**Záródolgozat feladatkiírás**

Tanuló(k) neve[[1]](#footnote-1): Tamás Róbert, Rabóczki Erik, Gonda Szabolcs Krisztián

Képzés: nappali

Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

**A záródolgozat címe:**

**„NASAPC” WebShopProject**

Konzulens: Horváth Norbert

Beadási határidő: 2022. 04. 29.

Győr, 2022. 04. 28

**Módos Gábor**igazgató

**Konzultációs lap[[2]](#footnote-2)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | A konzultáció | | Konzulens aláírása |
| ideje | témája |
| 1. | 2022.02.15. | Témaválasztás és specifikáció |  |
| 2. | 2022.03.14. | Záródolgozat készültségi fokának értékelése |  |
| 3. | 2022.04.17. | Dokumentáció véglegesítése |  |

**Tulajdonosi nyilatkozat**

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2022. április 15.

tanuló aláírása

tanuló aláírása

tanuló aláírása

Tartalomjegyzék

[1. Bevezetés: 5](#_Toc102116576)

[2. Specifikáció: 6](#_Toc102116577)

[3. Adatbázis 7](#_Toc102116578)

[4. JavaScript 9](#_Toc102116579)

[4.1 A JavaScript története 10](#_Toc102116580)

[4.2 Működése 10](#_Toc102116581)

[4.3 Mire képes a JavaScript? 11](#_Toc102116582)

[5. Node.js 12](#_Toc102116583)

[5.1 Célja 12](#_Toc102116584)

[5.2 Modulok 13](#_Toc102116585)

[5.3 npm 13](#_Toc102116586)

[5.4 Express 14](#_Toc102116587)

[5.4.1 Indoklás 14](#_Toc102116588)

[5.4.2 Előnyök 15](#_Toc102116589)

[5.5 Express funkciók 16](#_Toc102116590)

[5.5.1 Gyorsabb szerveroldali fejlesztés 16](#_Toc102116591)

[5.5.2 Middleware 16](#_Toc102116592)

[5.5.3 útválasztás 16](#_Toc102116593)

[5.5.4 Sablonmotorok 16](#_Toc102116594)

[5.5.5 Hibakeresés 17](#_Toc102116595)

[5.5.6 Összegzés 17](#_Toc102116596)

[6. Frontend 19](#_Toc102116597)

[6.1 Navigation komponens 19](#_Toc102116598)

[6.2 Footer komponens 20](#_Toc102116599)

[6.3 Home page 20](#_Toc102116600)

[6.4 About us 20](#_Toc102116601)

[6.5 Product (user) 21](#_Toc102116602)

[6.6 Singup és Login 22](#_Toc102116603)

[6.7 Wishlist 25](#_Toc102116604)

[6.8 Orders és a Cart 26](#_Toc102116605)

[6.9 Admin products 27](#_Toc102116606)

[6.10 Password change 29](#_Toc102116607)

[7. Backend 30](#_Toc102116608)

[7.1 Bevezetés 30](#_Toc102116609)

[7.2 Belépő fájl 30](#_Toc102116610)

[7.3 Termékek lekérdezése 30](#_Toc102116611)

[7.4 Regisztráció (Register & Verify Account) 31](#_Toc102116612)

[7.5 Bejelentkezés 33](#_Toc102116613)

[7.6 Új jelszó igénylése 34](#_Toc102116614)

[7.7 Kívánság lista (WishList) 35](#_Toc102116615)

[7.8 Megrendelések (Orders) 36](#_Toc102116616)

[7.9 Értékelések (Reviews) 37](#_Toc102116617)

[7.10 Admin 38](#_Toc102116618)

[7.10.1 Termék létrehozása 39](#_Toc102116619)

[7.10.2 Termék módosítása 40](#_Toc102116620)

[7.10.3 Termék törlése 41](#_Toc102116621)

[7.11 Backend teszt 42](#_Toc102116622)

[7.11.1 Bevezetés 43](#_Toc102116623)

[7.11.2 Bejelentkezés 44](#_Toc102116624)

[7.11.3 Munkamenet (Session) 44](#_Toc102116625)

[7.11.4 Termékek / Termék ID alapján 45](#_Toc102116626)

[7.11.5 Kívánság lista (WishList) 46](#_Toc102116627)

[7.11.6 Termék létrehozása 47](#_Toc102116628)

[7.11.7 Termék módosítása 47](#_Toc102116629)

[7.11.8 Termék törlése 48](#_Toc102116630)

[7.11.9 Új jelszó igénylése 48](#_Toc102116631)

[7.11.10 Termék módosítása (Modify product) 49](#_Toc102116632)

[8. Android app 49](#_Toc102116633)

[8.1 Android Alkalmazás 49](#_Toc102116634)

[8.2 Az alkalmazás felépítése: 50](#_Toc102116635)

[9. Kommunikáció a csapattagok között: 54](#_Toc102116636)

[10. Felhasznált források: 55](#_Toc102116637)

# 1. Bevezetés:

Manapság 2022-ben és az elmúlt években egyre nagyobb teret hódítottak az online vásárlási felületek, és az IT szféra fejlődésének köszönhetően egyre jobban arra törekszünk, hogy amit csak lehet, azt valamilyen online felületen végezzük; ez az ügyintézéstől fogva a megbeszélések (meeting) és találkozók megtartásán keresztül egészen a vásárlásig terjedt ki. Ezért döntöttünk úgy, hogy mi sem állunk a változás újtába és a növekvő igénynek megfelelve, mi is ezen a területen szeretnénk létrehozni egy projektet, amely teljesen életképes a ma, azaz 2022 igényeinek, teljes mértékben megfelelő és kielégítő felhasználói élményt nyújt.

A weboldal tárgyát/témáját illetően több lehetőség is megfogalmazódott;

Például:

* Bútorbolt
  + Indoklás: A környezetünkben több ilyen irányultságú cég található, a projektet megpróbáltuk volna náluk értékesíteni.
  + Elhatározás: Végül is úgy döntöttünk, hogy más ötleteket is megvizsgálunk.
* „OnlySelfies” online tartalommegosztó oldal
  + Indoklás: Magas kereslet a hasonló tartalommegosztó oldalak iránt.
  + Elhatározás: Ezt a praktikát etikátlannak találtuk és így az ötletet elvetettük.
* Online kaszinó
  + Indoklás: Az online tartalomgyártók promócióinak köszönhetően, ez az iparág világszerte nagy teret hódított az elmúlt pár év alatt.
  + Elhatározás: A piac telítettsége miatt más ötlet mellett döntöttünk.

# 2. Specifikáció:

Végsősoron egy webáruház megvalósítása mellett döntöttünk. Mivel napjainkban az online vásárlás nagy teret hódított, ezért kézenfekvőnek tűnt, hogy mi is egy webshopot készítsünk. Véleményünk szerint ez a projekt kellően összetett és a modern igényekhez mérten aktuális ahhoz, hogy az elvárásoknak megfeleljen.

A projektünk egy teljes értékű, adatbázissal támogatott webáruház, különválasztott backend és frontend használatával.

A webshop teljesértékű felhasználói élményt nyújt: a vásárló tud böngészni a feltűntetett termékek között, rá tud keresni konkrét termékre, vagy kategóriák szerint szűrni. A kiválasztott termékről tud bővebb információhoz jutni és akár regisztráció nélkül is a kiválasztott terméket kosárban tudja tárolni, továbbá a felhasználó regisztráció nélkül is képes rendelést leadni.

Szükségünk van keretrendszerekre, illetve azok alkalmazására, hiszen ezeknek a segítségével, jóval gyorsabban dolgozhatunk, gyorsabban valósíthatjuk meg az adott funkcionalitásokat. A keretrendszer lényege az, hogy a különböző alkalmazásokban legnagyobb gyakorisággal használt elemeket egy helyre csoportosítja, gyakorlatilag készen kínálva ezeket a fejlesztők számára. A programozók így nagyon sok elvégzendő munkától szabadulnak meg. A keretrendszerek nagymértékben könnyítik az egyes tipikus feladatok elvégzését. A komplexebb folyamatokat leegyszerűsíti, tiszta és újra felhasználható kódot írhatunk velük. Megkönnyíti a hibakeresést és a tesztelést, és így rugalmasabb alkalmazásokat eredményez.

Mindezek következtében úgy döntöttünk, hogy a backend-et Express.js, a frontend-et Vue.js segítségével írjuk meg. Adatbázishoz NoSQL-t használunk, amit MongoDB-vel fogunk kezelni.

# 3. Adatbázis

Ami az adatbázis típusát illeti, szintén két lehetőség közül választottunk: a MySQL és a NoSQL között.

Az adatstruktúrák jelentősen különböznek, de mind a két megoldás teljesen átlátható, tehát ez nemvolt szempont. A NoSQL az adatokat BSON formátumban tárolja majd küldés előtt alakítja át JSON-be, ez miatt az adatforgalom gyorsabb, bár ez az előny leginkább nagyobb adathalmazoknál mutatkozik meg, a mi projektünknél ez az előny elhanyagolható.

Végsősoron a könnyebb kezelhetőség és átláthatóság és az miatt, hogy talán a jövőben a NoSQL-ben szerzett rutinnak majd nagy hasznát vehetjük, elégnek bizonyult ahhoz, hogy ezt a megvalósítási módot válasszuk.

Annak érdekében, hogy a fejlesztés során az adatbázisunkkal a lehető legátláthatóbban tudjunk dolgozni egy mongoose nevű npm csomagot telepítettünk a backenden, és a mongoose metódusaival dolgoztunk.

Szintén hasznos eszköz a mongodb compass is, ez szoftver nagyon egyszerűen egyszerű GUI felületet biztosít az adatbázisunkkal kapcsolatos egyszerűbb műveletek végrehajtásához.

A képen szöveg látható

Automatikusan generált leírás

Ahogyan az a fenti mongodb compass GUI felületéről készült képen is látszik, mindössze három kollekcióra volt szükség.

Külön kollekcióban tároljuk a felhasználókat és azok adatait, külön a termékeket és azok adatait és végül a rendeléseket szintén a hozzájuk tartozó adatokkal.

Alapvetően elmondhatjuk, hogy a NoSQL-ben az egyes rekordok milyensége, ez alatt értem a rekordok tulajdonságait többek között, sokkal kötetlenebb, mint a MySQL-ben. De annak érdekében az rekordoknak (például egy bizonyos termék az egy rekord) tudjunk egy kötött adatstruktúrát adni úgynevezett schemákat alkalmaztunk.

A schéme gyakorlatilag egy séma, megadhatjuk benne, hogy az egyes kollekcióban lévő adatok milyen tulajdonságokkal rendelkezzenek, továbbá az egyes tulajdonságokkal szemben is több lehetőséget biztosít. Ilyen lehetőség lehet például az, hogy a schemán bellül tudjunk rögtön validálni az adatot.

A képen szöveg látható

Automatikusan generált leírás

Ebben az esetben például az egyes termékek discount tulajdonságát tudjuk befolyásolni a következő módokon:

Csak számot fogad, 0 alapértelmezett értékkel, a megadása kötelező és csakis 0-100 -ig terjedhet az értéke.

De nem is feltétlenül csak integer, boolean, string stb typusú értékeket adhatunk meg, akár egy egész tömb is szolgálhat tulajdonságként.

Ilyen megoldást alkalmaztunk a rendelések(orders) és a kívánságlista(wishlist) rögzítése érdekében is.

A képen szöveg látható

Automatikusan generált leírás

Tehát a NoSQL használatával nincs szükség összekötő táblák alkalmazására, a kód és a táblák mennyisége is csökken a könnyebb átláthatóság, ugyanazon adatok könnyebb tárolása és a lekérdezések sebességének növelése érdekében.

Az egyes kollekciókhoz írt scemákat úgynevezett modellekben tároljuk, értelemszerűen a backend is három modellt tartalmaz hiszen három schemánk van, és az ezekkel a modellekkel való további interakciót, a modell beimportálásával rögtön lehetővé tudjuk tenni hisz a mongoose által nyújtott beépített metódusok azonnal használhatóak.

# 4. JavaScript

A JavaScript (röviden JS) egy programozási nyelv, amelyet kifejezetten az internetre fejlesztettek ki. A legtöbb webböngésző szoftver, és a modern okostelefonok is mind támogatják a JavaScriptet.

A JavaScriptet első sorban arra használják, hogy gazdagabb, felhasználóbarát élményeket teremtsenek vele az internetet böngészők számára, például dinamikusan frissülő weboldalakat, intuitív felhasználói felületeket, menüket, párbeszédpaneleket, 2D-s és 3D-s grafikákat, interaktív térképeket, videólejátszókat, és számos egyéb elemet, illetve funkciót. A JavaScript ilyesfajta, webböngészőkben történő alkalmazását kliensoldalú JavaScriptnek szokás nevezni.

A JavaScript az internetes háromszög egyik eleme, a másik kettő pedig a CSS, illetve a HTML. A HTML feladata a weboldal leírása (szövegek, grafikák stb.), a CSS pedig a weboldal megjelenéséért felel. A JavaScript nem esszenciális, de a fent említett hasznos funkciók miatt nagyon fontos eleme a webdizájnak, hiszen dinamikus weboldalakat lehet létrehozni vele, és lehetővé teszi a programozható elemek elhelyezését is ezeken a weboldalakon.

## 4.1 A JavaScript története

A JavaScript fejlesztését 1995-ben kezdte el a Netscape Communications, az akkoriban népszerű Netscape böngésző készítője. A céljuk egy jobb felhasználói élmény megalkotása volt egy „ragasztó nyelv” segítségével. Brendan Eich-et kérték fel arra, hogy beágyazza a Scheme programnyelvet. Azonban mivel akkoriban a Java volt az új, felkapott programnyelv, úgy döntöttek, hogy az új nyelv szintaxisát inkább ahhoz hasonlóra készítik majd el – így született meg a JavaScript, amely egy nyelvben ötvözte a Scheme funkcióit, a SMallTalk objektumorientáltságát, és a Java szintaktikáját.

## 4.2 Működése

Miután egy webböngésző betölt egy weboldalt, és átvizsgálja a HTML kódját, elkészít egy úgynevezett dokumentum objektum modellt (DOM) a weboldal tartalmairól. A DOM egy élő képet mutat a JavaScript kódnak a weboldalról, és a kód módosításokat végezhet a DOM-on, mielőtt az megjelenne a felhasználónak. Mindez maximum néhány másodperc alatt történik.

A böngésző továbbá lehetővé teszi, hogy a kód felhasználói műveletekre aktiválódjon, például egérmozgásokra, gombkattintásokra, stb. Mindez rengeteg lehetőséget ad egy JavaScript programozó kezébe, aki bármilyen kisebb vagy nagyobb alkalmazást készíthet különböző feladatok elvégzésére.

Mint említettük, amikor egy böngésző szoftver betölt egy webhelyet, a HTML vizsgáló kielemzi a HTML kódot, és elkezdi felépíteni a dokumentum objektum modellt. Amikor a vizsgáló CSS vagy JavaScript kódot talál (legyen az dokumentumon belüli, vagy kívülről betöltött), átnyújtja azt a CSS vagy JavaScript elemző motornak. A JavaScript motor betölti a külső JavaScript fájlokat vagy az inline JS kódot, de még nem futtatja le őket azonnal – előbb megvárja, hogy a HTML és a CSS elemzés befejeződjenek.

Miután az elemzés kész, a JavaScript abban a sorrendben fut le, amilyen sorrendben az elemző találkozott vele az oldalon – definiálásra kerülnek a változók és a függvények, a függvények meghívódnak, aktiválódnak az eseménykezelők, és így tovább. Ezek mind frissítik a HTML elemző által felállított DOM-ot, és végül a weboldal megjelenik a böngészőben.

## 4.3 Mire képes a JavaScript?

A JavaScript egy komplett programnyelv-fordító, amely közvetlenül a webböngésző szoftverekben működik. A JavaScripttel mindent meg lehet oldani, amit a Java-val, vagy bármilyen hasonló, általános programozási nyelvvel, például:

* Deklarálhatunk vele változókat
* Értékeket tárolhatunk és hívhatunk meg
* Függvényeket definiálhatunk és használhatunk
* Saját osztályokat hozhatunk létre
* Külső modulokat használhatunk

A JavaScript egy olyan sokoldalú programozási nyelv, hogy még vírusokat és rosszindulatú alkalmazásokat is lehet írni benne – sajnos rengeteg hacker ki is használja ezt az óvatlan felhasználók ellen. Ez lehetővé teszi a támadóknak, hogy böngésző sütiket, jelszavakat, bankkártya-adatokat és egyéb személyes adatokat lopjanak a felhasználóktól, vagy akár konkrét vírusokat telepítsenek a számítógépiekre. Éppen ezért fontos volt a weboldalunk számára, hogy egy funkcióban gazdag, de biztonságos felületet biztosítsunk a felhasználó(k)-nak.

# 5. Node.js

A Node.js (vagy másnéven Node) egy rendszer, melyben JavaScript-ben írhatunk szerver oldali alkalmazásokat, tehát a böngészőn kívül. Maga a rendszer C/C++-ban íródott, és egy esemény alapú I/O rendszert takar a Google V8 JavaScript motorja felett. Az egyik lényeges különbség például a PHP-hoz képest azon kívül, hogy JavaScript-ben írjuk az alkalmazásunkat, hogy nincsen szükségünk HTTP szerverre (mint amilyen az Apache httpd), mivel mi magunk írjuk a HTTP szerverünket.

## 5.1 Célja

A Node célja, hogy egy egyszerű felületet adjon arra, hogy skálázható hálózati alkalmazásokat írjunk. Azt tudjuk, hogy a JavaScript a világon a jelenleg egyik legnépszerűbb programozási nyelv, többek között ezért esett erre a nyelvre a választás. A JavaScript nyelvnek vannak más sajátosságai, amelyek szintén előtérbe kerülnek a rendszer használata során. Ahogy a böngészőben, úgy a Node alatt is egyetlen esemény hurokban (event loop) fut a program, és ez adja az egyik legfontosabb előnyt.

A Node.js-t úgy írták meg, hogy (szinte) minden esemény aszinkron legyen, ezért a program sosem blokkolódik, azaz nem kell várni, hogy egy művelet befejeződjön, vele párhuzamosan futtathatunk további műveleteket. Ez pontosan ugyan úgy működik, mint a böngészőben levő XHR kérések vagy más események, mint például a click, mouseover. Ez az alkalmazás folyamatosabb futását, több alkalmazás párhuzamosítását, valamint az egész rendszer optimálisabb működését teszi lehetővé.

## 5.2 Modulok

A Node.js alapértelmezetten is rengeteg modullal érkezik. A modulok a CommonJS specifikáció szerint íródtak, és ennek megfelelően a require('modulneve'); paranccsal tudjuk behúzni őket az alkalmazásunkba. Ezek egy objektumot tesznek elérhetővé, és egy változóhoz rendelve később kényelmesen tudjuk használni is őket. Az alap telepítésben elérhető modulokról a Node hivatalos API dokumentációjában olvashatunk.

## 5.3 npm

A Node hivatalos csomagkezelője az npm, amellyel kényelmesen listázhatjuk, telepíthetjük, frissíthetjük és eltávolíthatjuk az elérhető modulokat, valamint mi magunk is publikálhatjuk az általunk fejlesztett modulokat az npm rendszerébe. Az npm azaz Node Package Manager - vagy Node Csomag Kezelő - node programok telepítésére és karbantartására használható. A rendszerben jelenleg több százezer csomag van és naponta több milliót installálnak programozók. Ma már az npm a node szerves része, úgyhogy nem kell külön installálni.

**Csomagok helyi (local) telepítése:**

Egy csomagot a helyi mappába a nagyon egyszerű npm install <csomag\_neve> paranccsal lehet installálni.

Például: npm install lodash.

A parancs lefuttatása után a node létrehoz (ha még nincs) egy node\_modules mappát és ha kilistázzuk a tartalmát (ls node\_modules) akkor láthatjuk, hogy megjelent a lodash könyvtár.

**Csomagok globális telepítése:**

Egy csomagot kétféle képpen lehet telepíteni, az egyik a már fentebb említett helyi telepítés, a másik pedig a globális. Globálisan olyan csomagokat szokás telepíteni, amiket többnyire terminálból akarunk futtatni (pl: grunt, jshint). Maga a parancs nagyon hasonló az előzőhöz: npm install -g jshint.

## 5.4 Express

Az Express egy ingyenes és nyílt forráskódú webalkalmazás keretrendszer a Node.js számára. A webes alkalmazások gyors és egyszerű tervezésére és építésére használják. Az Express a Node keretrendszere, ami azt jelenti, hogy a kód nagy része már előre meg van írva a programozók számára. Az Express segítségével egyetlen oldal-, többoldalas vagy hibrid webalkalmazást is létrehozhatunk.

Az Express JavaScript könyvtára segít a programozóknak hatékony és gyors webalkalmazások létrehozásában. Az Express meglehetősen leegyszerűsíti a Node használatát, megkímélve a programozókat, azáltal, hogy magába foglal olyan komponenseket, melyek a fejlesztés során gyakran használtak.

### 5.4.1 Indoklás

A legértékesebb eszköz minden vállalkozásban az idő. A döntő többségének a programozónak meg kell küzdeniük azért, hogy rövid időn belül hatékony webalkalmazásokat hozzanak létre. De a webalkalmazások kódolása és tesztelése rendkívül időigényes. Ez az, ahol az Express.js életmentővé válik a programozók számára.

Az Express felére csökkentheti a kódolási időt, de ettől függetlenül gyors, tiszta és jó alkalmazást fejleszthetünk. Nemcsak csökkenti az időt, hanem csökkenti a webalkalmazások létrehozásához szükséges erőfeszítéseket a különböző funkciók segítségével, amely segíthet a mentális egészség megőrzésében.

Az Express használatának másik oka a JavaScript. Az Express.js lehetővé teszi még a kezdők számára is, hogy belépjenek a webes alkalmazásfejlesztés világába, mert támogatja a JavaScriptet. A JavaScript nagyon könnyen megtanulható bárki számára, még akkor is, ha nem rendelkezik előzetes ismeretekkel más nyelvekről.

Minden vállalkozás másik fontos eszköze a pénz. Fontos, hogy a pénzt hatékonyan használjuk a nyereség maximalizálása érdekében. Mivel az Express.js egy nyílt forráskódú és ingyenes rendszer, amely számos nagyszerű funkciót kínál, nincs különösebb ok arra, hogy ne használjuk.

### 5.4.2 Előnyök

* Gyorssá és egyszerűvé teszi a webalkalmazások fejlesztését.
* Könnyen konfigurálható és testre szabható.
* Különböző köztes szoftvermodulokat tartalmaz, amelyekkel kérés és válasz metódusok segítségével további feladatok hajthatóak végre.
* Könnyen integrálhatóak a különböző sablonmotorok, mint például Jade, Vash, EJS, Pug.
* Lehetővé teszi a Middleware (köztes szoftver) használatát.
* Lehetővé teszi REST API-kiszolgáló létrehozását.
* Könnyen csatlakoztatható adatbázisokhoz, mint például a MongoDB, Redis, MySQL

## 5.5 Express funkciók

### 5.5.1 Gyorsabb szerveroldali fejlesztés

Az Express a Node számos általánosan használt funkcióját biztosítja olyan funkciók formájában, amelyek könnyen használhatók bárhol a programban. Ez megszünteti a több órás kódolás szükségességét, és így időt takaríthatunk meg. A gyorsabb fejlesztés következtében, egyszerűbb nehezebb feladatokat ellátni rövid idő alatt, ami természetesen azt is jelenti, hogy magasabb bevétel várható. Le egyszerűsítve; pénzt és időt spórolhatunk meg.

### 5.5.2 Middleware

A Middleware (köztes szoftver) a program olyan része, amely hozzáfér az adatbázishoz, a klienskérésekhez és a többi köztes szoftverhez. Elsősorban az Express különböző funkcióinak szisztematikus szervezéséért felelős.

### 5.5.3 útválasztás

Az Express.js egy rendkívül fejlett útválasztási mechanizmust biztosít, hasonlóképpen a Laravelhez.

### 5.5.4 Sablonmotorok

Az Express.js olyan sablonmotorokat biztosít, amelyek lehetővé teszik a fejlesztők számára, hogy dinamikus tartalmat építsenek a weboldalakon HTML-sablonok létrehozásával a szerveroldalon. Ilyen sablon lehet például az EJS vagy a Pug.

### 5.5.5 Hibakeresés

A hibakeresés elengedhetetlen a webes alkalmazások sikeres fejlesztéséhez. Az Express.js megkönnyíti a hibakeresést azáltal, hogy hibakeresési mechanizmust biztosít, amely képes pontosan meghatározni az alkalmazásban a pontos részt, amely hibákat tartalmaz.

### 5.5.6 Összegzés

Az Express egy ingyenes és nyílt forráskódú webalkalmazás keretrendszer a Node.js számára. A webes alkalmazások gyors és egyszerű tervezésére és építésére használják. Az Express a Node keretrendszere, ami azt jelenti, hogy a kód nagy része már előre meg van írva a programozók számára. Az Express segítségével egyetlen oldal-, többoldalas vagy hibrid webalkalmazást is létrehozhatunk.

Az Express egy elterjed, és sikeres rendszert mert:

* Könnyű megtanulni az alapokat. Sok frontend fejlesztő már ismeri a JavaScriptet, tehát nem kell új nyelvet tanulniuk ahhoz, hogy megtanulják az Express-t.
* Sokkal könnyebbé teszi a backend fejlesztést, sok időt és pénzt spórolhatunk meg.
* Sablonmotorok segítségével frontendet is könnyen létrehozhatunk.
* Az Express egyszerűen testre szabható, hogy megfeleljen a fejlesztési igényeknek.
* Az Express biztosít köztes szoftverrendszert, így könnyebben szétválasztható a kód, könnyebben olvasható, jobban érhető lesz. Így a fejlesztés is gyorsabban, hatékonyabban haladhat.

A frontend terén is több lehetőséget vettünk fontolóra:

A Typescript alapú Angular és a Javascript alapú Vue.js lehetőségét mérlegeltük. Egyedül natív kódolással hasonló okok miatt most sem szerettünk volna dolgozni, mindenképpen most is az imént említett lehetőségekből szerettünk volna választani és natív Javascript kódot csak akkor szerettünk volna alkalmazni, amikor olyan dolgot szerettünk volna megvalósítani, amelyet a framework nem tartalmaz.

Az Angular bár napjainkban egy elég elterjedt megoldás, a környezetünkben is elismert cégek programoznak benne, mi személyes preferencia alapján a Vue.js mellett döntöttünk.

Szeretnénk megjegyezni, hogy bár most nem az Angulárra esett a választásunk, de mindenképpen megéri most a közös projekt mellett és a későbbiekben is, nagy hangsúlyt fektetni rá, hiszen a jelenlegi ismereteink alapján elmondhatjuk, hogy sok lehetőség van benne. A TypeScript bár egyesek mondhatják, hogy nehezebb nyelv, mint a JavaScript, mindenképpen megéri a bele fektetett időt, melyet később majd a személyes gyakorló projektek által igenis bele fogunk fektetni.

A frontedet Vue.js-ben írjuk!

A Vue.js egy **nagyon könnyen megtanulható, és használható JavaScript könyvtár**. Alacsony Learning Curve-vel rendelkezik.

A Vue.js-t annak ellenére, hogy könnyebben elsajátítható, rengeteg feladatra könnyedén fel lehet használni. Kifejezetten szeretik startupok is használni, valamint kiváló párosítást alkot [Laravel](https://gremmedia.hu/edukacio/bejegyzes/laravel-keretrendszer-mit-erdemes-tudni-rola-tenyleg-legnepszerubb-keretrendszer)keretrendszerrel.

Igaz most ezt az előnyét nem fogjuk kihasználni.

Szintén pozitívumnak könyveltük el, hogy a Vue.js-ben minden eddiginél könnyebben tudtunk teszt adatokat felvenni olyan struktúrában, amilyenben majd már élesben fogja kapni az adatokat és így a kezdetektől fogva koncentrálni tudtunk egy dinamikus frontend létrehozására.

# 6. Frontend

Mielőtt a frontend fejlesztésének neki láttunk volna, megkellett határoznunk, hogy milyen legyen a dizájn, és hogy milyen komponenseket szeretnénk belerakni a projektbe.

A dizájn és a komponensek meghatározásának az érdekében, egy meeting keretein belül a döntéseinket egy dizájn demo elkészítésével örökítettük meg.

Ez nem a végleges verzió, sokkal inkább egy iránymutató, arról hogy a dizájn terén milyen irányba menjen el a frontend. Ami a dizájn milyenségét illeti több oldalról is inspirációt merítettünk: többek között a tchibo oldaláról.

Már a projekt létrehozásakor tudtuk, hogy a bootstrap külső komponenseket, stílus elemeket és a grid rendszert szeretnénk majd használni ezért ez már a projekt elején telepítésre került. További a Primevue által felkínált listákat és formokat is fontolóra vettük.

## 6.1 Navigation komponens

Bár az imént emlétett források tartalmaznak előre elkészített navigation panelt, úgy döntöttünk, hogy nem vagyunk teljes mértékben megelégedve a felkínált lehetőségekkel ezért itt, egy külső komponens használata helyett, saját komponens megírása mellett döntöttünk.

A komponensel kapcsolatban több alap kikötésünk is volt:

Az első kikötés a reszponzivitás volt, jól kell mutatnia mind mobilon mind nagyobb felbontású képernyőkön. Ezt a kritériát és a tényt, hogy mobilon nézetben nehezebb több információt megjeleníteni anélkül, hogy a felhasználót elárasztanánk üzenetekkel, a weboldal átláthatóságát csökkentenénk és a felhasználói élményt rontanánk, ez a feladat olyan megoldást igényelt, amely lehetővé teszi, hogy a felhasználó kontrollálni tudja a megjelenített információ mennyiségét.

Az imént említett gondolatokat figyelembe véve, felismertük, hogy a felhasználónak nemszükséges minden pillanatban látnia a menüt mobil nézetben, ezért annak és a navigációs menühöz kapcsolódó paneleknek a megjelenítését dinamikussá, láthatóságát egy fix pozíciójú gomb által testre szabhatóvá tettük.

Továbbá megfigyelve több modernebb kialakítású weboldalt, azt tapasztaltuk, hogy az a dizájn melyet a mobileszközökön való adatmegjelenítésnél használnak teljes képernyős nézetben is egyre nagyobb teret hódít, ez a megközelítés tetszett nekünk így hasonló dizájn fejlesztésébe kezdtünk.

Ami a menüpontokat illeti az eredeti elképzelés szerint a weboldal dizájn teljes mértékben dinamikus lett volna és minden adatot adatbázisból kértünk volna le. De végül ebből az elképzelésből engedtünk az adatforgalom csökkentése érdekében.

## 6.2 Footer komponens

A footer manapság egy standard komponensnek számít minden weboldalon, a megléte gyakorlatilag már-már elvárt. Ami a funkcióját illeti elsősorban két célunk volt a létrehozásával. Először is bizonyos információk megjelenítése, többek között a felhasznált technológiák logóinak feltűntetése, copyright jogok feltűntetése és a weboldal logója.

Másodszor pedig némely navigációs link ismételt megjelenítése. Erre azért van szükség, mert a látogatónak a figyelmét nagyobb valószínűséggel tudjuk megragadni, hogyha egyes menüpontok a weboldal tetején és a footer segítségével a weboldal alján is megjelennek, hiszen így a további szolgáltatásainkra például a fő oldal áttekintése után ismét felhívjuk a figyelmet.

## 6.3 Home page

A fejlesztés során több elképzelés is született a fő oldal megvalósításával kapcsolatban. Az alapelv az volt, hogy a fő oldalnak figyelemfelkeltőnek kell lennie, de nem is szabad túl sok információval elárasztania a felhasználót. Az imént említett navigációs panel kialakításának módját figyelembe véve és azt, hogy alapértelmezetten a navigation komponens nem jelenik meg, így, több adatot tudunk megjeleníteni úgy, hogy az még ne legyen zavaró.

A fő oldal elsődleges célja a látogató figyelmének a felkeltése és a további böngészésre való ösztönzés, ezért a fontosabb szolgáltatásaink linkjei a főoldalon akár háromszor is megjelenhetnek különböző formában.

További szempont volt a megfelelő vizuális megjelenítés is. Tehát az információ ízléses tálalása. Az egyik ilyen módszer volt a carousel komponens felhasználása, mely periodikusan egy képlistán megy végig magyarázó szöveg kíséretében. Ez a fajta megjelenítés napjainkban nagyon elterjedt és jól mutat mind típusú kijelzőn, hiszen a komponens reszponzívitás könnyen megoldható. A weboldalunkon a bootstrap carousel külső komponenst használtuk fel.

A forráskód átírásával a komponens dinamikusan a megadott adatok alapján generálódik le.

## 6.4 About us

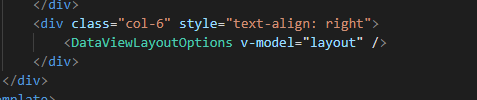
Az about us oldal egy egyszerű oldal ahol magunkról, a projekt céljáról és olyan dolgokról melyek, a látogatónak szükségesek lehetnek, osztunk meg alap információkat. Talán ez a legegyszerűbb oldal a projektben ugyanis, egyszerű kiíratásról van itt szó listákba a pragrafusokba rendezve.

## 6.5 Product (user)

Az oldal elsődleges célja hogy az elérhető termékek közül a látogató tudjon böngészni és egyéb műveleteket végre hajtani. A termékek úgynevezett cardok segítségével vannak feltűntetve. Mind a bootstrapnek és a primevue-nak vannak card moduljai, de ezek nemteljesen feleltek meg az elvárásainknak.

A cardokkal kapcsolatban a következő elvárásaink voltak:

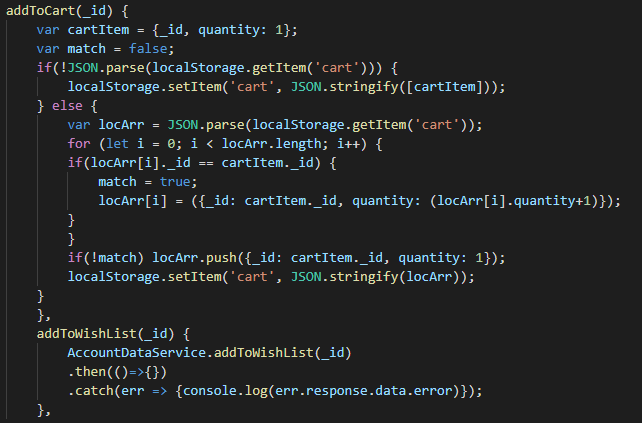
1. A cardnak tartalmazni kell egy lekicsinyített képet a termékről.
2. A termék nevét és árát felkell tudnunk rajta tűntetni.
3. Ha a felhasználó vagy egy gombra kattintva vagy mouseover esemény következtében a cardnak a méretét megkell növelnünk annyira, hogy az utóbbi információk mellett még egy részletesebb leírást is megtudjunk jeleníteni.



A termékek megjelenítésére több megjelenítési mód is rendelkezésre áll.

A products oldalon a felhasználó több műveletet és képes végrehajtani, névlegesen a terméket kosárba tudja helyezni, a választott termékről bővebb információhoz juthat, ehez a művelethez a product-details oldalra való átlépés szükséges, és a lehetőség nyílik a termék kívánságlistára, való felvételére is.

Ezekhez a műveletekhez backend api-hívásokra lesz szükség.

A termék kosárba való felvételéhez, a felhasználónak nemszükséges regisztrálnia és bejelentkeznie, hiszen a kosár tartalmát nem adatbázisban tároljuk, azonban a kívánságlistára való felvételhez alap követelmény hogy a felhasználó be legyen jelentkezve, az api hívás csak így lehetséges, a hívás előtt egy session api hívásával, illetve a visszakapott boolean változó értékével validáció történik. Igaz, már az oldal renderelése előtt lefut egy hasonló validáció, hogy olyan gombokat, menüket és információkat ne osszunk meg melyekre a felhasználónak nincs jogosultsága. Ez azértfontos, mert így eltudjuk kerülni, azt hogy összezavarjuk a felhasználót. 

Hogy nem regisztrált felhasználók is használni tudják a weboldalt, a kosár lokálisan kerül eltárolásra.

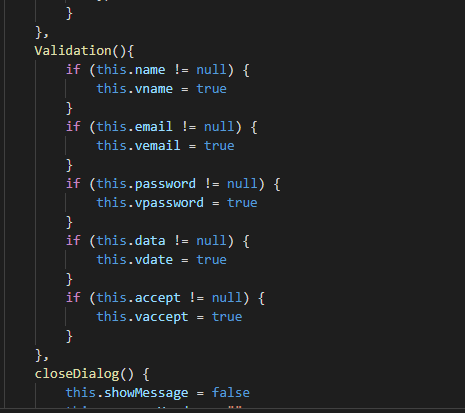
## 6.6 Singup és Login

A signup oldal teszi lehetővé, hogy a felhasználó regisztrálni tudja magát az oldalra, és hogy aztán élvezhesse annak előnyeit.

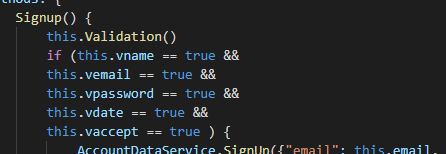
A regisztrációs folyamathoz egy form kitöltése szükséges, melyen több hiba lehetőséget is figyelembe kellet vennünk.

1. A megadott adatokat validálni kell, hogy véletlenül se lehessen hibás adatokat megadni.
2. A felhasználónak további segítséget kell nyújtani az által hogy:
   1. Segítünk megfelelő erősségű jelszót választani a Primevue: password komponens segítségével.
   2. Redukáljuk az elírás lehetőségét azáltal, hogy validációt készítünk a jelszó kétszeres lekérése után.
   3. A form kitültése után validáljuk a megadott adatokat.

Továbbá az kitöltött mezőket adatkötésekkel tároljuk.



A felhasználó nem hagyhat egy mezőt sem üresen



A signup api csakis sikeres validáció után kerül meghívásra.

A validációk sikeressége után, egy api hívás segítségével folytatódik a regisztrációs folyamat. Egy felugró ablak jelzi, hogy a link mellyel az Account aktiválható a megadott emailre lesz elküldve.

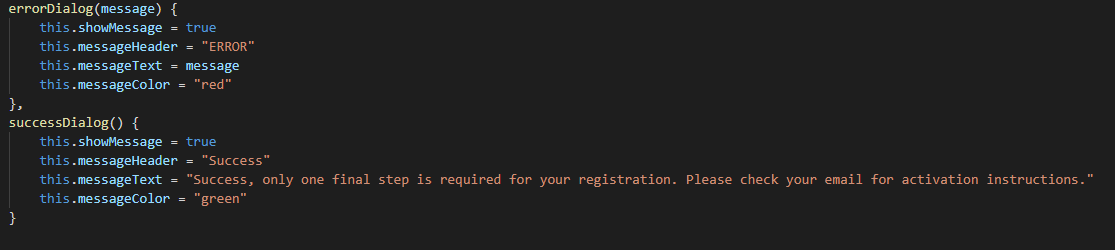
Az emailben elküldött link által való fiók aktiválását követően, a fiók adatbázisban való tárolása megtörténik. Ez úgy valósul meg, hogy az emailben elküldött link, betöltésekor a frontend az URL-ből kinyeri az aktiváláshoz szükséges tokent, melyet felhasználva egy API hívást követően az adatbázisban az account aktiválásra kerül.

Ha ez sikerült egy új felugró ablak tájékoztatja a felhasználót a művelet sikerességéről vagy az esetleges hibákról, majd átirányítja a főoldalra.

A login oldalon hasonló képpen, a validációs műveletek után egy login api hívást követően a frontend átnavigálja a felhasználót a főoldalra, ami egy ismételt session api hívással megállapítja, hogy a kliensoldalon való session valid, így a bejelentkezett userek számára elérhető tartalmak renderelésre kerülnek.

Ilyen tartalmak többek között, a navbar tartalma, és hozzáférés egyes pagek-hez.





Mind a signup mind a login és egyéb oldalakon is úgynevezett Dialog felugró ablakokkal oldottuk meg a felhasználó tájékoztatását a műveletek állásáról.

## 6.7 Wishlist

A wishlist a weboldal azon eszközei közé tartozik, melyeknek előnyeit csak regisztrált felhasználók élvezhetik. Ugyanis ez egy adatbázisban tárolt lista készítésének a lehetőségével ruházza fel a felhasználót. Így az ezen a listán feltűntetett termékek árában történő változásokat, potenciális leértékeléseket sokkal könnyebben nyomon lehet követni.

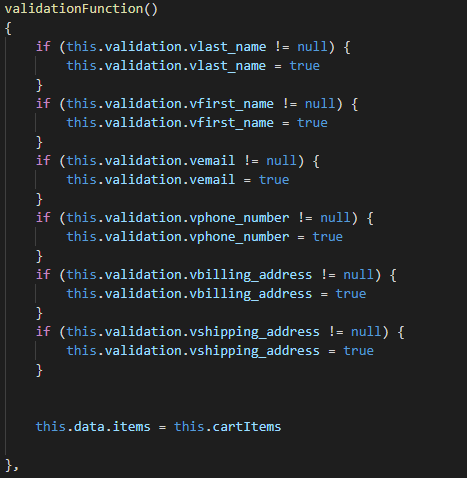
A lista szabadon szerkeszthető, a tulajdonosa szabad hozzáférést kap minden ezzel kapcsolatos tevékenységre.

## 6.8 Orders és a Cart

:

A felhasználó a vásárlás során az „add to cart” gombra kattintva tudja a kiválasztott termékeket a kosárba helyezni. Az hogy egy termék a kosárban van, önmagában nemjelent semmit, a rendelés csak akkor lesz véglegesítve, ha a felhasználó tovább folytatja a vásárlást a megfelelő gombra kattintva. Ha ezt megteszi, akkor a megfelelő adatok rögzítése után (regisztrált felhasználóknak ez a lépés kihagyható, elmentett fizetési adatok esetén) a cart azaz kosár tartalma átkerül az orders azaz rendelések listába. A megrendelések listában lévő termékek már elvannak különítve a vásárlónak és a termékek már nincsenek számon tartva, mint elérhető termékek, úgyis mondhatjuk, hogy ami a többi látogatót illeti, számukra a rendelések listában lévő termék már eladott így raktáron nem elérhető termékként van kezelve.

Az orders listából a megrendelt terméket el lehet távolítani, így azok ismét nem eladott, eladásra szánt terméknek tekinthetők. A sikeres kiszállítás és kifizetés után, az orders lista tartalma törlődik.



Az adatok a rendelés leadása előtt egy validációs folyamaton mennek keresztül.

## 6.9 Admin products

Az admin products célja különbözik a mindenki számára publikus products oldaltól. Ehhez az oldalhoz a hozzáférés csakis validált adminok számára lehetséges.

Az ezen az oldalon használt szerkesztett és teljes mértékben testre szabott primevue modul segítségével az admin hatékonyon tud API hívásokkal hozzáadni, szerkeszteni és törölni termékeket, tehát úgy is mondhatjuk, hogy egy kész CRUD product interface áll a rendelkezésükre a könnyű és hatékony adatbázis kezelés érdekében, mindezt egy oldalon.

A CRUD műveletekkel való munkának minden változatát további felugró panelek könnyítik meg, lehetővé téve programozási előismerettel nem rendelkező kollégák számára is az admin munkakör gyors, kielégítő szintű ellátását.

A felugró ablakokkal, üzenetekkel és interfacekkel az admin folyamatos visszajelzést kap a munkája állapotáról illetve annak sikerességéről. Illetve a kritikus műveletek, mint például a törlés előtt felugró megerősítő ablakok minimalizálják a véletlenül elkövetett törlések és változtatások előfordulását.

Fontos megemlíteni, hogy az admin products interface használata elsősorban számítógépről való munkavégzéshez lett tervezve. Természetesen ez nem azt jelenti, hogy itt a reszponszivitást mint szempontot mellőztük, de egy ekkora táblázatot ésszerűtlen lenne, mobilnézetben megjeleníteni úgy, hogy mellőznénk az X irányú csúszka használatát. Természetesen a csúszkától eltekintve, mint minden más az oldalon ez az oldal is a reszponszivitás alapelvei szerint lett tervezve.

Ez az oldal, mint minden más admin oldal nem elérhető csak adminok számára. Annak érdekében, hogy ez a kritérium megvalósulhasson. Az oldal betöltésekor minden alkalommal a háttérben lefut egy autentikációs algoritmus, ami két dolgot ellenőriz: elsősorban hogy a felhasználó be van e jelentkezve, másodszor pedig hogy a felhasználó rendelkezik e admin jogosultsággal. Ha ezek közül a kritériumok közül bármelyik nem teljesül, akkor az oldal egy kényszerített átirányítással visszalép a főoldalra.

## 6.10 Password change

A jelszó változtatás egy két lépcsős művelet, elsőnek például a login oldalon a jelszó változtatást kérvényezni kell. Ez a megfelelő gombra való kattintással lehetséges, ez után egy felugró ablakban a felhasználónak megkell adni az emailcímét. Fontos hogy azt az email címet adja, meg amivel már előzőleg regisztrált az oldalra.

Hibás adat esetén vagy bármely egyéb probléma esetén az oldal ennek megfelelően informálja a felhasználót. Ha egy érvényes email cím megadása megtörtént, akkor a backend, a felhasználót emailben fogja értesíteni a művelet sikerességéről és egy token és link emailben történő elküldésével a jelszó változtatás következő lépése kezdetét veheti.

A linkre kattintva, a backend ellenőrzi a linkben lévő tokent és a token hitelességének megfelelően, lehetőség nyílik egy új jelszó megadásához. A megfelelő gomb lenyomásával az API amely a jelszóváltoztatást elvégzi meghívásra kerül, és miután a változtatás megtörtént, az oldal átnavigálja a usert-t a login oldalra.

# 7. Backend

## 7.1 Bevezetés

A backend a webshophoz Express.js keretrendszer segítségével került megvalósításra; elsősorban a már meglévő tudás miatt került választásra. A backend egy teljes értékű REST API-t valósít meg, amit a frontend kényelmesen tud hívni. Fontos cél volt, hogy a kód rendezett, átlátható legyen, hiszen, hogyha bármilyen fejlesztésre van szükség, egyszerűen lehessen bővíteni vagy átírni a kódot, még akkor is, hogyha a fejlesztő még nem látta előtte a projektet. Adatbázishoz NoSQL-t használtunk, amit MongoDB-vel kezeltünk. A NoSQL nem sémaérzékeny, mint az SQL, ettől függetlenül célszerű sémákat készíteni; célszerű felkészíteni a programot megfelelő adatok fogadására, hiszen nem lenne ésszerű, hogyha értelmetlen adatok kerülnének felvitelre.

## 7.2 Belépő fájl

A program indításakor az app.js nevű, a backend főkönyvtárában megtalálható fájl kerül meghívásra. Ez a fájl felelős a program előkészítésért, itt valósulnak meg a modellek, controllerek és modulok importálása is.

## 7.3 Termékek lekérdezése

Termékek megtekintéséhez nincs szükség hitelesítésre, hiszen vendégként is lehet vásárolni. Egy specifikus termék adatai is lekérhetők, amelyhez szintén nem szükséges, hogy a felhasználó be legyen lépve.

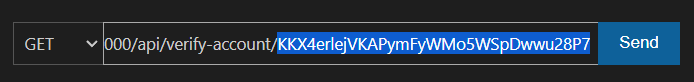
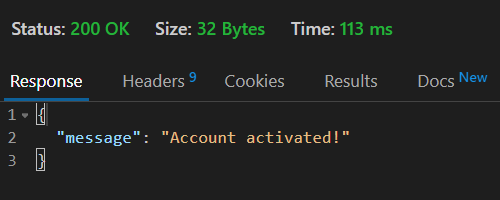
## 7.4 Regisztráció (Register & Verify Account)

Habár a felhasználók bejelentkezés nélkül is tudnak vásárolni, kényelmi szempontból érdemes regisztrálni. A felhasználó eltárolhatja adatait, így azokat később nem kell megadnia újra és újra. Regisztráláskor kötelezően csak egy valódi email cím és egy jelszó az elvárás, így azok nem megléte hibaüzenetet von maga után.

Regisztráció igényt követően a várható felhasználónak az adatbázisban létrejön a pofilja, és egy activatorToken mező is. Ennek feladata, hogy amíg ez létezik, addig a felhasználó nincsen aktiválva.

A képen szöveg látható

Automatikusan generált leírásA tokent a regisztráló email-ben kapja, amely átirányítja egy útvonalra, aminek a feladata, hogy végrehajtsa az aktiválást. Az adatbázisból kikerül az activatorToken, a felhasználó mostmár aktív. Természetesen hibák esetén a megfelelő üzenet kerül visszaküldésre. Amennyiben egy felhasználó regisztrált, de nem aktiválta azt, és megpróbál újra regisztrálni, akkor is a megfelelő hibaüzenet jelenik meg.

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírás

## 7.5 Bejelentkezés

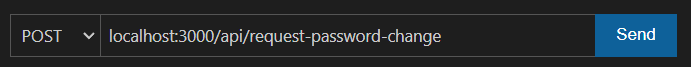
A képen szöveg látható

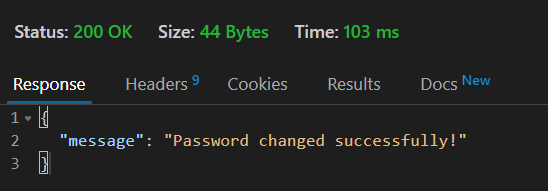
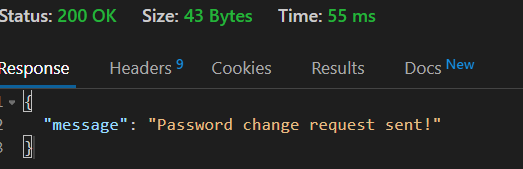
Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásA képen szöveg, képernyőkép, képernyő látható

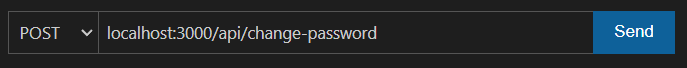
Automatikusan generált leírásBelépésnél megfelelő email és jelszó bevitele után a felhasználó kap egy LOCAL\_KEY nevű cookie-t, amelynek a hitelesítésben van szerepe. Ennek hiányában nem képes a weboldal funkcióit használni.

## 7.6 Új jelszó igénylése

A képen szöveg látható

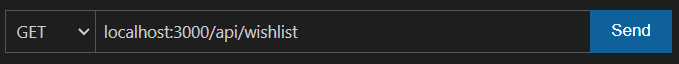
Automatikusan generált leírásA képen szöveg látható

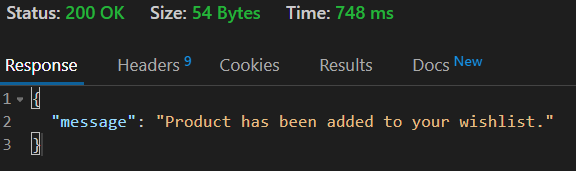
Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásLehetőség van új jelszó kérésére is a felhasználónak, hiszen nem biztos, hogy emlékszik a régire, vagy csak újat szeretne létrehozni fiókjának biztonsága érdekében. Elvárt mező csakis az email, amelyben kap egy linket, amely segítségével tudja frissíteni a jelszavát.

## 7.7 Kívánság lista (WishList)

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásA felhasználóknak lehetőségük van saját kívánságlista létrehozására. Ebben a listában termékek helyezhetők, de kikötés az, hogy egy termék többször nem szerepelhet. Természetesen ez a probléma a program által kezelve van. A kívánság lista le is kérdezhető, hasonlóan a termékekhez, de itt már elvárt a hitelesítés.A képen szöveg látható

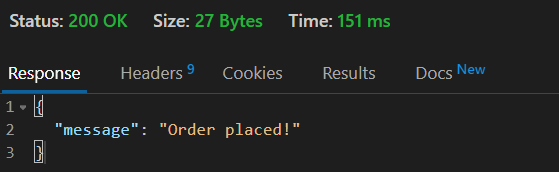
Automatikusan generált leírás

## 7.8 Megrendelések (Orders)

A képen szöveg, képernyőkép, monitor, fekete látható

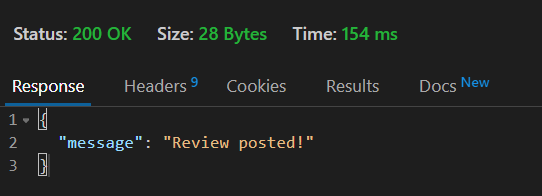
Automatikusan generált leírásA képen szöveg, képernyőkép, monitor, képernyő látható

Automatikusan generált leírásA képen szöveg, képernyőkép, képernyő látható

Automatikusan generált leírásA képen szöveg látható

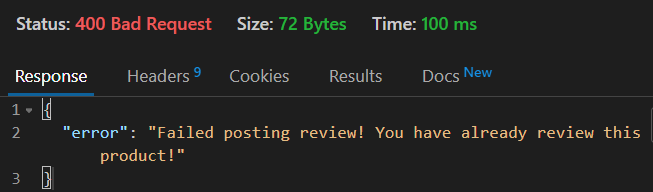
Automatikusan generált leírásFelhasználóknak természetesen lehetőségük van rendelések leadására. A frontenden tárolt kosár tartalmát alakítja át rendeléssé. Rendelés leadását követően a vásárló e-mailes tájékoztatót kap arról, hogy miket vásárolt.

## 7.9 Értékelések (Reviews)

A képen szöveg látható

Automatikusan generált leírásA felhasználóknak van lehetőségük megjegyzés/értékelés hozzáfűzéséhez. Egy felhasználó egy termékhez csak egyszer tud hozzászólni. Természetesen van lehetősége a megjegyzését törölni. Kötelezően elvárt mező a rating és a comment.

A képen szöveg látható

Automatikusan generált leírás

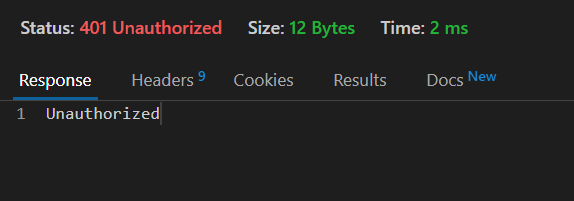
## 7.10 Admin

A képen szöveg látható

Automatikusan generált leírásAdminisztrátorok csakis manuálisan állíthatók, és csakis ők tudnak a termékek létrehozásával, módosításával foglalkozni. Az adatbázisban kapnak egy „admin” nevű tulajdonságot, amely ellenőrzésre kerül, ha a felhasználó létre kíván hozni egy terméket. Természetesen, ez igaz a termékek módosítására és törlésére is.

A képen szöveg látható

Automatikusan generált leírásSikertelen azonosítás esetén a funkció nem elérhetőek, a megfelelő hibakód visszaküldésre kerül.



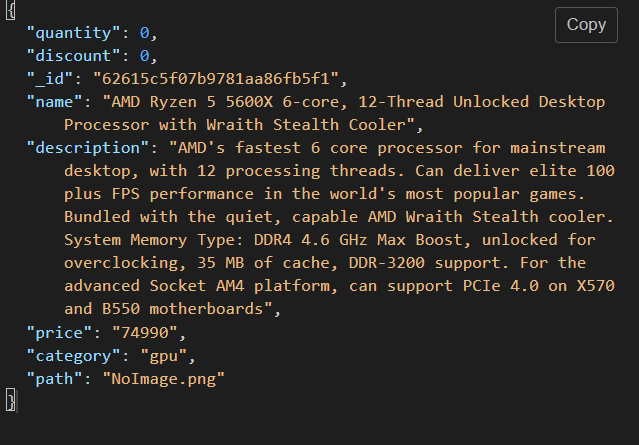
### 7.10.1 Termék létrehozása

A képen szöveg látható

Automatikusan generált leírásÚj termék létrehozásához a már említett admin jog szükséges. Új termék felvitelekor kötelezően elvárt mezők közé tartozik a termék neve, leírása, ára, kategóriája. Az akció mértéke nem elvárt, alapból 0, azaz 0% akció lesz a terméken. Létrehozás után a termék darabszáma 1-re lesz beállítva, ami módosítható. A képen szöveg, képernyőkép, képernyő látható

Automatikusan generált leírás

### 7.10.2 Termék módosítása

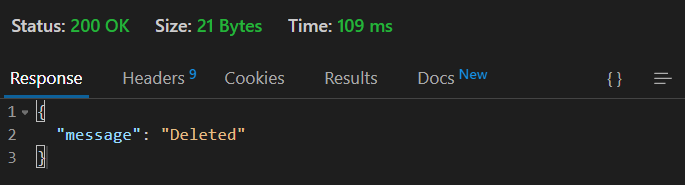
Hasonlóképpen, mint a termékek létrehozásához, a termékek módosítását is csak adminisztrátor végezheti. Kötelezően elvárt mező nincsen, amely mezőket nem adjuk meg, azok nem kerülnek módosításra, a path kivételével. Amennyiben a path-ot nem adjuk meg, az automatikusan a NoImage.png path-ot választja magának.A képen szöveg látható

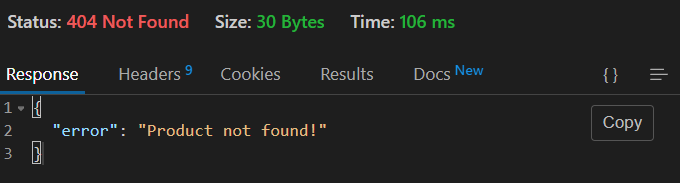
Automatikusan generált leírás

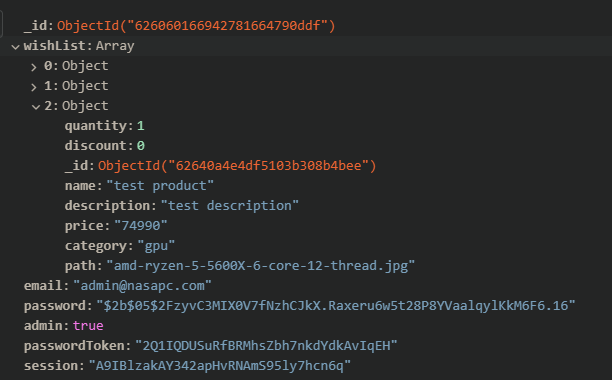
### 7.10.3 Termék törlése

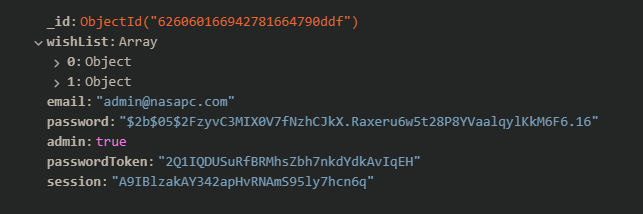
A képen szöveg látható

Automatikusan generált leírásTermék törlését is csakis adminisztrátor végezheti. Az útvonalban átadott termék ID-ja alapján a program megkeresi majd kitörli a terméket.



Nem létező termék esetén hibaüzenet visszaadása megtörténik.

Fontos észben tartani a kívánságlistát. Nem jó megoldás az, hogyha egy kitöröl termék még mindig ott lenne törlés után valamelyik, esetleg több felhasználó listájában. Éppen ezért, a függvény végigmegy a létező felhasználók WishList-jén és frissíti azokat.

Termék törlése után:

## 7.11 Backend teszt

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírásEgy jó programhoz elvárás, hogy legyenek tesztek. A tesztek segíthetnek a program ellenőrzésében, megkönnyítve a munkánkat, hiszen, ha egy új funkciót beépítünk a programba, elég lefuttatni a már létező teszteket az ellenőrzésre. A Thunder Client „tesztelés” nem megfelelő, hiszen idő szempontjából nagyon lassú minden funkciót letesztelni. Ezért a cél az volt, hogy a futtatható teszt fájl kerüljön megvalósításra. Egy Windows batch fájllal is indítható teszt lett a végeredmény, amely a backend funkcióit teszteli. Bármiféle hiba esetén, az adott teszt vagy tesztek nem futnak le.

### 7.11.1 Bevezetés

A teszteléshez a chai-http nevű modult használtam. Először is különálló teszt mappát hoztam létre, amiben található egy JavaScript fájl. A backend főkönyvtárából kiadva az npm test parancs ezt a test JavaScript fájl fogja elindítani.

Először is importáltam a modulokat, amelyekre szükségem lesz a tesztek megírásához. A sütik kezelésére csakis úgynevezett „agent” típusú változó volt alkalmas. Ez a legfontosabb eleme a kódnak, agent nélkül nem fogok tudni sütiket alkalmazni, ami problémás lenne, hiszen maga a backend használ sütit.

A képen szöveg látható

Automatikusan generált leírás

Ezen kívül létrehoztam báb adatokat, amelyek automatikusan felhasználásra kerülnek a program futása közben, nincs szükség, hogy kézzel állítsuk be.

A képen szöveg látható

Automatikusan generált leírás

A képen szöveg látható

Automatikusan generált leírásEgy hibaellönrző logikát is tartalmaz a kód, ami eltárolja a hibákat, hiszen több is előfordulhat; ezeket mind érdemes ki is íratni.

### 7.11.2 Bejelentkezés

A teszt a backend legtöbb funkcióját leteszteli. Admin belépéssel kezd, hiszen előnyös lenne, hogyha az admin funkciókat is le lehetne ellenőrizni. Az admin is tulajdonképpen egy felhasználó, így a felhasználót érintő funkciók is tesztelhetők.

A képen szöveg látható

Automatikusan generált leírás

A teszt elküldi a megfelelő útvonalra az adatokat, és azt követően vizsgálja, hogy milyen válasz érkezett (end szekció). Jelen esetben az az elvárandó, hogy a kód 200-as http státusz kódot küld vissza. Ezen felül elvárt, hogy maga a meghívó agent tartalmazza a bejelentkezéskor kapott sütit. Ez utóbbi elengedhetetlen a folytatáshoz, enélkül nem lehetséges az authentikációt igénylő funkciók letesztelése.

### 7.11.3 Munkamenet (Session)

A képen szöveg látható

Automatikusan generált leírás

A bejelentkezést követően az aktuális munkamenet lekérdezése a cél. Elvárás, hogy belépés után 200-as státusz kód kerül visszaküldésre. Ezen kívül a válaszban elvárt, hogy legyen email mező, amelynek értéke megegyezik az admin e-mailjével. Természetesen nem felejtendő a süti sem. További elvárás, hogy az admin mező értéke true legyen, hiszen szeretném az admin funkciókat is ellenőrizni.

### 7.11.4 Termékek / Termék ID alapján

Mindezek előtt leteszteltem a nem authentikációt igénylő végpontok közül kettőt: Az összes termék lekérdezését, és egy specifikus termék lekérdezését. Utóbbit a termék ID-je alapján. Ezek a funkciók nem igényelnek ellenőrzést, hiszen vendégek is járhatnak az oldalon, és nekik is meg kell adni a lehetőséget, hogy böngésszék a termékeket, vagy egyetlen termék adatait is megtekintsék.

A képen szöveg látható

Automatikusan generált leírás A teszt a válaszban elvárja, hogy létezzenek a fontosabb mezői a terméknek; amennyiben ez nem valósul meg, akkor a teszt ezen része hibával tér vissza.

### 7.11.5 Kívánság lista (WishList)

Visszatérve a bejelentkezést igénylő funkciókhoz, előnyös lenne a kívánság lista működését is letesztelni. Ez a teszt csak akkor fog lefutni, hogyha az agent-nek létezik sütije. Amennyiben nem létezik, ez a teszt hibával tér vissza, pontosabban 401-es hibakóddal.

A képen szöveg látható

Automatikusan generált leírás

Elvárás, hogy a 200.as státusz kód érkezzen a válaszban. Szintén elvárás, hogy a válasz törzsében létezzen egy message nevű mező, aminek meg kell egyeznie a backend-ről visszaküldött helyes válasszal. Amennyiben ez megérkezett, a funkció helyesen működik.

Azt is érdemes lenne ellenőrizni, hogy lekérhető a kívánságlista tartalma.

A képen szöveg látható

Automatikusan generált leírás

Jelent esetben a válaszban elvárás a 200-as státusz kód, és az, hogy a válasz törzse egy objektumnak felel meg. Ezen kívül szeretném ellenőrizni, hogy adott termék törölhető a kívánságlistából. Amennyiben nem találja a kívánságlistában a terméket, vagy egyéb hiba van a tesztben, akkor hibával kell visszatérnie. Elvárás, hogy sikerüljön kitörölni a nemrég kívánság listához adott terméket.

A képen szöveg látható

Automatikusan generált leírás

### 7.11.6 Termék létrehozása

Az admin műveleteit illetően, új termék felvitele egy elvárt funkció, ezért kihagyhatatlan az ellenőrzése.

A képen szöveg látható

Automatikusan generált leírás

Ebben a tesztben több mező is ellenőrzésre kerül, és maga a funkció rendkívül fontos, ezért érdemes nem csak egy általános hibával visszatérni hiba esetén, hanem az összes hibát kidobni a képernyőre.

Először elküldésre kerül a báb adat, majd a válaszban elvárás, hogy 201-es státusz kód kerüljön visszaadásra. Ezen kívül a válasz törzsének 10 kulcsot kell tartalmaznia, amelyek név szerint elvártak (például \_id, name, price).

### 7.11.7 Termék módosítása

A képen szöveg látható

Automatikusan generált leírásA termék módosításának lehetősége is egy olyan funkció, amelynek hibás működése nem tolerálható. Itt is hiba, vagy hibák esetén az összes hiba visszaküldésre kerül. Azonban jó lenne az utolsó elemet, vagyis amit a tesztben felvittünk, megtalálni. Ezért még a végpont hívása előtt az összes termék lekérdezésre kerül, melyek közül egy, a legfrissebb kerül átadásra egy prod nevű változóba. Mindezek után elvárás, hogy a válasz 200-as státusz kódot adjon vissza, a válasz törzse objektum legyen, majd hasonlítsa össze a visszaadott értékeket a kód elején megadottakkal.

### 7.11.8 Termék törlése

Termékek törlése is olyan funkció, melynek megléte elengedhetetlen, éppen ezért tesztet igényel.

A képen szöveg látható

Automatikusan generált leírásHasonlóképpen a termék módosításához, ez a teszt is megkeresi a legutolsó / legfrissebb terméket, majd törli azt. Elvárt a 200-as státusz kód, és a megfelelő megerősítő üzenet megléte.

### 7.11.9 Új jelszó igénylése

Általános felhasználói funkció a jelszóváltoztatás. Ebben a tesztben csak maga a jelszó változtatási kérelem kerül ellenőrzésre.

A képen szöveg látható

Automatikusan generált leírás

A végpont csak egy email mezőt vár a kérés törzsében. Elküldés után a válaszban 200-as státusz kód az elvárt. Ezen kívül a message mező megléte és a megfelelő megerősítő üzenet is.

### 7.11.10 Termék módosítása (Modify product)

A legutolsó teszt csak annyit ellenőriz, hogy képes-e kijelentkezni a felhasználó.

A képen szöveg látható

Automatikusan generált leírás

Az elvárás ebben az esetben az, hogy a válasz 200-as státusz kódot tartalmazzon, és hogy az agent-nek ne legyen már LOCAL\_KEY nevű sütije.

# 8. Android app

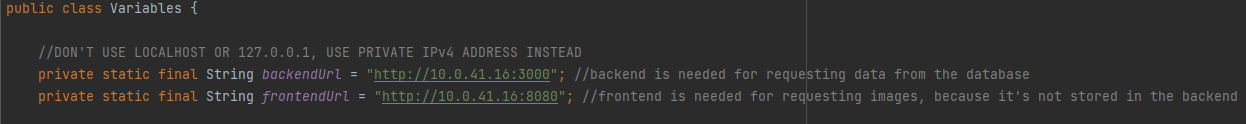
## 8.1 Android Alkalmazás

A webshop egyszerűbben használható a mobil alkalmazás segítségével, hiszen így a felhasználó könnyebben tud navigálni a termékek és menük között, mint egy weboldalon. Mivel az alkalmazás telepítésre kerül az eszközön ezzel a felhasználó adatforgalma is csökkenthető, hiszen nem szükséges az összes fájlt letölteni, csak, ami az adatbázisból kell.

A telefonos alkalmazásnál több szempontot is figyelembe vettünk, hogy melyik eszközöket érné meg legjobban célozni. Mivel országunk túlnyomó többsége Android operációs rendszerrel ellátott eszközöket használnak, így Java programozási nyelven írtuk meg az alkalmazást az Android Studio program segítségével. Ezzel, hogy natív kódot tudunk írni az eszközre, sokkal gyorsabban futtatható, optimalizált és kevesebb tárhelyet foglaló alkalmazást érhetünk el. Hátrányban sajnos elmondható, hogy így iOS-en nem futtatható az alkalmazás, de más szempontból se lett volna előnyös, hiszen iOS-re az alkalmazást csak megbízható fejlesztőtől lehetne telepíteni, míg Android-ra ez nem szükséges.

Az alkalmazásban a backend-et használva tudunk adatokat kinyerni az API segítségével, hiszen az adatok az adatbázisban vannak eltárolva. A frontend szerverre is szükség van, hiszen a képek csak ott vannak eltárolva.

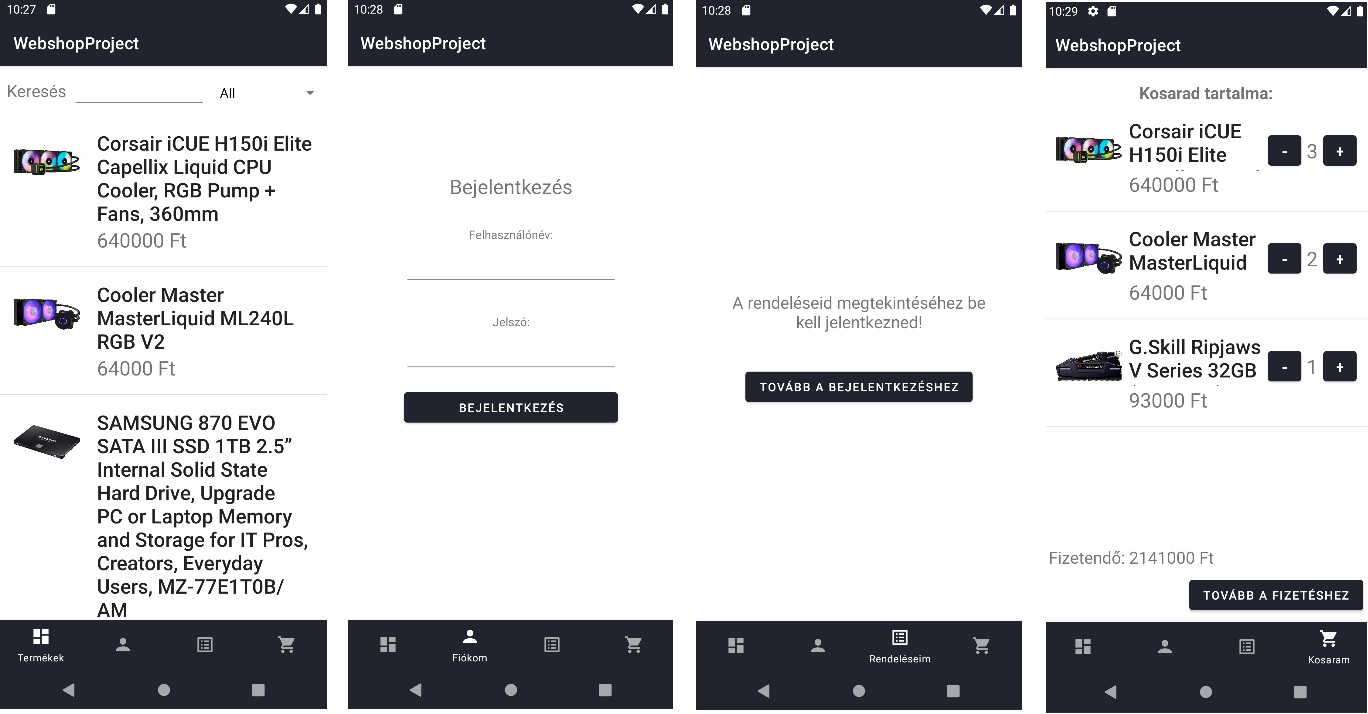
Mivel hogy a projekthez nem béreltünk VPS-t, így nincs állandó IP cím, ahonnan az adatokat le lehessen kérdezni, ezért az Android alkalmazás program kódjában folyamatosan meg kell változtatni a backend és a frontend URL-nek a címét, hogy a telefonos alkalmazás a megfelelő helyről kérje le a szükséges adatokat. Ezeket a szerver címeket a Variables osztályban érhetjük el, ahol az adatok nagy része van tárolva.



## 8.2 Az alkalmazás felépítése:

Az Android alkalmazás tartalmaz egy alsó navigációs menüt 4 elemmel:

* Termékek listája (Főoldal)
* Profil (Hitelesítés)
* Rendelések (Csak bejelentkezett felhasználóknak)
* Kosár

Illusztráció a menükről:

A termékek listája a [**GET**] *http://<backend\_szerver>:<port>/api/products* –ról töltődnek be. Ilyenkor megjelenítésre kerül a termék neve, ára és képe. Hogyha a felhasználó részletesebb leírást szeretne kapni, akkor rákattinthat a listában lévő termékre, így elolvashatja a termék leírását és a képet is nagyobb felbontásban láthatja, de ezen túl itt adhatja hozzá a terméket is a kosárhoz.

A termék sikeres hozzáadását egy felugró kis ablak (Toast) jelzi.

A profil menüpontban a felhasználó betud jelentkezni az alkalmazásban. Ez az oldal önmagában nem szolgáltat túl sok információt, de a későbbiekben még bővíthető lehetne egyéb funkciókkal, pl. jelszó változtatás, vagy egyéb felhasználónak szánt beállítás megváltoztatásához. Jelenleg csak a bejelentkezett felhasználó e-mail címét jeleniti meg ez az oldal.

A megrendelések kilistázásához külön route-ok lettek létrehozva, hiszen az Android-os alkalmazással nem lehetett megoldani a Cookie-k kezelését, vagy csak nagyon nehezen lehetett volna, ezért az adatokat a [**GET**] *http://<backend\_szerver>:<port>/api/mobile/getUserOrders/<userId>/<sessionKey>* címről tudjuk lekérdezni. A lekérdezett adatokból egy listát generál az alkalmazás a felhasználónak, itt megtekintheti a meglévő megrendelésének adatait, mint a: rendelés azonosító (6 jegyű kód), megrendelt termékek száma, a megrendelt termék neve és a fizetett ár.

Végül az utolsó menüpontban a kosarunk tartalmát láthatjuk. Ha nincs semmi a kosárban, akkor az „Üres a kosarad” felirat jelenik meg, ilyenkor elvan rejtve a megrendelés gombja, és a összesen fizetendő felirat. De hogyha valami hozzáadásra kerül, akkor ezek megjelennek és automatikusan frissülnek a kosár állapotának változásáról. A kosárban a terméknél a mínusz (-) és plusz (+) jelekkel tudjuk a megrendelni kívánt termékből a darabszámot módosítani, hogyha ez az érték 1 alá csökkenne, akkor automatikusan eltávolítódik a termék a kosárból.

Hogyha a felhasználó kiválasztotta a számára megfelelő termékeket, akkor kosárban a „tovább a megrendeléshez” gomb segítségével egy új oldalra navigáljuk, majd itt egy űrlapot kell kitölteni-e, ahol meg kell adnia a nevét, telefonszámát, email címét és a szállítási címet. Külön el kell fogadni a vásárlással kapcsolatos szabályzatot, ahol a checkbox-ot be kell pipálni, különben a rendelés nem kerül feladásra, és egy hiba üzenetet kapunk. De ugyan így a többi mező is validálásra kerül, így üres mezőkkel nem lehet rendelést feladni. A rendelés feladásakor a [**POST**] *http://<backend\_szerver>:<port>/api/mobile/place-order/<sessionKey>*  route hívódik meg, ahol a body-ban átadásra kerülnek a következő értékek:

* first\_name
* last\_name
* email
* phone\_number
* shipping\_address
* billing\_address
* items

A rendelés feladása után a felhasználót átirányítjuk a következő oldalra, ahol a „Sikeres rendelés feladás” fogadja. Ezután a bejelentkezett felhasználó megtudja nézni a feladott rendelését.

A képen szöveg látható

Automatikusan generált leírásIllusztráció a megrendelés folyamatáról:

# 9. Kommunikáció a csapattagok között:

A csapat tagjainak a fejlesztés során volt lehetősége személyes kapcsolattartásra így a közös munka zavartalanul tudott zajlani a fejlesztés túlnyomó részében, ám amikor ez az állapot nem állt fen, több különböző módot is felhasználtunk a kapcsolattartásra.

Az elsődleges módja a kommunikációnak a Facebook Messenger használata volt, illetve az azon való chatelés, de a fejlesztés első fázisaiban más módszereteket is fontolóra vettünk.

A Discord is nyújtott volna egy stabil platformot a zavartalan kommunikáció fenntartására. A Discord egy kommunikációs platform, amely lehetőséget nyújt több fél közti chatelés, streamelés és meetek indítására, továbbá képek fájlok és egyéb kisebb tartalmak közzétételére.

A projektet a GitHubon keresztül fejlesztettük. A GitHub segítségével nyomon tudtuk követni ki, mit tett közé, és esetleges ütközések esetén könnyen tudtuk a hibákat kezelni.

# 10. Felhasznált források:

[Mit is jelent pontosan a JavaScript kifejezés és hogyan működik, mire jó? (matebalazs.hu)](https://matebalazs.hu/javascript.html)

[Node.js alapok · Weblabor](http://weblabor.hu/cikkek/nodejs-alapok)

[What is Express.js? | Why should use Express.js? | Features of Express.js (besanttechnologies.com)](https://www.besanttechnologies.com/what-is-expressjs)

([Index | Node.js v17.8.0 Documentation (Node.js.org)](https://nodejs.org/api/))

[Introduction | Vue.js (vuejs.org)](https://vuejs.org/guide/introduction.html#what-is-vue)

1. Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot. [↑](#footnote-ref-1)
2. Szakmajegyzékes, csoportos konzultációs lap [↑](#footnote-ref-2)