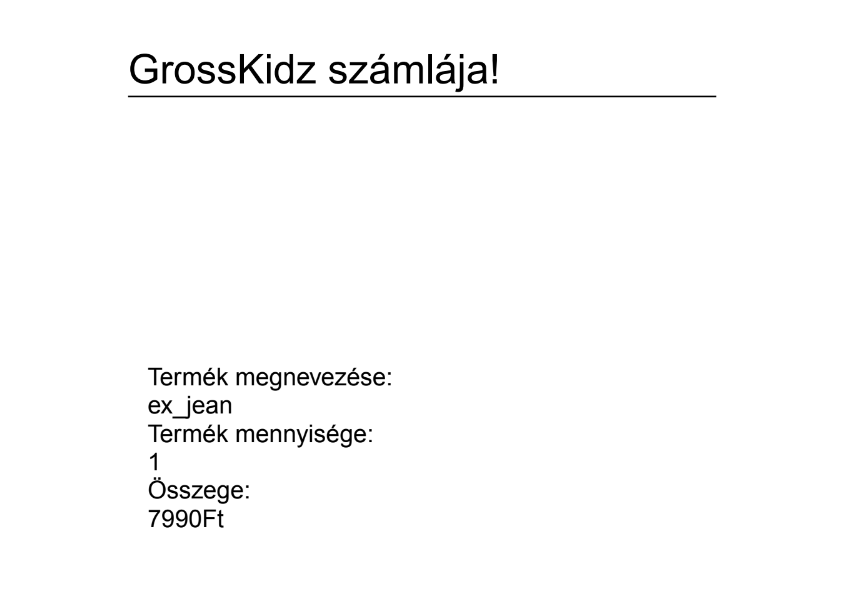
Korábban említettük, hogyan érhetsz el egy olyan terméket amely megtetszett számodra. Természetesen hogy ne legyen túlbonyolított a vásárlás menete itt egy gombnyomásra terméked a kosárba kerül, kosaradban található termékek törlésére is adott a lehetőség az itt található törlés gombbal. Terméked kosárba helyezése után és a vásárláshoz elengedhetetlen bejelentkezést követően kosarad tartalmát megrendelheted. Kosarad teljes tartalmát a regisztrációnál megadott emailfiókra kiszámlázzuk és ezen az emailcímen tájékoztatunk rendelésed esetleges állapot változásáról. Természetesen, ha már van egy terméked a kosárban de mégis vennél mást, az oldal elhagyása vagy frissítése esetén sem tűnnek el termékeid.



Maga a kosár és tartalmának letisztult megjelenítése.



Az említett számla mely rendelést követően az emaileden vár.



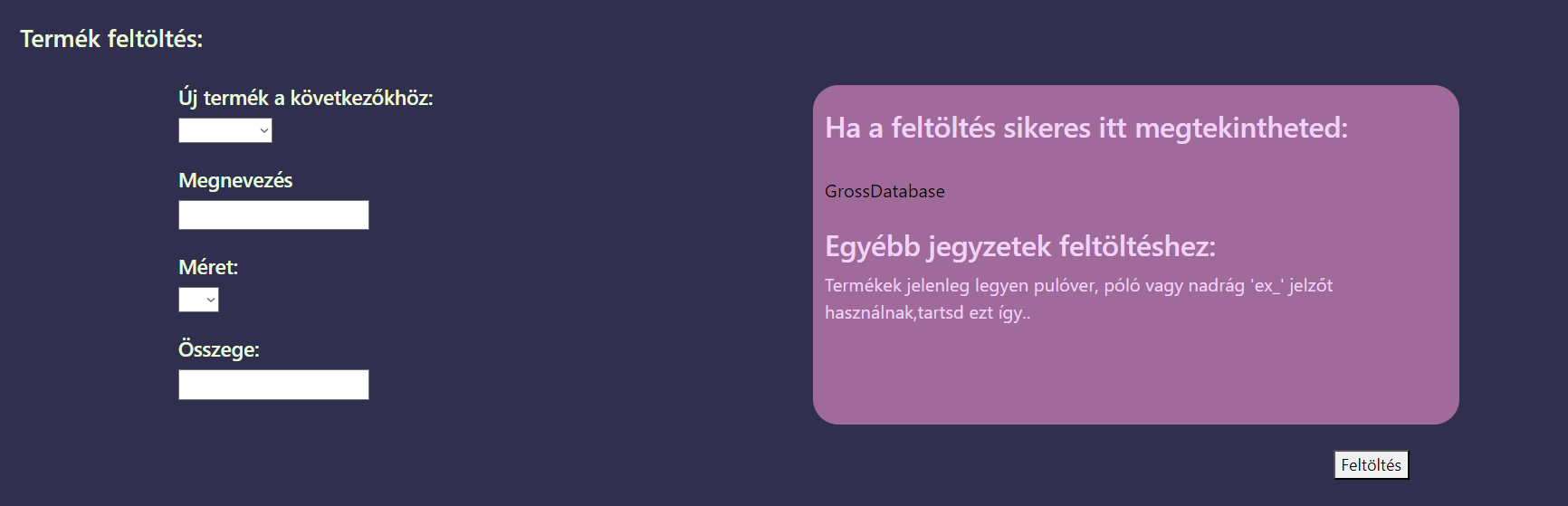
Valamint a számlád tartalma rendelésedtől függően:

A fentebb említettekkel együttesen igyekszünk minél elfogadhatóbb és jobb felhasználói élményt biztosítani, megőrizve a biztonságodat is. Abban az esetben is ha a továbbiakban oldalunk fejlődne a felhasználói igényekből nem engedünk.

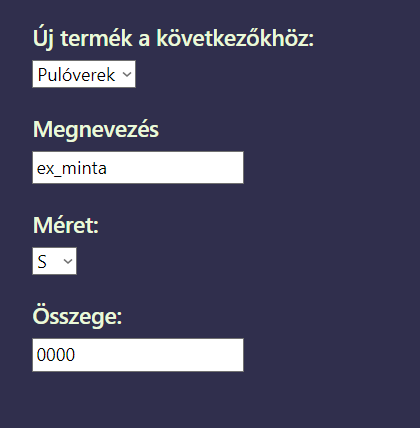
**Admin részről:**

Biztosítva van természetesen az admin oldalunk is ahol terméket vihetsz fel adatbázisunkba, abban az esetben ha terméket távolítanál el pedig az adatbázisba való átirányítást is biztosítottuk.

***Adminként ez a látvány vár rád:***



Termék feltöltését egyszerűen és könnyen intézheted innen kitöltve minden hozzátartozó adatot. Természetesen a jegyzet füleknél az adatbázison kívül az éppen életben lévő jegyzetek segítséget nyújtanak ha rászorulsz. Az adatok megadása után feltöltésnél minden adattal a termék felkerül adatbázisunkba és onnantól meg is jelen az oldalon.



Szükséges adatok megadása és feltöltés admin oldalról.



A frissen feltöltött termék adatbázisban.

## Fejlesztői rész, Backend és arról röviden:

Mi az a backend: A Back end a háttérben futó folyamatokkal foglalkozik, pl. szerveroldali programozással, űrlapon beküldött adatok feldolgozásával, statisztika készítéssel – tálalással például. Szinte leggyakrabban Php-val találkozhatunk backenden belül, mi viszont Node.Js-t alkalmaztunk.

***Php-ról röviden:***

Hogy érthetőbb legyen Php alatt egy szerveroldali dinamikus szkript nyelvről van szó melyet természetesen weboldalak készítésére használunk. Erőssége hogy külső fájl használata helyett HTML oldalba ágyazható aminek segítségével kényelmesebb használatot eredményezhet.

A hagyományos HTML lapokkal ellentétben a kiszolgáló a PHP-kódot nem küldi el az ügyfélnek, hanem a kiszolgáló oldalán a Php-értelmező motor dolgozza fel azt. A programokban lévő HTML elemek érintetlenül maradnak, de a Php kódok lefutnak. A kódok végezhetnek adatbázis-lekérdezéseket, létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli kiszolgálókkal. Számos rendszergazda használja automatizálási célokra eltekintve attól hogy ezt hagyományosan Perl vagy Shell szkriptekkel oldották meg. Nem vonatkozott kikötés arra hogy mást nem lehet használni ezért mi esetünkben a teljes backendünk Node.Js-t alkalmaz.

***Node.Js:***

Olykor félreértik hogy a Node csak és kizárólag Backendbeli munkára használható, egyaránt lehet Frontendbeli részeken alkalmazni de tény, gyakoribb Backenden. Maga a Node.Js egy nyílt forrású cross-platform szerver környezet mely elfut bárhol, legyen szó Windows-ról, Linuxról de akár macOsról is. Röviden egy JavaScript nyelven működő valós időben futó szerver környezet amit a V8 Js motor hajt meg, és Js kódokat végez el a webünkön.

Node.Js lehetővé teszi többek között azt is hogy a Web szerverek is használhassanak JavaScriptet és egy egész kollekciónyi “modult” biztosít ami kezeli egyes esetekben az alap funkcionalitást. Számos modul közül van például amit használunk hálózati körökben ( DNS, HTTP) de akár adat kezelési körökben is.

Szerverünket egyszerűen indíthatjuk miután ellenőrizzük hogy jó mappában vagyunk e és a konfiguráció is hibátlan, csupán egy terminálba beírjuk a következőket: npm start. Ezzel a sima paranccsal szerverünk a megadott porton elindul. Maga a képessége hogy Js kódókat alkalmazhassunk egy szerveren teszi használhatóvá arra hogy dinamikus weboldal tartalmat hozzunk létre mielőtt az a felhasználó elé kerül. Elhanyagolhatatlan része hogy rendelkezik egy olyan eseményalapú szoftverarchitektúrával, amely segítségével képes legyen aszinkronizációra amit mi is (függvényeinknél többnyire) alkalmaztunk.

Ha ennyi nem lenne elég, ez a pár nagy vállalat sok másik közül is alkalmaz Node.Js-t a dinamikus weboldalak számára:

* LinkedIn
* Netflix
* Trello
* PayPal
* EBay

Php-val szemben összevetve ha szeretnénk egy “skálázható” web projektet ami egyenletesen és zökkenő mentesen gyűjt be adatot összefüggő vagy megszokott adatbázisból akkor mi is ajánljuk a Node.js-t. Ha szeretnénk hozzáférést több adatbázis kezelési lehetőséget, erre php a mi fegyverünk.

## GrossKidz backend

***Telepítési útmutató Node.Js használatához:***

Kezdetként letöltjük a forráskódót az [eredeti odalról,](https://nodejs.org/en/download) majd a letöltést követve telepítjük. Ezt folytatva VS Codeban létrehozunk egy directoryt ahol a projektünk lesz. Vagy egy üres megnyitunk vagy Terminalban a mkdir {mappa neve} segítségével létrehozzuk. Szükségünk lesz egy app.js fájlra ami a szerverhez szükséges részeket tartalmazza:



Végül egy npm start paranccsal elindul projektünk és kezdhetünk is a munkához. További segítség [itt](https://nodejs.org/en/docs/guides/getting-started-guide) található.

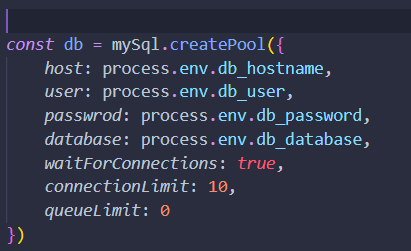
**Npm vagyis Node package manager:**

Ez az alap package menedzser amit használ egyaránt Node és Vue is. Ez áll az npm kliensből, egy online adatbázisból ami tartalmaz publikus és egyaránt fizetős csomagokból amit npm registry-nek hívunk. Továbbiakban akkor találkozunk vele ha a szervert inditjuk Node-on belül: npm start, Vue.js indításakor npm run dev, valamint csomagok kezelésekor npm i , mellyel ellenőrizzük van e rendelkezésre álló frissítés a csomagjainkhoz. Új csomag **telepítésekor** npm i {csomagnév};

**Adatbázis csatlakoztatása:**

Erre a célre létrehoztunk egy .env fájlt mely tartalmazza a szükséges adatokat(hostname,user...) valamint itt hoztuk létre a secret változót ami a Bcrypt-nél kerül alkalmazásra.

Egy db változó alkalmazásával a szerverünkön létrehozunk egy mySql poolt mert ez fogja a query-k lefutásakor az elérést biztosítani. Továbbá szükségünk van az előbb létrehozott .env fájlra is. A require(‘dotenv’).config() segítségéveltudjuk titkosítani a lent mutatott adatbázis részeket(host, felhasználó,jelszó,database) hiszen az .env fájlok nem kerülnek forráskódón belül tárolásra, valamit csatlakozási limitet is beállíthatunk. és már dolgozhatunk vele ezt követően.



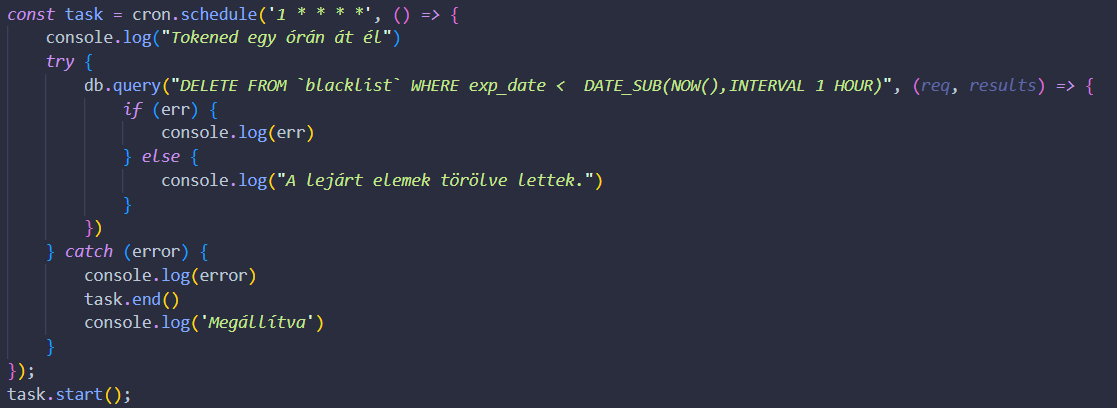
MySql pool létrehozása

***RestApi:***

A RestApi XML vagy JSON formátumban képes erőforrás adatokat megadni. A külső alkalmazásokat a RestApi segítségével végezhetik el az alkalmazásadatok lekérdezését és frissítését. A szabványos HTTP Get, Post, Put, Delete metódusok használatával támogatja az erőforrásokon létrehozási, lekérdezési, frissítési és törlési műveleteket. Személy szerint mi csupán Get és Postot használtunk melyekre lejjebb kitérünk részletesen. A Node.Js számára az npm az alap package kezelő melyet szinte állandóan használunk. Legyen szó a mi esetünkben JSON Web Tokenekről (jwt), cron időzítőről minden package alkalmazása telepítésével indul: Npm i (package neve). Egyik alaposzlopa szinte minden weboldalnak a biztonság megteremtése és annak kiépítése, nálunk itt jön képbe a jwt tokenek használata és a bcrypt jelszó titkosítás ami nem fejthető vissza.

***Cron időzítő:***

A bejelentkezést kővetően lép élet, és beállításának köszönehtően egy órán ált van életben.Amint a rendelkezésre álló egy óra letelik indítunk egy query-t hogy töröljük a lejárt tokent amit az exp\_date-től számolunk.Elindiásra task.start(), leállítására task.end() beépített függvényeket használunk.



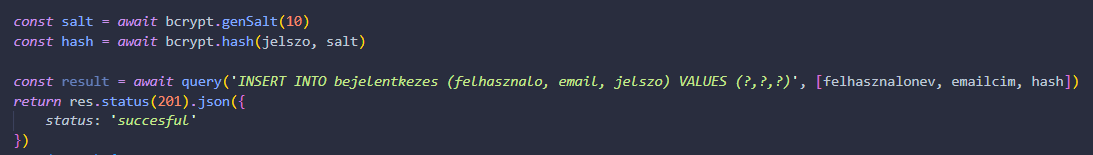
Cron időzítő használata

***Jwt Tokenek:***

Jwt (JSON web token) egy generált 32 karakterből álló stringet amibe kódolva vannak információk melyeket nem lehet visszafejteni és biztonságot nyújt a felhasználó számára. A bejelentkezés során biztosít rendszerünk egy akkor legenerált tokent ami teljesen személyreszabott és a fent említettek alapján is biztonságos. Két féleképp ér véget egy token élete nálunk: a felhasználó vagy kijelentkezik vagy lejár a használati ideje, utóbbira egy külön package-et használtunk. A Cron package segítségével egy órás időtartamot állítottunk be. Ezután amellett hogy a token eltávolításra kerül a felhasználótól, elhelyezzük adatbázison belül a feketelistánkra. Abban az esetben ha a felhasználó idő elött kijelentkezik a token akkor is feketelistára megy, itt kezeljük a végső sorban a tokeneket, ahol egy rövid ideig tároljuk. Tárolásra szükségünk van mert automatikus törlésre nem keríthetünk sort, mivel a Node.Js alapú szerver folyamatosan fut, ezért itt tároljuk egy órán át. Hogy mióta tároljuk itt ezt a tokent azt az exp\_data határozza meg. Az itt lévő tokent egy óra múlva töröljük, ezalatt még odafigyelünk arra bejelentkezésnél hogy az újonnan generált token nem egyezik-e a feketelistással.

***Bcrypt:***

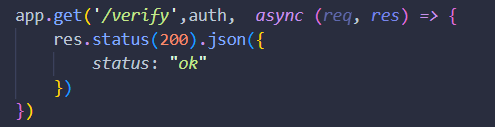
Ezt a packaget használjuk a jelszavak titkosítására. Egy szimpla jelszót egy fix hosszúságú karakter láncá alakít amit mi “hash”-nek hívunk. Azonban mielőtt egy jelszót “hash”-elünk a bcrypt autómatikusan elhelyez benne egy úgynevezett “salt”-ot, mely lényegében egy egyedi string ami kiszámíthatatlanná teszi a hash-t. Részünkről ez a folyamat regisztráció során történik, hisz az adatbázisban sem tároljuk el a konkrét jelszót, növelve ezzel a biztonságot.

A jelszón való bcrypt alkalmazása, hash és salt folyamat.

Onnantól hogy a rendszerben lévő felhasználók jelszavát bcrypt-el titkosítjuk, a bejelentkezésnél is előkerül még ezek használata melyet lejjebb mutatunk.

**Felhasználói hitelesítés:**

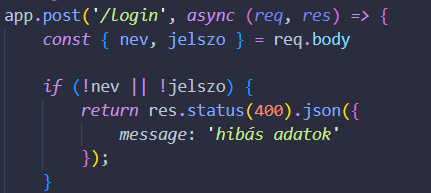
Get metódus használatával a hitelesítésünk bejelentkezés során történik admin és normál felhasználó számára egyaránt. Ellenőrzi az auth függvény meghívásával hogy a felhasználó tokene rendben van e. Mivel a bejelentkezést követően fut le ezért asynckronizált. Ha a status code amivel visszatér megfelelő, sikeres az hitelesítés és minden rendben zajlik tovább.



*Hitelesítés async függvénye a /verify útvonalon*

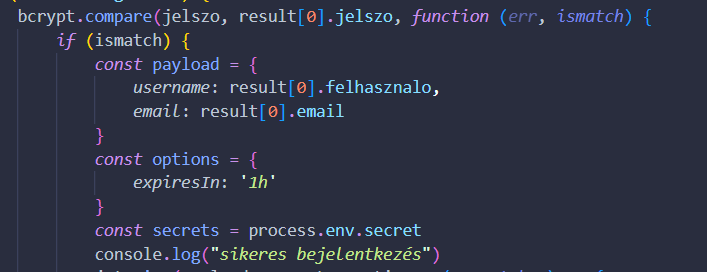
**Bejelentkezés:**

A regisztráció és kijelentkezés melletti szinte elengedhetetlen folyamatok egyike a bejelentkezés, melyet Post metódus használatával valósítottunk meg. Itt is természetesen asynckronizált függvényről van szó. Lekérjük egy változóba a nevet és jelszót bodyból mert ezekkel dolgozunk később. Hiba esetén itt egyből értesülünk hogy hibás adatokat adtunk meg.



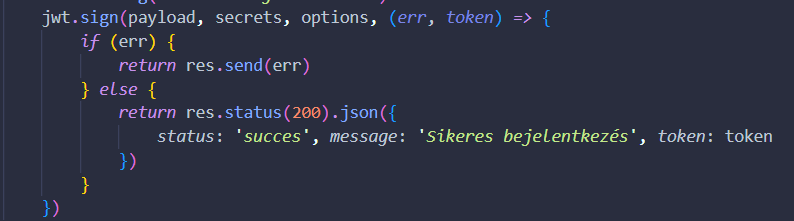
Megadott név és jelszó lekérése és ellenőrzése

Mivel a jelszó a rendszerben titkosítva van így itt a bcrypt compare-használatával ellenőrizzük hogy helyes a megadott jelszó. Itt adjuk meg siker esetén hogy a token 1 órán belül lejár, valamint tartjuk a titkosítást.



bcrypt compare használata, token időtartam beállítva

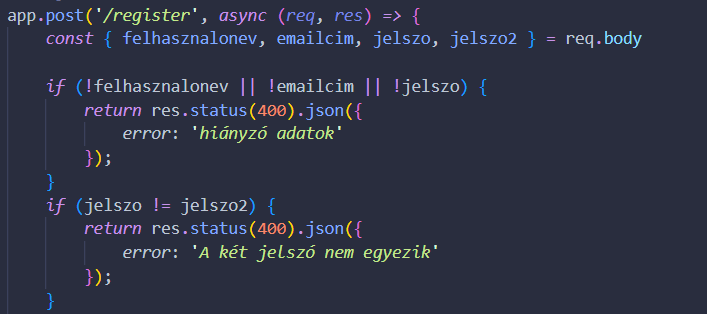
Ha minden adat megfelel a jwt.sign() segítségével megtörténik a token generálása ami az utolsó lépcsőfok a bejelentkezésnél. Sikeres generálás esetén a felhasználó megkapja a tokent és jogosult vásárlás kezdeményezésére.



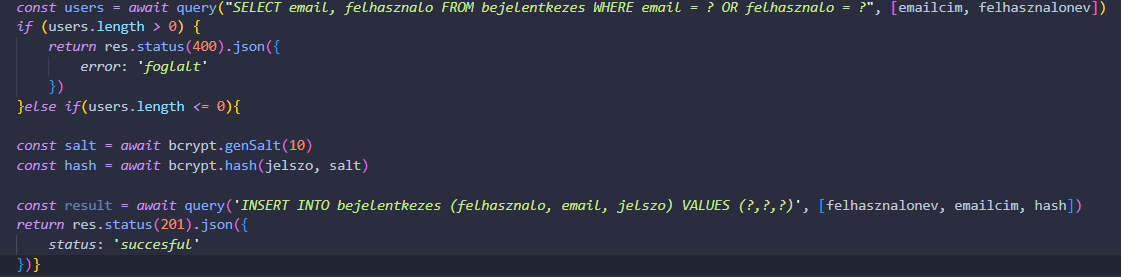
Token generálás és a sikeres bejelentkezés kezelése

**Regisztráció:**

Hasonlóan mint a bejelentkezésnél, ez is Post metódus használatával jön létre. Itt is lekérjük a leendő felhasználó adatait és ellenőrizzük hogy minden megfelel velük. Ha ez a rész megfelelt, továbbiakban ellenőrizzük azt is hogy a megadott email cím és felhasználónév foglalt e. Ha a felhasználó ezen a részen is túl jut az általa megadott jelsót bcryptel titkosítjuk ahogy korábban említésre került és végül felküldjük a rendszerbe. Regisztráció után a felhasználónk egyből bejelentkezhet vagy tetszés szerint áttérhet más oldalra.



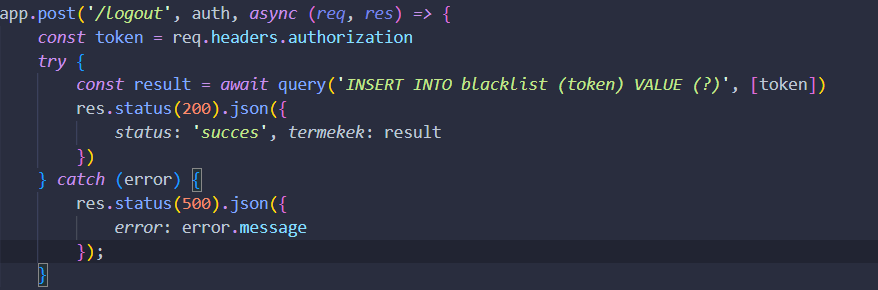
Regisztrációban történő adatok ellenőrzése



Jelszó titkosítás ha addig a pontig is minden megfelel a regisztrációhoz

**Kijelentkezés:**

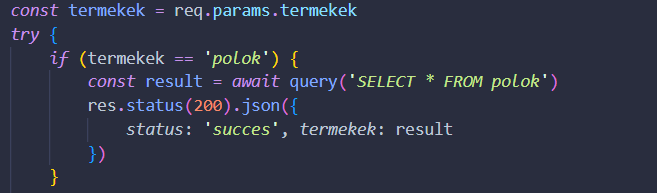
Végülaz alap funkciókból csak a kijelentkezés maradt, ez is egy Post metódussal kerül megvalósításra. Itt csupán lekérjük a token body-ból hisz kijelentkezés következtében az eddig használt token felkerül a feketelistára majd az tárhelyéből is eltűnik. Sikeres kijelentkezés esetén a felhasználó vagy admin visszakerül a kezdéshez ahol betud jelentkezni, vagy terméket megtekinteni tetszés szerint.



Kijelentkezés kódban, a fent említettek alapján

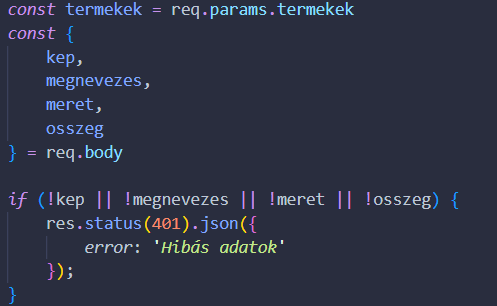
**Termékek lekérése, kezelése:**

Webshopunk termékeit elérhetik a termék fajtája alapján. Abban az esetben ha az adminunk feltölteni szeretne terméket a metódus Post. Fajtájuk alapján megvalósítás egy Get metódussal indul, valamint rendelkezik az eltérő útvonallal az id alapúhoz képest. Kezdésként lekérjük termékek változónkba a termékeket.



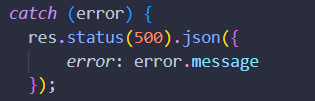
Termékek változó létrehozása, vizsgálat hogy pólókat kértek-e

Ha a felhasználó a pólókat szeretné megtekinteni, result változónkban az összes elérhető póló bent lesz. ***Feltöltés*** alig tér el ettől, itt is úgy indul hogy lekérünk paramétereket egy változóba mert ezekkel dolgozunk később. Egyszer mindenképp rávizsgálunk arra hogy rendben vannak e a lekért adatok.



Adatok lekérése és a lekérés eredményességének ellenőrzése

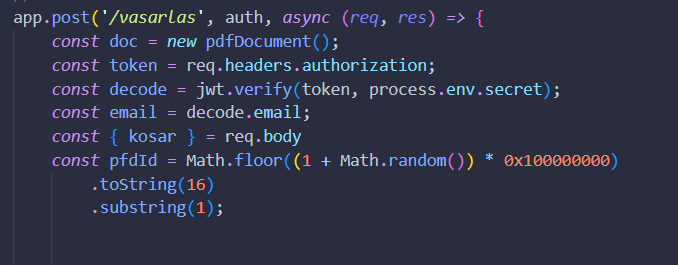
Ahogy a kódban is látható egy nagy try-catch része. Ha a lekérés nem sikerülne az errort is visszaadjuk.



Hiba esetén a catch rész biztosítja a visszajelzést

**Vásárlás és menete, email küldés:**

Ezen a részen valósítjuk meg a vásárlást és az azt követő email küldést. Jelen esetben Post metódust alkalmazunk a ‘/vasarlas’ útvonalon. Szerves része a vásárlás lebonyolításának hogy async függvényről van szó így ez is szerepel az authorizáció meghívása mellett.



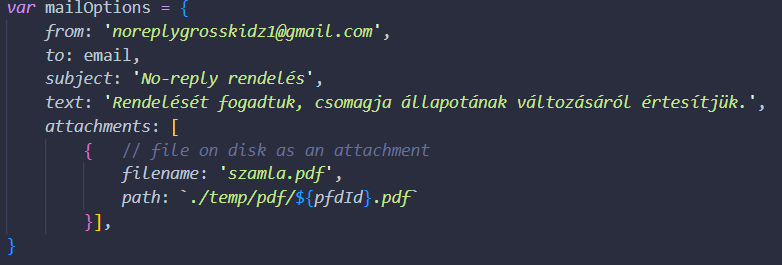
Szükséges csomagok behívása, pdfId készítése

Ennek a lebonyolításához szükségünk volt NodeMailer package-re hogy emailt küldhessünk, pdfKit-re amit a pdf számla létrehozásához és kezeléséhez használtunk.

Sikeres vásárlás végén elküldünk egy emailt ami a számlát tartalmazza rendeléstől függetlenül. Szükség volt erre egy a projekthez tartozó emailre, engedélyezni kell a használathoz a két faktoros authentikációt melyet a transporter változón belül találunk.

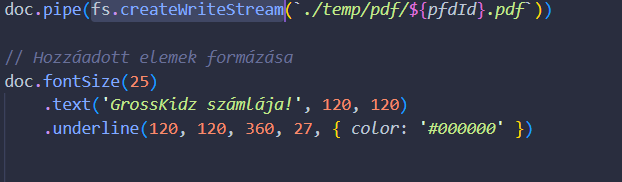
A küldő email, szolgáltatás valamint a két faktoros auth-hoz szükséges jelszó is itt található

A mailOptions változó tartalmazza hogy Ki küldi és kinek az említett emailt, milyen üzenettel és milyen tartalommal.



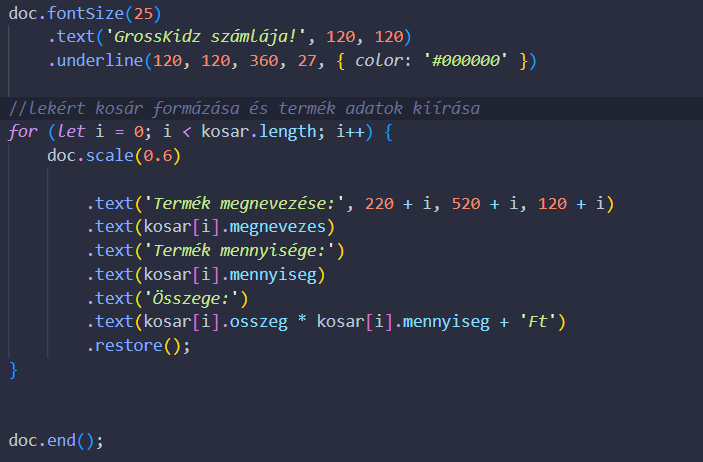
Email küldés alapjainak lebonyolítása

A mellékelt képen látszik a fájl neve és elérése. Maga a számla egy generált pdf a kosár adataival amint kiküldünk arra az emailcímre amit a felhasználó regisztrációnál megadott. Kezdésként létrehozzuk a pdf számlánkat ami a számlánk lesz, fs.createWriteStream() függvény használatával megadjuk az elérési útat a számlánkhoz. Lehetőségünk van formázni a hozzáadott elemket melyekből a számlánk áll a doc.fontsize(); text() és underline() mi esetünkben az amiket használtunk.



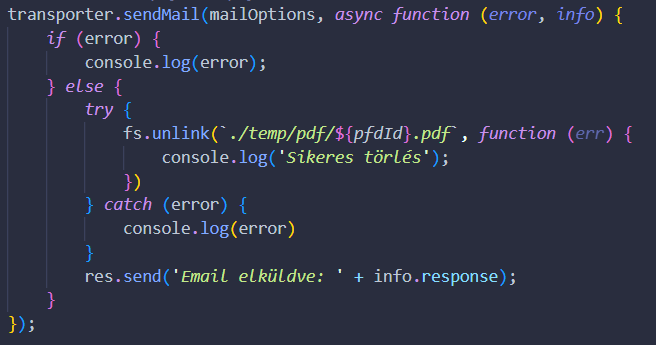
Pdf létrehozás és elemei formázása

Ahhoz hogy megjelenítsük miket rendeltünk szükségünk van kezdésként a kosárra. Ezt listába eltárolva lekérjük a body-ból mert így fogunk dolgozni vele. Egy for ciklus segítségével bejárjuk a kosár tömbünket és onnantól már csak formázásokkal kiiratjuk. Ha végeztünk a dokumentummal a doc.end() -et használva jelezzük hogy hol ér véget a kódolásunk.



Termék adatok megjelnítése és a számla minden elemének formázása

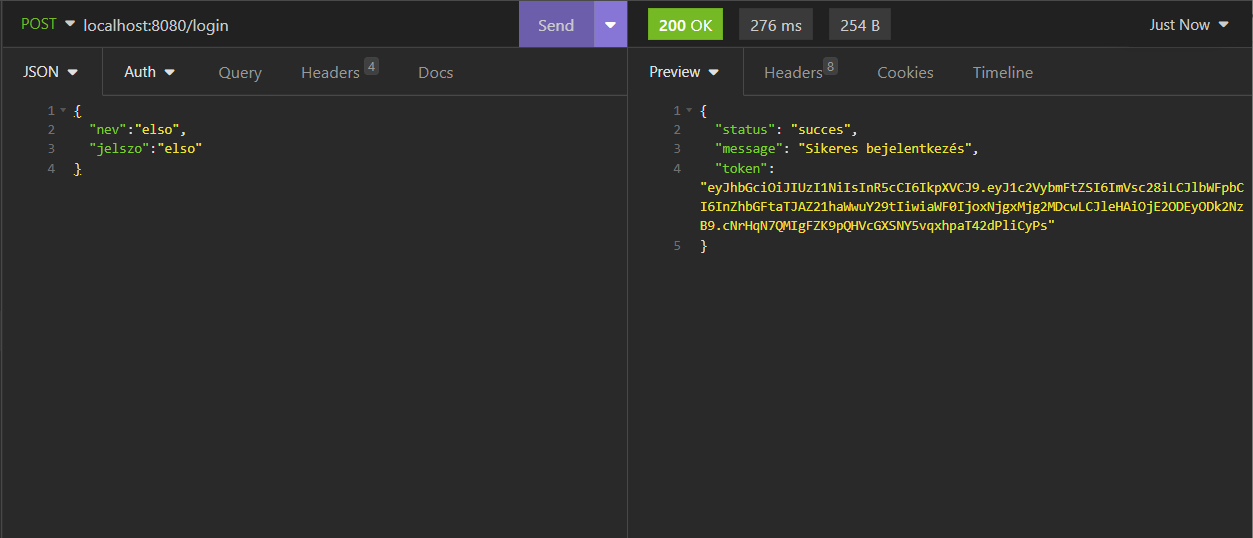
Ezen a ponton már elkészült a számlánk, megvan a hozzá tartozó elérési út, valamint az email is amiről küldhetjük. Így már csak a küldés maradt a sendMail() függvénnyel. Itt a transporter változóhoz fűzve a sendMail-t, mailOptions változónkat itt használva mely tartalmazza hogy ki és kinek mit küld a feladat vége felé haladunk. Miután nem futunk hibára és az email sikeresen lett elküldve fs.unlink() -et alkalmazva a gépünkről töröljük a számlát. Így a temp/pdf mappában nem halmozódik fel rengeteg régi számla. Sikeres emailküldés esetén es.send() segítségével visszajelzünk hogy minden rendben zajlott és az emailt is küldtük.



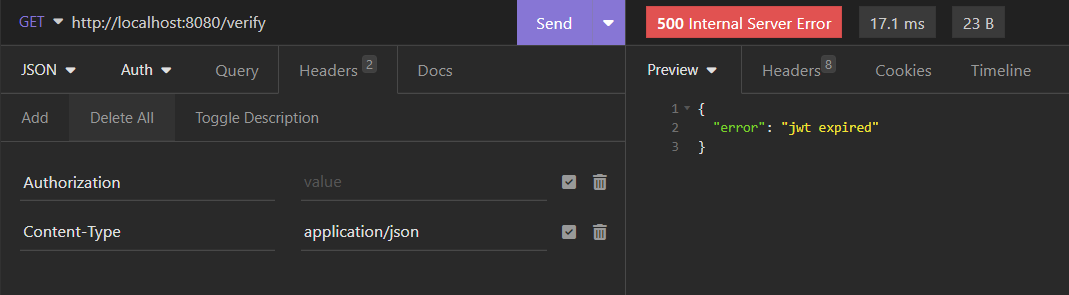
A számla emailben történő küldésének utolsó lépése

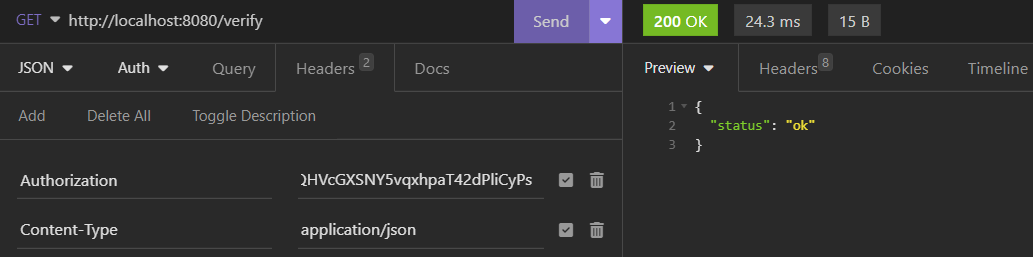
**Backenden történő eseményeink készítésekor tesztelésekhez használtunk Postmant méghozzá Insomniat.**

Lehetszó loginról, hitelesítésről vagy bármiről itt egy megadott útvonallal és szükséges értékpárok megadásával megnézhetjük működőképes e munkánk.

Bejelentkezés ellenőrzése és a hozzátartozó eredmény

Egyes esetekben eltérhet alkalmazása mint például a hitelesítésnél. A metódus itt Get, valamint mivel itt a bejelentkezéskor kiosztott token-t ellenőrizzük az authorization részhez be kell illeszteni az említett tokent, ellenkező esetben 500-as error-t kapunk:

Sikertelen hitelesítés, valamint a sikeres hitelesítés alattunk

Sikeres hitelesítés

## Frontend és alkalmazása:

Ezen részünkön [Vue.js](https://vuejs.org)-t használtunk, melyet be is mutatunk a továbbiakban.

***Mi az a Vue.js:***

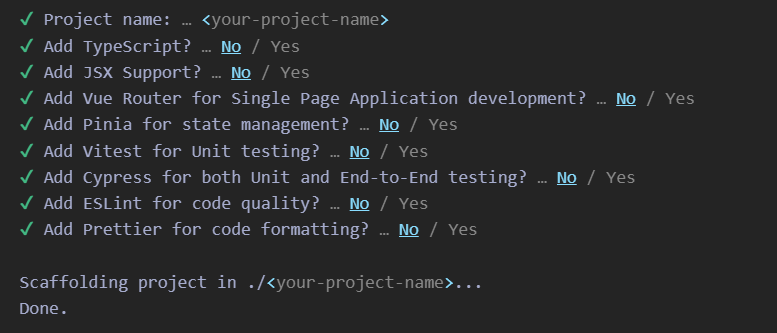
A Vue.JS egy szintén egy nyilt forrású model-view-viewmodel tervezést alkalmazó JavaScript framework melyet interfészek és egyszerűbb aplikációk készítésére használunk. Maguk a Vue komponensek “kibővítik” az alap HTML elemeket hogy azokat egysébe (vagy egységekbe) zárja. (Enkapszuláció)

Magasabb szinteken a komponensek már személyreszabható elemek amikhez rendelhetünk vagy fűzhetünk müködést. Része egy HTML-apalú template szintaxis aminek segítsévégel hozzáfűzhetjük a Vue adataihoz a dokumentum obejktum modelt.(DOM)

Továbbá Vue biztosít nekünk egy interfészt ami megváltoztatja mi van megjelenítve az érintett weboldalon és a hozzátartozó url-en, függetlenül attól hogyan volt az változtatva(frissítés, link, beágyazott link).

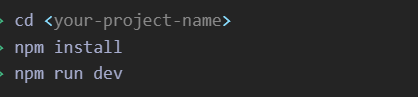
***Telepítési útmutató a Vue.Js-hez:***

Szükségünk lesz egy mappára ahol a projektünk lesz, egy naprakész Node.js -re valamint egy terminálra. A következő paranccsal “npm init vue@latest” promptokból kiválaszthatjuk miket szeretnék hogy projektünk tartalmazzon mivel később ezeket alkalmaznánk.



Vue.js applikációnk telepítése

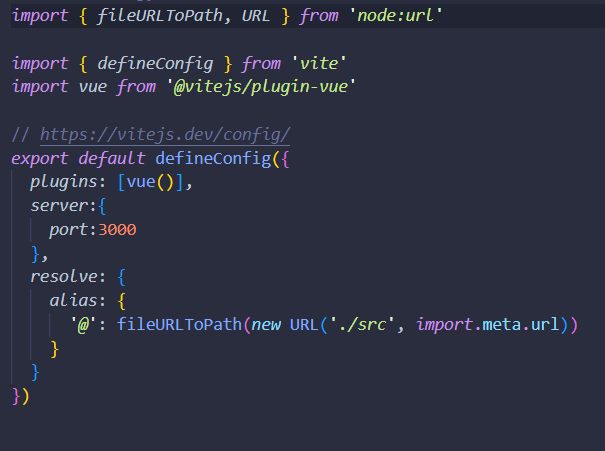
Miután projektünk elkészült a következők és egyben utolsó lépésekkel befejezhetjük a telepítést. Ezután amikor csak elindítanánk az npm run dev parancsal indíthatjuk Vue.js applikációnkat.



Utolsó lépések

**Hogyan fog kommunikálni a frontend és a backend?**

Ami itt zajlik az nem egészen hívható kommunikációnak. Mi esetünkben a backend a 8080 porton fut, frontend esetében ez 3000 port. A backend nem fog felkéréseket intézni a frontend felé, csak a frontend küld felkérést a backendnek amiket ő feldolgoz és egy JSON formátumú adathalmazzal tér vissza. Vue.js-ben portért a vite.config felel.



A használatban lévő port megadása a Frontend számára

**Navigációnk a frontenden:**

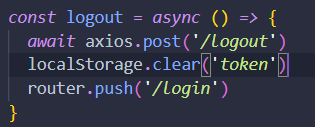
Az app.vue fájlunk felel a navigációnkért, és itt található a kijelentkezés gomb abban az esetben ha már rendelkezik felhasználónk tokennel. A navbarhoz szükséges használnunk router linke module-t amit már kezdésnél importáltunk. Használunk itt async függvényt a kijelentkezésnél ezért az axios module-t is importáltuk.



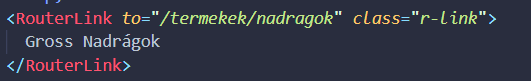
Modulok importálása

**Bejelentkezés a LoginView oldalunkon**

A login függvény korábbanemlítetten async az “await “ jelző használatával képes megvárni a szükséges adatokat a (‘/logout’) útvonalról, majd a token-t törli a localStorage-ból és router.push() segítségével átirányít a kívánt oldalra.



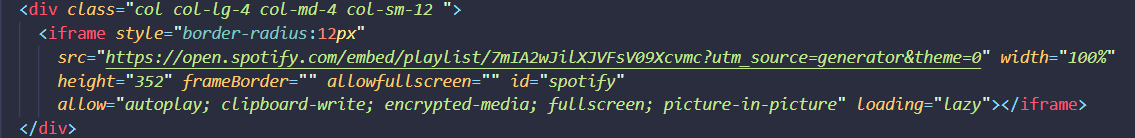
Login függvény frontend részen



RouterLink példa, mely a megadott útvonalon ( ”/termekek/nadragok”) a Nadrágokhoz vezet

**Főoldalunk, HomeView**

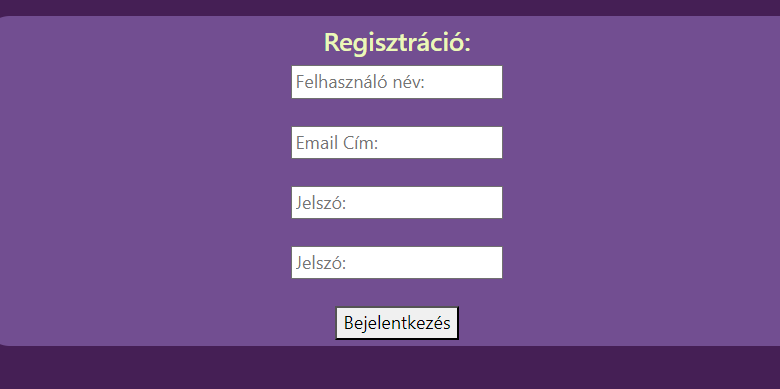
Főoldalunkon nincs szükség authorizációra, bejelentkezés nélkül megtekinthető az itt lévő tartalom. Navigáció mint korábban említettük az oldal tetején található. Amíg vásárlásra vagy kosár megtekintésére nem kerül sor nincs szükség bejelentkezésre. Itt található az előadók képe hisz őket is represzentáljuk bizonyos formában, továbbá információs rész a termékekről és egy Spotify ablak ahol előadóink zenéjét hallgathatják meg. A termék infó csupán egy külön div amiben bármely tetszőleges infót megjeleníthetünk a vevők számára. A spotifyi kivitelezéséhez iFramet használtunk, mely betölti az open.spotify-t az erre alkalmazott ablakba.

iframen belüli spotify link, valamint az ehhez tartozó formázások

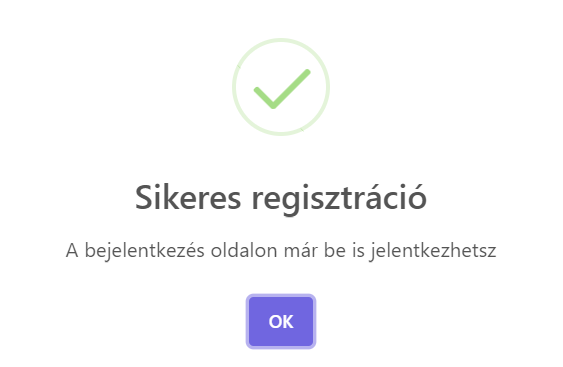
**Regisztrációs felület, RegisztView**

Ezen a felületen biztosítottuk a leendő felhasználók számára a regisztrációs folyamatot. Kezelése könnyű és az oldal maga is végletekig letisztult.

Sikeres regisztráció esetén az új felhasználó adatai egyből a rendszerünkbe kerül és az átirányítás után be is jelentkezhet.



Regisztrációs felület

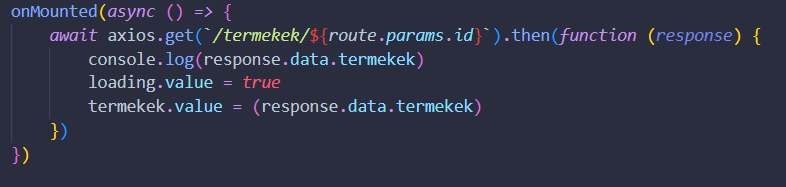


Sikeres regisztráció

**Termékek és kiválasztott termékek**

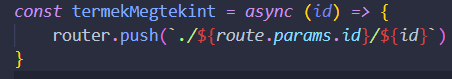
Attól függően hogy milyen termékeket szeretnénk megtekinteni a navigációs sávon keresztül tehetjük meg. Kategórián belül elérhető és adatbázisban lévő termékek sorakoznak elöttünk, rövid és lényeges leírással, valmaint méretük szerinti növekvő sorrendben. A termék megtekintése gombra való kattintáskor megtekinthetjük az adott terméket és tetszés szerint a kosárba is helyezhetjük. A több termék megjelenítéséért kategórián belül a következő felel:

Onmounted hookal hívunk meg egy függvényt az oldal betöltésekor. A függvény lekéri a termékek útvonal tartalmát majd az idetartozó adathalmazt használjuk a továbbiakban.

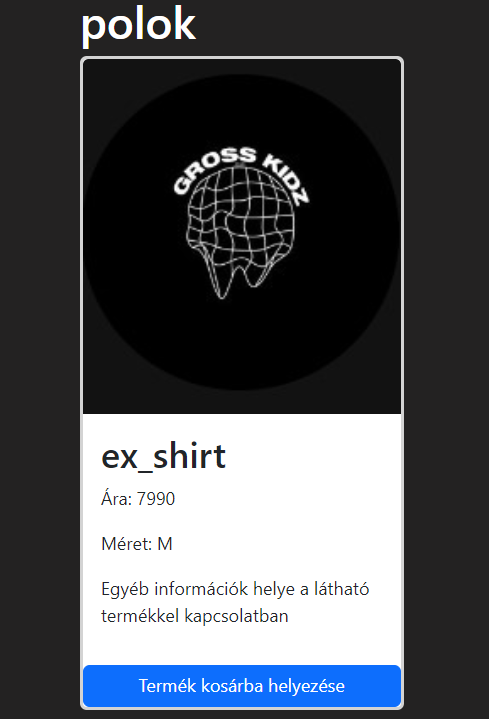


A kategória alapján történő lista feltöltése

Abban az esetben ha szeretnék egy adott terméket megtekinteni másképp járunk el.Itt a route.params melyel átírányitunk egy adott irányba id taggal bővül a végén:

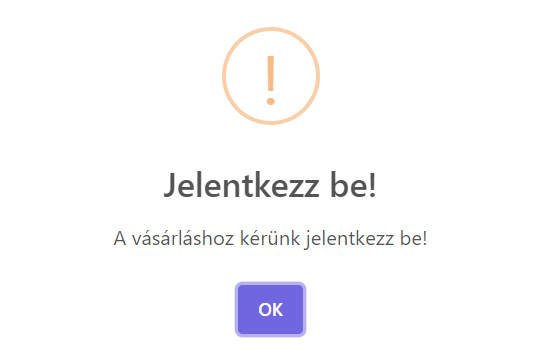


Termékek egyenkénti megtekintéséhez tartozó kód

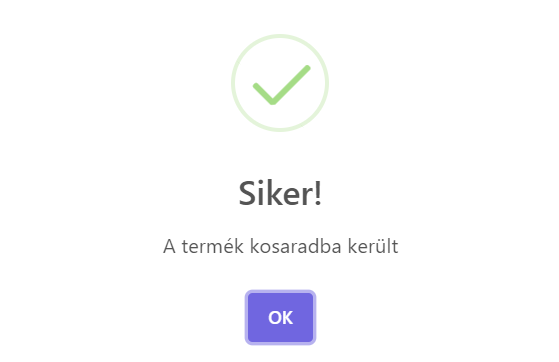


Tetszőlegesen választott termék megtekintése

Ezen a ponton túl ha már megtörtént korábban a bejelentkezés a termékünk a kosárba kerül amiről itt is egy SweetAlert értesít, ellenkező esetben ha szükséges a bejelentkezés mert kimarad erről is tudatosulunk.Természetesen gondokodtunk arról hogy a kosárban bekerült termékek ne tünjenek el az oldal elhagyása vagy frissítése után.



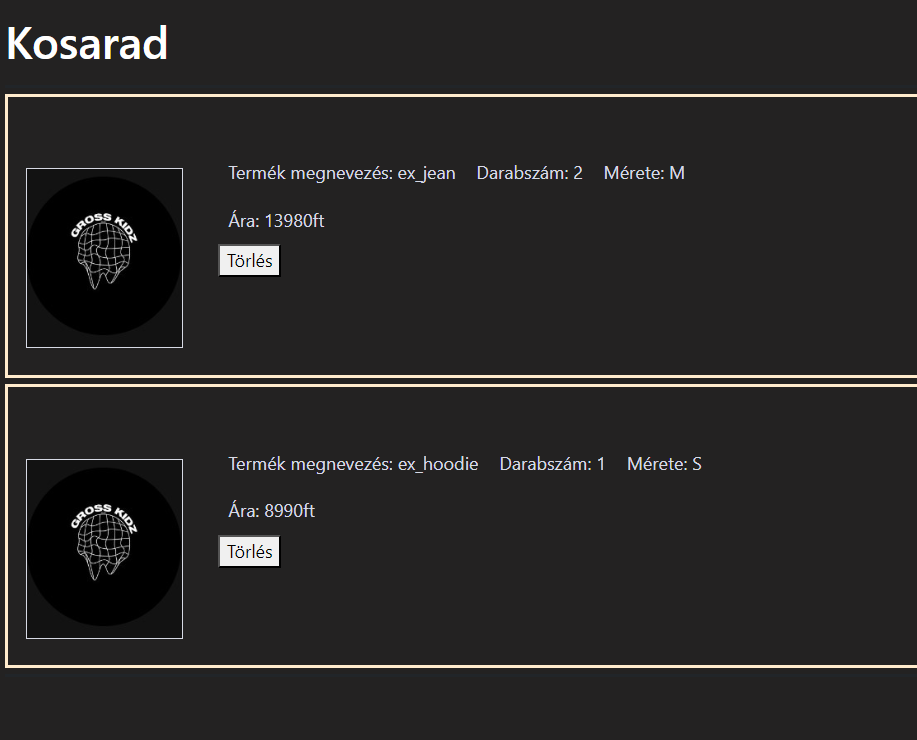
Bejelentkezés nélkül nem tudunk terméket a kosrába helyezni



Sikeres kosárbahelyzéshez tartozó visszajelzés

**KosárView**

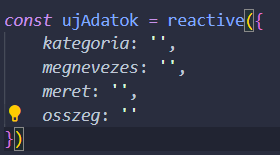
Itt bonyolíthatjuk le a vásárlást. Korábban már kitértünk a számla készítésére most pedig a kosáron a sor. Megjelenésben itt is törekedtünk arra hogy minél egyszerűbb és letisztultabb képet adhassunk. Kosarunkban láthatjuk a termék megnevezését darabszámát, árát(alapár\*darab), méretét valamint a lehetőséget törlésre aminél csökkentjük a termék mennyiségét míg el nem tűnik a kosárból.



Kosár tartalma

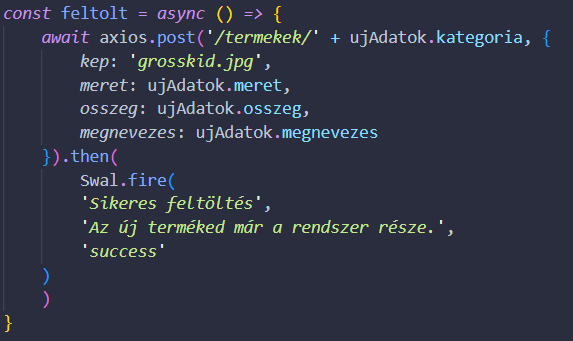
**Admin felületünk:**

Korábban mutattuk miképp látható az admin felület. A felületet új termékek felhelyezésére szántuk hogy ne kelljen adatbázisból feltölteni, ha nagyon muszáj akkor csak törölni.Az adatok from-ból történő megadásához szükségünk lesz egy reactive objektumot, majd a feltölteni kívánt adatokhoz tartozó kulcsértékpárokat.



Reactiv objektum kulcsérték párjainak létrehozása

Feltöltést is külön kezeljük.Ehhez is async függvényre van szükségünk melyen belül Post metódust alkalmazunk. Lényege hogy a kategória részhez a megadott útvonalra történik az összekötés, majd beállítjuk az érték párokat is.Vegül sikeres feltöltés esetén SweetAlert-el éretsülünk a sikerről.



A feltölsét kezelő függvényünk.

A termékhez tartozó adatonként(megnevezés,fajta stb) társul egy egy form. Például kategória a kategóriához, megnevezés a megnevezéshez.



Példa a formhoz amivel az adatokat beállítjuk

Sikeres feltöltés után átnézhetünk az adatbázisba hogy leelnőrizzünk vagy ott végezzünk összetetteb munkát, a linket a jobb oldali jegyzeteknél tekinthetjük meg.

**Axios.js ami megkönnyíti a feladatunkat:**

Ebben a fájlban elsőként beimportáljuk az axios csomagunkat. Az axios.default.. segítségével implementáljuk az urlt amit használni fogunk hogy késöbb ne kelljen újra és újra megtenni.

*Példa:* axios.defaults.baseURL = <http://localhost:8080/>

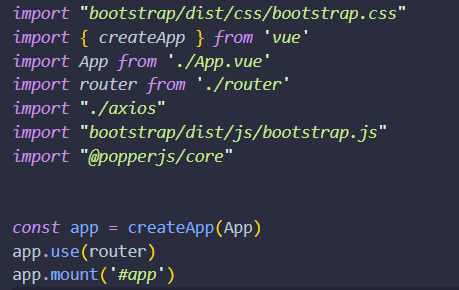
Itt még sort kerítünk az authorizáció megadására is hogy ne kelljen ismételten újra megadni. Ez a sor lehetővé teszi hogy a headers-ben utazzon a token amit localStorage-ból kapunk meg.

*Példa:*

axios.defaults.headers.common['Authorization'] = JSON.parse(localStorage.getItem('token'))

**Main.js:**

Ezt a fájlunkat hívhatjuk annak ami a teljes frontendünket köti össze, ő felel a működtetésért. Azokat a modulokat amikre szükség lesz itt importáljuk, majd Az inde.html-ben található app-divbe beküldjük.



Main.js tartalma

**Router feladata, és felépítése:**

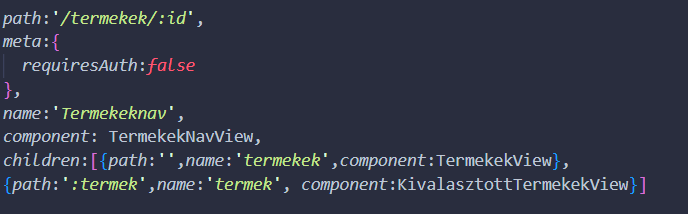
Ő a mi “forgalom irányítónk”. Itt kezdésként beimportáljuk azokat az oldalakat amikhez az útvonalakat szeretnénk hozzárendelni, megadhatjuk hogy van szükség authorizációra. Az érintett oldalakon kívűl még beimportáljuk az axios modulunkat is hiszen szükségünk lesz egy külön auth függvényre és a routerek igénylik.

Minden modul importálása.

A createRouter() segítségével létrehozhatjuk az útvonalakat, részei:

* Path : hivatkozhatunk arra hogy a komponens milyen útvonalon történjen.
* Meta : mi esetünkben itt az auth szükségét állíthatjukbe true/false értékekkel.
* Name : elnevezés
* Component: a beimportált oldalak közül köthetjük ide azt amelyiket szeretnénk elnevezés alapján

Egyébb elemekkel is kibővíthetjük a metát pl. Örököltetéssel. Itt megadhatjuk hogy mikkel rendelkezzenek a leszármazottak.



**Authentikáció ellenörzése:**

Megvizsgáljuk többek között hogy kapunk-e megfelelő tokent.Abban az esetben ha megkapjuk és rendben van itt lesz szükségünk egy axios.Get metódusra ami a verify útvonalon fut majd. Lekéri a tokent, majd ha megfelel hitelesnek minősíti. Ellenkező esetben az itt használt valid változónk “false” értéket kap, és eltávolítjuk a tokent. Végül a visszatérési értékünk vagy a valid változó vagy hamis.



Authentikáció ellenőrzése