Tartalomjegyzék

[1. Bevezetés: 3](#_Toc100656985)

[2. Specifikáció: 4](#_Toc100656986)

[4. JavaScript 5](#_Toc100656987)

[4.1 A JavaScript története 6](#_Toc100656988)

[4.2 Működése 6](#_Toc100656989)

[4.3 Mire képes a JavaScript? 7](#_Toc100656990)

[5. NodeJS 7](#_Toc100656991)

[5.1 Célja 8](#_Toc100656992)

[5.2 Modulok 8](#_Toc100656993)

[5.3 npm 9](#_Toc100656994)

[5.4 Express 10](#_Toc100656995)

[5.4.1 Indoklás 10](#_Toc100656996)

[5.4.2 Előnyök 11](#_Toc100656997)

[5.5 Express funkciók 12](#_Toc100656998)

[5.5.1 Gyorsabb szerveroldali fejlesztés 12](#_Toc100656999)

[5.5.2 Middleware 12](#_Toc100657000)

[5.5.3 útválasztás 12](#_Toc100657001)

[5.5.4 Sablonmotorok 12](#_Toc100657002)

[5.5.5 Hibakeresés 13](#_Toc100657003)

[5.5.6 Összegzés 13](#_Toc100657004)

[6. Frontend 15](#_Toc100657005)

[Navigation komponens 15](#_Toc100657006)

[Footer komponens 16](#_Toc100657007)

[Home page 16](#_Toc100657008)

[About us 17](#_Toc100657009)

[Products (user) 17](#_Toc100657010)

[Signup és Login 18](#_Toc100657011)

[7. Backend 19](#_Toc100657012)

[7.1 Bevezetés 19](#_Toc100657013)

[7.2 Belépő fájl 20](#_Toc100657014)

[?. Kommunikáció a csapattagok között: 20](#_Toc100657015)

[Felhasznált források: 21](#_Toc100657016)

EZ ITT MEG NEM JO, NE NÉZZED, KI KELL JAVÍTANI

# 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 felé hajlunk, hogy amit csak lehet, azt valamilyen online felületen végezzük, ez az ügyintézéstől fogva a meetingek és találkozók megtartásán keresztül egészen a vásárlásig terjedt ki. Ezért döntöttünk úgy mi sem állunk a változás újtába és a növekvő igénynek megfelelve, mi ezen a területen szeretnénk létrehozni átfogó porjektet, amely teljesen életképes a ma, azaz 2022 igényeinek, teljes mértékben megfelel és kielégítő felhasználói élményt nyújt.

Fejünkben többféle remek ötlet is megfordult, példának okáért egy futárszolgálat weboldala, online kaszinó ötlete és egy mezőgazdasági videómegosztó platform is.

A weboldal tárgyát/témáját illetően több lehetőség is megfogalmazódott;

Például:

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

# 2. Specifikáció:

Vésősoron 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.

Lesz lehetőség regisztrált felhasználóknak, wishlist létrehozására. Lehetőség lesz arra is, hogy a felhasználó kérjen értesítő emailt, hogy mindig naprakész információja legyen a legújabb akciókról és új termékekről.

Regisztrált felhasználók továbbá tudnak hűségpontokat gyűjteni, ha egy bizonyos érték felett vásárolnak.

Szükségünk van keretrendszerekre, illetve azok alkalmazására, hiszen ezeknek a segítségével, jóval gyorsabban dolgozhatunk, valósíthatjuk meg az adott funkcionalitást. A keretrendszer lényege az, hogy a különböző alkalmazásokban legnagyobb gyakorisággal használt elemeket egy helyre csoportosítja, ezáltal gyakorlatilag készen kínálva ezeket a webfejlesztők, illetve a különböző programok számára.

A programozók így nagyon sok elvégzendő munkától szabadulnak meg. A keretrendszerek egységesen önmagukban nem használhatóak, de nagymértékben könnyítik az egyes tipikus feladatok elvégzését. A komplexebb folyamatokat leegyszerűsíti, tiszta és újra felhasználható kódot írhatunk velük. Megkönnyíti a hibakeresést és a tesztelést, rugalmasabb alkalmazásokat eredményez, illetve webshopkészítés során is igen hasznos lehet.

Ennek következtében, ami a backendet illeti a natív nyelven írt backend ötletét elvetettük, a szükségtelen bonyolultsága miatt.

Ezen felül, gondoljunk csak a böngésző kompatibilitásra: ez a probléma nem csupán CSS esetén merülhet fel. A megírt kódunk nem egyformán fog működni minden böngésző, illetve azok verziói alatt.

Elsősorban két lehetőséget vettünk fontolóra: A PHP alapú Laravel framework-öt és a JavaScript NodeJs alapú ExpressJs –t.

# 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.

Az új programnyelvnek több neve is volt (Mocha, LiveScript), mielőtt 1995 decemberében megkapta végleges nevét.

## 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 lehetoldani, 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. NodeJS

A Node.js (vagy másnéven Node) egy rendszer, melyben JavaScriptben 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 alkalmazások írására. 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 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. ([Index | Node.js v17.8.0 Documentation (nodejs.org)](https://nodejs.org/api/))

## 5.3 npm

A Node hivatalos csomagkezelője az [npm](http://npmjs.org/), 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 mint 150.000 csomag van és naponta több mint 60 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 (global) 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 stb). Maga a parancs nagyon hasonló az előzőhöz: npm install -g jshint

## 5.4 Express

Az Express egy ingyenes és nyílt forráskódú webalkalmazás keretrendszer a NodeJS 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.

A framework szintén az MVC modellt követi, de főleg az miatt döntöttünk az express mellett, mert ez egy JavaScript alapú rendszer, melynek szintaxisát jelentősan jobban kedveltünk a PHP-val szemben.

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 komponsenseket, melyek a fejlesztés során gyakran használtak. Az imént felsorolt okok miatt a választás számunkra egyértelmű volt, hogy szavazatunkat az ExpressJS mellett tegyük le.

### 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 ExpressJS 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. Ezért az ExpressJS lehetővé teszi a fiatal tehetségek számára, hogy belépjenek a webalkalmazás-fejlesztés területére és sikeresek legyenek.

Minden vállalkozás másik fontos eszköze a pénz. Fontos, hogy a pénzt hatékonyan használjuk a nyereség maximalizálása érdekében. Mivel az ExpressJS egy nyílt forráskódú és ingyenes webes alkalmazás, amely számos nagyszerű funkciót kínál, nincs különösebb ok arra, hogy ne használjuk ezt a remek rendszert.

### 5.4.2 Előnyök

* Gyorssá és egyszerűvé teszi a webalkalmazások fejlesztését.
* Könnyen konfigurálható és testre szabható.
* Különböző köztes szoftvermodulokat tartalmaz, amelyekkel kérés és válasz methódusok segítségével további feladatok hajthatóak végre.
* Könnyen integrálhatóak a különböző sablonmotorok, mint például Jade, Vash, EJS, Pug stb.
* 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 ExpressJS egy rendkívül fejlett útválasztási mechanizmust biztosít, hasonlóképpen a Laravelhez.

### 5.5.4 Sablonmotorok

Az ExpressJS 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 ExpressJS 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 NodeJS 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.

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

Bár a MySQL tűnhet az egyértelmű megoldásnak, a NoSQL lehetőségét komolyan fontolóra vettük és végül az mellett is döntöttünk.

Erre a döntésre nem voltak hasonló indokaink, mint az előzőekben, hiszen mind a két megoldás teljesen valid és széles körben használt megoldás.

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 döntésünket személyes preferencia alapján hoztuk meg, így tehát NoSQL-t fogunk használni.

# 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.

## 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.

## 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.

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.

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.

## Products (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 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 azért fontos, mert így eltudjuk kerülni, azt hogy összezavarjuk a felhasználót.

## Signup 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 validációk sikeressége után, egy api hívás segítségével folytatódik a regisztrációs folyamat.

Az emailben elküldött link által való fiók aktiválását követően, a fiók adatbázisban való tárolása megtörténik.

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.

# 7. Backend

## 7.1 Bevezetés

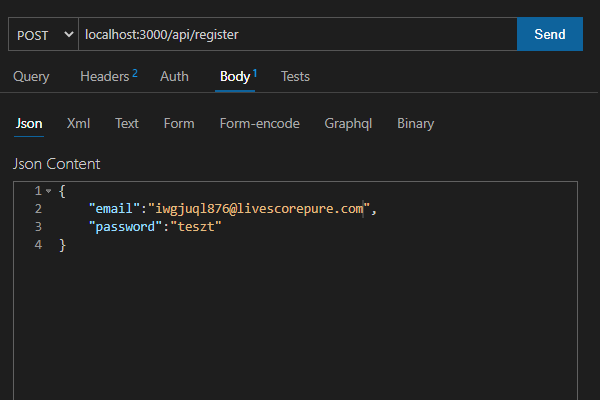
A backend a webshophoz ExpressJS keretrendszer segítségével került megvalósításra; elsősorban a már meglévő tudás miatt került választásra.

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 szükség van, 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.

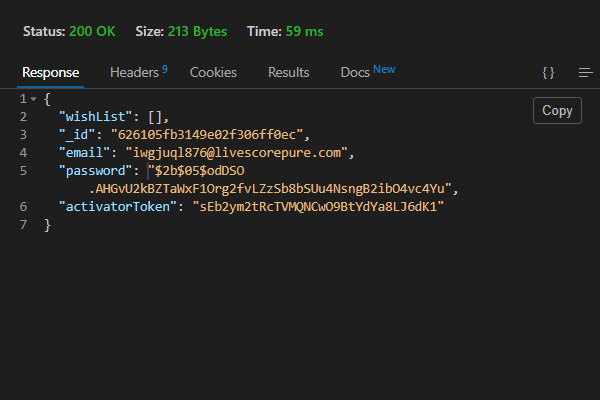
## 7.2 Belépő fájl

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 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. A regisztrált felhasználók pontokat gyűjthetnek, amelyek további juttatások felhasználásra alkalmasak, például ingyenes szállításra. Regisztráláskor kötelezően csak egy valódi email cím és egy jelszó az elvárás.

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

A 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.

**ide még kép jön**

## ?. 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 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.

Bár ezek mind előnyösek lehetnek a zavartalan munka fenntartásában, a Facebook lehetőséget biztosított az egymás között folytatott üzenetküldésre, a fájlmegosztásra pedig a github szolgáltatásait vettük igénybe. A streamelésre és a konferenciák indítására nemvolt szükség.

A projektet a github-on vezettük ahová a projekten történő változtatásainkat és munkáinkat commiteltük. Ez és hogy visszatudjuk állítani a projekt egy korábbi verzióját hiba észlelése esetén arra indított minket, hogy a github verziókezelő segítségével fejlesszük és tároljuk a projektet.

# Felhasznált források:

[PHP programozás alapjai 1. rész (phpmuhely.hu)](https://phpmuhely.hu/muhely/segedletek/olvas/php-programozas-alapjai-1-resz)

[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)