Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Pál Krisztián Zoltán

2023

Szegedi Tudományegyetem

Informatikai Intézet

Italárusító webshop alkalmazás beépített játékkal

Szakdolgozat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Készítette:* |  | *Témavezető:* |  |
|  | Pál Krisztián Zoltán |  | Dr. Bilicki Vilmos |  |
|  | programtervező informatikus szakos hallgató |  | Oktató és konzulens |  |

Szeged

2023

# ***Feladatkiírás***

A szakdolgozat I. és szakdolgozat II. tárgyak keretein belül egy különböző alkoholos és alkoholmentes italokat árusító webáruházat alkot meg a hallgató, amelybe az általánosabb webshopokba található funkciók mellett egyedi funkciók valósítása is a feladat része. Többek között beleértve a fizetési rendszer integrálását, emellett az árucikkek közötti gördülékeny böngészést.

A szakdolgozat teljesen öncélú, viszont felhasználhatósága kiterjedhet akár más sajátosan megalkotott, de ettől eltérő árucikkekkel foglalkozó weboldalak fejlesztésének alapjaként. Az elkészített webalkalmazás meg kell, hogy állja a helyét az interneten fellelhető más webshopokkal szemben.

Amióta az internet egyre elterjedtebbé vált, egyre népszerűbbek lettek a webshopok, melyek segítségével a felhasználók saját igényük szerint gyorsan tudnak böngészni a termékek tárházában. Mindig szükség van, és a jövőben is szükség lesz ezen platformokra, melyek megkönnyítik, kényelmesebbé teszik az emberek életét. Az áruház tulajdonosok számára is hatalmas előny, hiszen rengeteg fogyasztóhoz el tudják juttatni termékeiket a helyi áruházaik termelésén túlmenően.

A hallgató feladatát képezi egy újabb webshop megalkotása, mely azt az egyediséget viseli, hogy egy egyszerű játék alkalmazás beépítése a webshop keretein belülre színesíti azt, amely segítségével különböző kedvezményekre tehet szert a felhasználó. Így egyedi módon juttatjuk a felhasználót pénzügyi előnyhöz, emellett a webshopon is több időt tölt el, ami azt eredményezheti, hogy új termékeket ismer meg, több energiát fordít a vásárlásra. Ezen üzleti logika elősegítheti, hogy a felhasználók több árucikket, avagy gyakrabban rendeljenek, de még így is talán szimpatikusabbá válik számukra a webalkalmazás rendszere.

# ***Tartalmi összefoglaló***

* **A téma megnevezése:**

Italárusító webshop alkalmazás beépített játékkal

* **A megadott feladat megfogalmazása:**

A feladat egy olyan webalkalmazás fejlesztése, amely dinamikusan tölt be egy általános webshopot, melyen italok vásárolhatóak (természetesen csak virtuálisan). A webáruház kényelmes böngészést biztosít a felhasználónak a neki tetsző termékek között. Különlegessége a beépített játék, mely a felhasználóknak lehetőséget biztosít minden nap bizonyos értékű kedvezmények megszerzéséhez.

* **A megoldási mód:**

Első tekintetben körültekintőbben meg kellett ismernem az Angular-t, mint frontend keretrendszert. Ezután komponensekbe, sőt modulokba szerveztem az alkalmazásom, ami eleinte csak egy játék nélküli webshop lett, már működő teszt fizetési rendszerrel. Szolgáltatásokat, materialokat és az Angular egyéb segítséget nyújtó eszközeit alkalmaztam a felhasználóbarát megjelenés végett. A TypeScript kódból könnyedén kapcsolódni tudtam a Firebase-hez, ami a backendjét szolgáltatja a projektnek. Mindez után belemélyedtem a webes játékfejlesztésbe, és ezután sikeresen beépítettem egy tömbféle játékmódot tartalmazó játékot, ami lehetőséget biztosít a felhasználóknak bizonyos százalékú kedvezmény megszerzésére.

* **Alkalmazott eszközök, módszerek:**

Az alkalmazás a Visual Studio Code nevű fejlesztői környezetben készült. A logika TypeScript nyelven íródott, a megjelenítés HTML és SCSS használatát követelte meg. A CSS-hez a Tailwind könyvtárat is előszeretettel használtam, ami megkönnyítette számomra az elemek stílusainak beállítását, emellett javított az átláthatóságon. A fejlesztésben egy multiplatformos keretrendszer, az Angular nyújtott segítséget, melynek eleinte a 13-as verzióját használtam, viszont, mivel az már elavult végrehajtottam egy frissítést a 14-es verzióra. Adatbázisként, tárolóként, sőt autentikációt és hostingot biztosító eszköz gyanánt a Firebase felhőalapú szolgáltatás volt segítségemre. Ennek a részeként a Firebase Cloud Functions nagy segítségemre volt a fizetési rendszer beépítésekor, és a játékban kapott életerőpontok időhöz kötésekor. A projektet időnként a Git verziókezelő segítségével GitHubra feltöltve tárolom.

## Tartalomjegyzék

[*Feladatkiírás* 3](#_Toc151823223)

[*Tartalmi összefoglaló* 4](#_Toc151823224)

[Tartalomjegyzék 5](#_Toc151823225)

[1. Bevezetés 7](#_Toc151823226)

[1.1 Problémafelvetés, motiváció 7](#_Toc151823227)

[2. Alkalmazott eszközök, technológiák bemutatása 9](#_Toc151823228)

[2.1 Fejlesztői környezetek 9](#_Toc151823229)

[2.1.1 Visual Studio Code 9](#_Toc151823230)

[2.1.2 Böngésző fejlesztői mód 9](#_Toc151823231)

[2.2 Programozási nyelvek 10](#_Toc151823232)

[2.2.1 TypeScript 10](#_Toc151823233)

[2.2.2 HTML 11](#_Toc151823234)

[2.2.3 SCSS 11](#_Toc151823235)

[2.2.4 Tailwind 11](#_Toc151823236)

[2.3 Keretrendszerek 12](#_Toc151823237)

[2.3.1 Angular 12](#_Toc151823238)

[2.4 Firebase 12](#_Toc151823239)

[3. Terület áttekintés 13](#_Toc151823240)

[3.1. Webshop alkalmazások terület áttekintése 13](#_Toc151823241)

[3.2. Piackutatás 14](#_Toc151823242)

[3.3. Piackutatás értelmezése 15](#_Toc151823243)

[4. Funkcionális specifikáció 18](#_Toc151823244)

[4.1. Bejelentkezés és regisztráció 18](#_Toc151823245)

[4.2. Böngészés a termékek között 19](#_Toc151823246)

[4.3. Termékek részletei 20](#_Toc151823247)

[4.4. Felhasználói visszajelzések 20](#_Toc151823248)

[4.5. Kosárkezelés 21](#_Toc151823249)

[4.6. Integrált fizetési rendszer 22](#_Toc151823250)

[4.7. Kedvezmények szerzése (Beépített játék, egyszer használatos link) 23](#_Toc151823251)

[4.8. Felhasználói fiók kezelése 25](#_Toc151823252)

[4.9. Bejegyzések írása 26](#_Toc151823253)

[4.10. Admin funkciók 27](#_Toc151823254)

[4.11. Use-Case diagramok 27](#_Toc151823255)

[5. A megvalósítás lépései fontosabb kódrészletek kiemelésével 29](#_Toc151823256)

[5.1. Komponensek és modulok 29](#_Toc151823257)

[5.2. Modellek (Interfészek) 36](#_Toc151823258)

[5.3. Backend kódok 38](#_Toc151823259)

# **Bevezetés**

A webáruházak egyre nagyobb népszerűségnek örvendenek. Az embereknek hatalmas kényelmi funkciót biztosít, hogy személyesen el sem szükséges menniük a termékekért, csupán csak rendelniük kell. Ennek okán az igény erre folyamatosan csak nő. Ezen webalkalmazás többek között sablonként is szolgálhat újabban felépülő, hasonló célokra készített oldalak felépítésekor.

A cél az, hogy felhasználóbarát, minél gyorsabb, és minőségibb weboldalakat fejlesszünk, hisz az emberek türelmetlensége csökkentheti a vállalkozók szeme előtt lebegő bevételt, ha nem töltenek elég időt az adott weboldalon. A figyelemfelkeltés céljából ezen webáruház a szokványos hirdetések mellett egy beépített játékot fog tartalmazni, mellyel a felhasználó akciókra tehet szert. Ezzel elérhetjük azt, hogy több időt töltsön az oldalon, ezáltal több neki tetsző terméket tárhatunk a szeme elé, ami vásárlásra ösztönzi. Valójában ez egy nyerő-nyerő szituáció, hisz így elérjük, hogy a felhasználó vásároljon így profitot termelve a vállalkozónak. Még ha kevesebb is a profit az akciók miatt, a rendszer úgy lesz felépítve, hogy így is bőven nyereséges legyen.

A kész alkalmazás rendkívüli felhasználóbarátsága már nem igényel bármiféle szaktudást a kezeléshez. Egy laikus felhasználó is könnyen boldogulhat egyedül. Az admin felület kiépítése hasonló könnyedséget von majd maga után, ugyanis fontos, hogy az üzemeltető is gördülékenyen tudja működtetni az alkalmazást. A dinamikus megvalósítás lehetővé teszi, hogy könnyeden, mondhatni akármennyi ideig fusson alkalmazás, hisz a hosting-nak köszönhetően a világhálón bármikor megtekinthető. A hibák kezelése szintén beépített feladata az alkalmazásnak.

# **Problémafelvetés, motiváció**

A szakdolgozatom témája tehát az előző fejezetben felvázolt italárusító (mely természetesen más termékekkel is lehetne) webalkalmazás sablon elkészítése, egy egyszerű beépített több játékmódot tartalmazó játékkal felvértezve. Korszerű technológia használatával, és folyamatosan bekerülő új funkciókkal.

A webáruház hozzájárulhat ahhoz, hogy szélesebb körben elterjedjen ez a fajta piaci rés kihasználás, miszerint egy játékkal csalogatjuk oda, sőt inkább tartjuk ott az oldalunkra tévedő felhasználókat.

A motivációm tehát annak a problémának a megoldására ad egy lehetőséget, miszerint a türelmetlen felhasználók túl hamar elhagynak egy adott webáruházat anélkül, hogy vásárolnának. Ez sokszor köszönhető a magas várakozási időnek míg betöltenek bizonyos részletek, a nem túl felhasználóbarát felületnek vagy épp a magas áraknak. Mindezek megoldása mellett még kedvezményt is biztosítok a vásárlók számára, amennyiben hajlandóak játszani és/vagy regisztrálni. A játékok köztudottan addikciót okozhatnak, és ezen függőség kialakulását lehetővé téve akár napi szinten visszatérő vásárlóink is lehetnek.

# **Alkalmazott eszközök, technológiák bemutatása**

# **Fejlesztői környezetek**

A fejlesztői környezetek (IDE - Integrated Development Environment) olyan szoftvereszközök, melyek megkönnyítik a programozók munkáját. Olyasfajta segítséget nyújtanak, amik jelentősen lerövidíthetik egy-egy alkalmazás elkészítésének az idejét, így produktívabb és gyorsabb munkát eredményezve egyaránt.

Általában egy fejlesztő környezet több mindent magába foglal, többek között tartalmaz fordítót, futtatókörnyezetet, és a szoftver fejlesztéséhez elengedhetetlen szövegszerkesztőt. Mondhatni egy integrált munkakörnyezet, melyben a programozók írhatnak, szerkeszthetnek, tesztelhetnek és hibát is kereshetnek a kódban, mindezt átlátható módon. Javaslatokat tesz, amikor elkezdünk gépelni egy adott parancsot, változónevet, függvénynevet, sőt képes a gépelési hibáink kijelzésére is.

A fejlesztői környezet sok mindent rejt magában, mint például szintaktikai kiemelés, kódszerkesztési segítség, hibakeresési eszközök, verziókezelési integráció és sok egyéb. Különböző programozási nyelveket és platformokat segítő variációi léteznek már, melyekből a programozó kiválaszthatja a saját igényeit és preferenciáit legjobban kielégítőt. A teljesség igénye nélkül néhány: Visual Studio Code, Eclipse, Notepad++, PyCharm, WebStorm stb.

# **Visual Studio Code**

Az általam használt Typescript nyelvhez a leginkább illeszkedő fejlesztői környezet a Visual Studio Code. Ez egy keresztplatformos eszköz, mely rengeteg programozási nyelvet és platformot támogat. Ezen felül sok egyéb fejlesztést segítő funkciót és bővítményt kínál. A Microsoft fejlesztette ki ezt a terméket, mely a kényelem megteremtését szolgálja, mint más fejlesztői környezetek, de tudomásom szerint a legszélesebb körű az alkalmazása.

# **2.1.2** **Böngésző fejlesztői mód**

Ha egy böngészőben aktiválni szeretnénk a fejlesztői módot csupán annyit kell tennünk, hogy az aktív ablakban lenyomjuk az F12 billentyűt, vagy jobb gombbal kattintunk az oldalon és kiválasztjuk a „Vizsgálat” menüpontot. Ekkor előjönnek különféle fejlesztést segítő eszközök (általában frontend fejlesztőknek nagy segítségére van). A legtöbb manapság használatos böngészőben már fellelhető ez a beépített funkció. Mindez lehetővé teszi, hogy az általunk éppen fejlesztett vagy akár mások által készített weboldalakat elemezzünk. Könnyen látható, hogy ez gyors hibakeresésre is alkalmas.

Több panellel rendelkezik, melyek különböző funkciókat tesznek lehetővé: elemek vizsgálata, a DOM szerkesztése, a CSS (stílusok), vagy akár a JavaScript módosítása, a hálózati lekérdezések monitorozását, a konzol üzeneteinek megtekintését és sok más eszközt, amelyek segítségével fejlesztés közben interaktívan dolgozhatunk a weboldallal.

# **Programozási nyelvek**

A programozási nyelvek afféle mesterséges nyelvek, melyeket azért hoztak létre, hogy az emberek utasítások megadásával tudjanak kommunikálni a számítógéppel. Értelemszerűen a gépek sokkal gyorsabbak, logikusabbak, mint az ember. Viszont közvetlenül nem képesek kommunikálni az emberekkel, hisz egy számítógép nem tud egyik emberi nyelven sem, mint ahogy nekünk sem egyszerű megérteni a számítógépek nyelvét. Szerencsére a programozási nyelvek megoldást nyújtanak ennek a problémának a megoldására.

A programozási nyelveknek saját szabályaik, szintaxisuk és funkcióik vannak. Ebben hasonlítanak az emberi nyelvekhez. Fontos, hogy egy programozási nyelv a programozni tanulók számára érthetővé válik, így biztosít könnyebb kommunikációt a számítógéppel. Általuk a fejlesztők írhatnak utasításokat, melyek sorozatából fel is épülnek a programok és az alkalmazások, amelyek képesek végrehajtani különféle feladatokat.

# **TypeScript**

A fejlesztés során kiemelt fontosságú, hogy milyen programozási nyelvet használunk, ugyanis előnyt jelenthetnek bizonyos programozási nyelvek az egyes feladatokat megvalósító alkalmazások elkészítésében. A TypeScript pedig rendkívül jól illeszkedik a Frontend fejlesztést segítő keretrendszerekhez, mint például az Angular, amit jelenleg használtam. A webalkalmazás logikája ezen a nyelven íródott így tehát több, mint a fejlesztés felét kitette ennek a nyelvnek a használata.

A TypeScript egy nyílt forráskódú, statikusan típusos objektumorientáltként is használható programozási nyelv, mely a JavaScript interpretált programozási nyelv hiányosságait terjeszti ki. Javarészt a Microsoft csapata fejlesztette, hogy megkönnyítsék a programozók munkáját. A TypeScriptet a fordító először JavaScript kóddá fordítja át, melyet már képes értelmezni a böngésző.

# **HTML**

A HTML (Hipertext Markup Language) egy leíró nyelv, ami a weboldalak struktúrájának felépítésére szolgál. Az oldal csupasz vázát felépíthetjük a HTML által előre definiált tag-párokkal, melyek egymásba ágyazásával egyedi struktúrát alakíthatunk ki. Értelemszerűen a Frontend fejlesztést segítő keretrendszereknek szüksége van valami ehhez hasonlóra. Többek között az Angular is a HTML használatát igényli azon felül, hogy valamennyi logikát is bevihetünk a használatával a HTML kódba.

# **SCSS**

Az SCSS (Sassy Cascading Style Sheet) egy stílusleíró nyelv, ami kiterjeszti a CSS-t, melybe előre definiált kulcs-érték párok megadásával tudjuk a HTML által felépülő struktúrának a megjelenését változtatni. Az SCSS könnyebb és hatékonyabb CSS stíluslapok írását teszi lehetővé, hisz kisebb logikai funkciókat biztosít, mint például a változók létrehozása. Egy preprocesszor dolgozza fel az SCSS fájlokat a fordítás részeként, és így hoz létre a háttérben CSS fájlokat.

# **Tailwind**

A Tailwind CSS egy modern és rendkívül testre szabható CSS keretrendszer, amelyet kifejezetten a gyors és hatékony webfejlesztéshez terveztek. A Tailwind ideológiája az, hogy a CSS osztályok közvetlenül definiálják a stílust, így minimalizálva az előre definiált stílusokat és maximalizálva a fejlesztői rugalmasságot. Az alacsony szintű osztályok lehetővé teszik a pontos kontrollt az elrendezési és stíluselemek felett, miközben könnyen testre szabhatók és újrahasznosíthatók. A Tailwind széles körben elterjedt a fejlesztők körében, akik értékelik a gyors és intuitív munkafolyamatot, valamint a könnyen karbantartható és szabványos kimenetet.

# **Keretrendszerek**

A keretrendszerek arra szolgálnak, hogy a funkcióikat felhasználva lerövidítsék a fejlesztésre szánt időt. Nagyobb projektek esetén rendkívül hasznos, hisz ahogy a nevéből is adódik egy keretet ad az egész projektnek. Lényegileg olyan funkciókat tartalmaznak, melyeket alap esetben szintén le kellene fejleszteni. Viszont a keretrendszerek adottak, és így bizonyos dolgok fejlesztésén időt spórolhatunk. Ennek a haszna, hogy sokkal inkább koncentrálhatunk azon területek implementálására, mely az adott alkalmazásunk lényegét képezi.

# **Angular**

Az általam is használt nyílt forráskódú webes keretrendszer az Angular, melyet a Google fejlesztett ki, és adott ki 2010. október 20-án még Angular JS néven modern webalkalmazások fejlesztésének céljából. A célja, hogy felhasználóbarát, jól skálázható, hatékony alkalmazást fejlesszünk általa. Eleinte a JavaScript nyelvet használta a logikájához, de már az Angular 2-től áttért a JavaScriptet kiterjesztő TypeScript nyelv használatára. Biztosít adatkötést a felhasználói felület és a logika között, könnyű navigációt a Router segítségével, és a sablonok segítségével dinamikus és interaktív felhasználói felületek hozhatóak létre. Komponensekbe, avagy modulokba szervezhetjük az alkalmazásunkat a jobb átláthatóság végett. A segítségével nem csak webalkalmazások építhetőek fel, hanem például mobilalkalmazások, de többek között még asztali alkalmazások is. Ezen keretrendszer 14-es verzióját használtam fel a webalkalmazás elkészítésekor.

# **Firebase**

A Firebase a Google által kifejlesztett népszerű és teljeskörű fejlesztői platform. A gyors, hatékony fejlesztésben, tesztelésben, sőt az üzemeltetésben is nagy segítségére van a fejlesztőknek. Az eszközei és szolgáltatásai rengeteg segítséget nyújtanak. Idetartozik a teljesség igénye nélkül a felhasználókezelés és hitelesítés, a felhőalapú tárolás, a valós idejű adatbázis, de akár még az alkalmazásnaplózás is. A Firebase könnyen integrálható webes-, avagy mobilalkalmazásokba egyaránt. Az Angular keretrendszer a TypeScript által könnyen képes alkalmazni a Firebase-t ezzel megkönnyítve a programozási munkálatokat.

# **Terület áttekintés**

# **3.1. Webshop alkalmazások terület áttekintése**

A webshopok afféle digitális központok, ahol a fogyasztók könnyedén böngészhetnek, vásárolhatnak, és az árukat a kényelmükből kifolyólag otthonukba szállíttathatják anélkül, hogy el kellene hagyniuk a házat. Ez lényegi hasonlóság minden webshop között, így az én megoldásom is erre épül. Ezen dinamikus online kereskedelmi platformok minden egyes szegmense fontos szerepet játszik a felhasználói élmény, az üzleti hatékonyság és az innováció terén.

A terület áttekintésének fontos része maga a marketing. A webshopok különféle módszerekkel igyekeznek vásárlásra bírni a fogyasztói társadalom tagjait. Személy szerint olyan egyedi megoldásokat választottam, melyek újító jelleggel hatnak a webáruházakra. Ugyanis a piackutatás elvégzése során nem találtam máshol beépített játékot, avagy blogírási lehetőséget. A marketing kontextusában a terület nemcsak a földrajzi helyet jelenti, hanem kiterjed a digitális térre is, ahol a versenytársak, a fogyasztók és a technológia együttesen formálják az online kis- és nagykereskedelmet.

A webshop területáttekintése nem csupán statisztikák összegyűjtése és táblázatok elkészítése a versenytársakról. Fontos volt átemelni a már jól bevált módszereket mások által épített alkalmazásokból. Ilyen a könnyű böngészés, a kifinomult felhasználói élmény kialakítása, a gördülékeny vásárlási lehetőség. Ez egy átfogó elemzés vizsgálatát takarja, mely a piaci trendek, fogyasztói szokások, és az aktuális technológiai fejlemények feltárására irányul. A cél nem csupán az volt, hogy ismereteket szerezzek a jelenlegi piaci helyzetről, hanem hogy ezekre az ismeretekre alapozva a webshopom maximális hatékonysággal és innovációval szolgálhassa ki a vásárlókat.

A következő alfejezetekben részletesen fogunk foglalkozni a webshopok által kínált lehetőségek tárházával, és megvizsgáljuk, hogyan járult hozzá a virtuális italokat árusító webshopom fejlesztéséhez és sikeréhez a világháló már használatban levő webáruházainak részletes kielemzése. A piackutatás során szerzett ismeretek és az áttekintés során feltárt lehetőségek összekapcsolásával teremtettem meg azt a szilárd alapot, amelyen az online kiskereskedelem sikeresen megélhet és fejlődhet a digitális térben.

# **3.2. Piackutatás**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Saját alkalmazás tervek** | **Kormorán Bottleshop** | **DrinkCentrum** | **TotalDrinks** | **Drink Online** |
| Linkek: | <https://trinkydrinky-webshop.web.app/> | <https://alkoholshop.hu/> | <https://drinkcentrum.hu/> | <https://totaldrinks.hu/> | https://www.drinkonline.eu/ |
| Interneten elérhető | Igen | Igen | Igen | Igen | Igen |
| Üdvözlő/Életkor ellenőrző üzenetek implementálása | Nem | Igen | Igen | Igen | Nem |
| Egyedi témák | Igen | Igen | Igen | Igen | Igen |
| Regisztrációs lehetőség | Igen | Igen | Igen | Nem | Nem |
| Böngészés a termékek között | Igen | Igen | Igen | Igen | Igen |
| Alkoholmentes italok forgalmazása | Igen | Nem | Igen | Igen | Igen |
| URL átirányítások | Igen | Igen | Igen | Igen | Igen |
| E-mailben való értesítések rendelés esetén | Nem | Igen | Igen | Igen | Igen |
| Regisztráció nélküli rendelés lehetőség | Igen | Igen | Igen | Igen | Igen |
| Megjegyzés/Komment írás a termékekhez | Igen | Igen | Nem | Nem | Nem |
| Értékelés a termékekhez | Igen |  |  |  |  |
| Saját blog bejegyzés írása | Igen | Nem | Nem | Nem | Nem |
| Termékhez hasonló termékek megjelenítése | Igen | Igen | Igen | Igen | Nem |
| Újdonságok gyors elérése | Nem | Nem | Igen | Nem | Nem |
| Webshop értékelésének megtekintési lehetősége | Nem | Igen | Nem | Nem | Nem |
| A vásárláshoz Facebook fiók használata | Nem | Nem | Nem | Nem | Igen |
| Többféle betűtípus alkalmazása több szín mellett | Igen | Nem | Nem | Igen | Igen |
| Logók, képek szereplése | Igen | Igen | Igen | Igen | Igen |
| Többnyelvűség | Nem | Igen | Nem | Nem | Nem |
| Beépített játékkal kedvezmény szerzése | Igen | Nem | Nem | Nem | Nem |

# **3.3. Piackutatás értelmezése**

A piackutatás alapján értelemszerűen kivehető, hogy mindegyik webshop megtalálható az interneten. Ez a lényegi értelme az összesnek, hogy kiszolgálja az arra tévedő felhasználókat a saját termékeikkel.

Mivel az alkohol vásárlás egy olyan tevékenység, melyhez szükséges az, hogy a vásárló életkora elérje a 18-at, ezért szokványos, hogy az adott webshop az oldal megnyitásakor felteszi a kérdést, hogy felnőttek vagyunk-e már. Ugyanis ez jogosulttá tesz minket arra, hogy szeszes italt fogyaszthatunk. Ezen felül vagy ehelyett mindenféle (nem felnőtt árucikkeket tartalmazó oldalak esetén is) szokás egy felugró üdvözlő ablak megjelenése, amely szimplán csak köszönti a lehetséges vásárlót. Ez személyesebbé teheti az élményt, ugyanis egy boltban való vásárlás esetén is lezajlik egy oda-vissza köszönés a fogyasztó és az eladó között.

Minden webshop habár ugyanazzal a céllal készül, még is egyedi tematikát, egyedi stílust, egyedi megjelenést kölcsönöz, amellyel megpróbálja felkelteni a lehetséges vásárlói figyelmét. Sokszor megfigyelhető, hogy egy-egy webshop teljesen máshogy épül fel, mint a többi. Ennek a gyakorlati haszna sokszor tényleg a figyelemfelkeltés, emellett a minél gördülékenyebb böngészés megteremtése, melynek többféle megoldása is felhasználóbarát élményt teremthet.

A böngészés a termékek között sokszínű. A leghatékonyabb az, ha könnyebben elszeparálhatóak egymástól a termékek. Akár kategóriák, színek, árak, név szerint. A név szerinti szűrés a legpontosabb, ugyanis lehetőségünk nyílik ténylegesen csak arra a termékre keresni, amit kívánunk megvásárolni.

A regisztráció és ezáltal a bejelentkezés lehetősége sokszor új kapukat nyit meg a fogyasztók előtt. Vannak webshopok, ahol csak a regisztrációt követően van lehetőségünk rendelni, valahol el tudjuk menteni bankkártya adatainkat, vagy esetleg kedvezményekhez férhetünk hozzá. A regisztráció egyfajta autentikációt biztosít, mely által könnyen azonosíthatják magukat a fogyasztók.

A megjegyzések írása, és az értékelések nagyban segítik a webáruházakat abban, hogy visszajelzéseket kapjanak a termékeik milyenségéről. Ennek segítségével tudják, hogy mi az, ami bevált, és mi az a termék, amin módosítani kell, esetlegesen teljesen el kell törölni. A vásárlók között kialakuló kommunikáció arra is szolgál, hogy egymást megsegítsék a termék milyenségét, minőségét, hasznosságát illetően. Sok esetben a megjegyzések és értékelések hozzáadása is a regisztrációt követően elérhető funkció, de egyesesetekben lehetőség van akár anonim hozzáfűzésre, értékelés nyújtásra is.

Saját blog bejegyzés írása ugyan nem szokványos egy webáruház esetén. Viszont úgy gondoltam, hogy ez színesítheti, és feldobhatja azt. Ez a funkció csupán arra szolgál, hogy plusz információhoz juttassa a felhasználókat, esetleg bővebben kifejthessék gondolataikat összeszedettebb formában.

A termékek keresését illetően nagy segítségre lehet az, hogy a megnyitott termék oldalán megjelennek hozzá igencsak hasonlító más termékek is. Ezzel motiválva lehetnek a fogyasztók esetleges új termékek megismerésére. Így magasabb vásárlási arányra tehet szert a webshop tulajdonosa.

Néhány weboldal kiemelt fontosságúnak tartja az újonnan érkező termékek nyomatékos feltüntetését. Általában jól látható helyeken helyezik el, és szinte egyből az oldal megnyitásakor találkozhatunk velük. Az új termékeknek még szüksége van marketingre, hogy az információ a létezésükről elterjednek így növelve a kínálatot.

Az értékelés nem csak egyes termékekre irányulhat. Az én esetemben például lehetőség nyílik a blog bejegyzések értékelésére is, amivel láthatjuk, hogy a benne található információ mennyire releváns, hasznos. Így arra sarkallva az oldalra tévedőket, hogy érdemes-e elolvasni egy adott bejegyzést vagy sem. Más weboldalakon olyan funkciót is fellelhetünk, mellyel a teljes rendszer egészét értékelhetjük. Így szintén mások segítségére lehet, hogy az adott webshop mennyire hiteles, mennyire jók a termékei.

A vásárlás indításához a legtöbb esetben szükséges létrehoznunk egy fiókot a weboldal felületén. Ebben segít a regisztráció. Máshol akár fizethetünk úgy is, hogy csak egyszeri megoldásként adjuk meg a fizetéshez szükséges adatainkat. Hasznos módszert biztosít az, amikor nem terheljük a felhasználókat azzal, hogy újdonsült fiókot hozzanak létre. Csupán csak hozzákapcsolják az egyik közösségi média fiókjukat, mint például a Facebook-ot, avagy a mindenki által előnyben részesített Gmail fiókot.

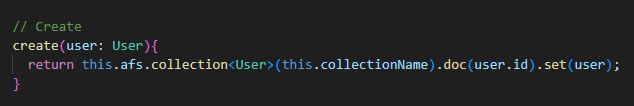
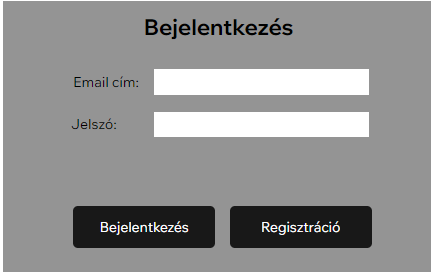
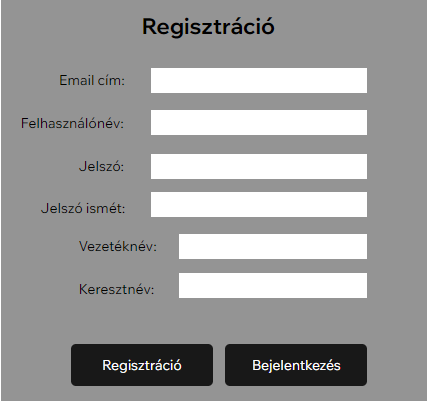
Az egyedi témákhoz kapcsolódóan a weboldalak rengeteg lehetőséget kínálnak felépítésük folyamán. A készítőnek hatalmas paletta áll rendelkezésére színek, betűtípusok, és más dizájn elemek használatát illetően. Ezek szintén arra szolgálnak, hogy olvashatóbbá, felhasználóbaráttá, egyszerűen jól kinézővé tegyék a szemünk elé táruló oldalakat. Ide tartoznak a weboldalt színesítő képek is, ugyanis sok esetben egy jól elkészített kép, és annak csábító emblémái, logói, egyéb díszei felhívják a figyelmet. Egy italárusító webshop esetén a termékek íze csak akkor derül ki, amikor már megvásárolták őket, viszont a kinézetük sokszor döntő fontosságú lehet a fogyasztók szemében. Mindemellett a kisebb, könnyen elhelyezhető logók könnyű marketing fogások.

Fontos kiemelni, hogy a weboldalak látogatói milyen nyelven beszélnek, milyen írott szövegeket értenek meg. Kiváló lehetőség az, hogyha lehetősége van a felhasználónak egyetlen kattintással módosítani a kívánt nyelvre a weboldal tartalmának teljes egészét. Ugyanis a megértés döntő fontosságú a rendelések számát illetően. Senki nem fog megrendelni olyan terméket, amiről azt sem tudja mi az. Habár az internet korában főleg otthoni rendelés esetén már könnyen fordíthatóvá válnak az egyes szövegrészletek a különféle fordítóprogramok segítségével. Azonban a fogyasztói társadalom tagjai rendkívül türelmetlenek és lusták tudnak lenni ezen téren. A többnyelvűség viszont megoldást nyújt erre a problémára. Lehetővé teszi, hogy a felhasználók kedvük szerint böngészhessenek akár a saját, akár azon a nyelven, amit ismernek. Ezért célszerű az angol nyelvet alapul venni itt Európában, ugyanis közel ez a legismertebb, legtöbb ember által beszélt nyelv.

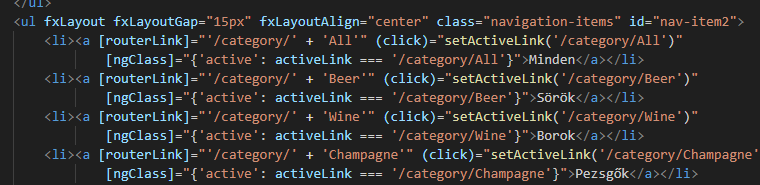
A webshopok nem szokványos eleme, hogy tartalmazzanak játékokat. Véleményem szerint viszont nagyszerű marketing fogás, amivel az oldalunkra csábíthatjuk nap, mint nap a fogyasztók egy bizonyos hányadát. Főképp, ha jutalom is társul a játékkal elvesztegetett perceinkért. A játékok köztudottan addikciót válthatnak ki egyes emberekből, és ezt az addikciót kihasználva feltételezhető, hogy akár a játék élvezete hatására megnő a piaci kereslet a termékek irányába. Ez nem egy elismert módszer, sőt nem is alkalmazott, viszont egy próbálkozást megér, hogy hogyan hat ez a fogyasztói réteg gondolkodására. Vajon ezen funkció tényleg meghozza a várt hatást, avagy csak plusz idegesítő elem a vásárlási folyamatban?

# **Funkcionális specifikáció**

# **4.1. Bejelentkezés és regisztráció**

 A Firebase és a hozzá kapcsolódó Angular-os technikák segítségével könnyedén megoldható az autentikáció egy webalkalmazás esetében. A Firebase gördülékenyen megjegyzi az újonnan regisztrált felhasználókat az Authentication fülön. A jelszavakat hash-elt formában tárolja el így nem tudjuk őket visszafejteni, szinte lehetetlen, így az alkalmazásunk biztonságosabb lesz az adathalászokkal szemben. A Firebase egy egyedi azonosítót (User UID) is hozzárendel minden újonnan regisztrált felhasználóhoz. Mindezt a kódban egy külön Service-be kiszervezve oldottam meg néhány függvény segítségével, mint ahogy az az ábrán is látható (bejelentkezés, regisztráció, kijelentkezés). Ezen felül a Firestore Database-ben még egy külön kollekció is rendelkezésre áll, ahol a felhasználókról egyéb adatokat is le tudunk tárolni dokumentumok formájában. Természetesen adatvédelmi szempontból a jelszót ekkor sem érdemes csak úgy eltárolni. A kódban ez úgy szerepel, hogy van egy külön modul létrehozva a regisztrációnak és a bejelentkezésnek is egyaránt, amelyek így külön oldalként jelennek meg, és igencsak könnyen használhatóak. A bejelentkezésben egy gombhoz kötött függvény hívja meg az authService login függvényét, és így történik meg a hitelesítés. A regisztrációhoz már arra is szükség van, hogy egy másik Service-t használjunk, melyben a CRUD (Create, Read, Update, Delete) műveletek vannak megvalósítva User objektumokra vonatkozólag. A create művelet segítségével a Firebase Database megfelelő kollekciójában is megjelenik az újonnan regisztrált felhasználó, és annak adatai azon felül, hogy az Authentication fülön is megtalálható lesz. Képernyőtervek a felületekről:

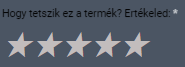
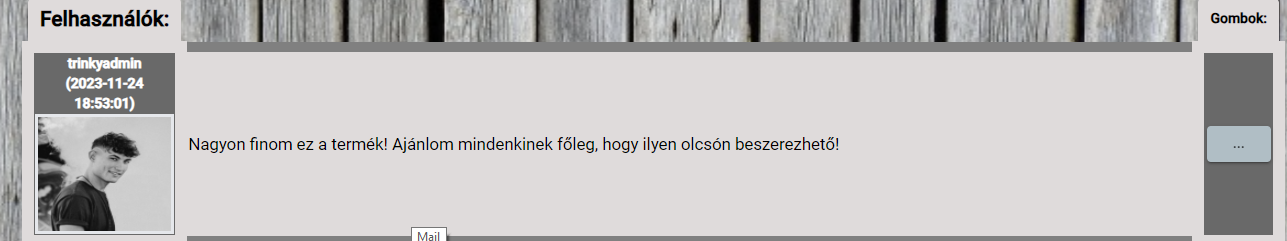
# **4.2. Böngészés a termékek között**

A webáruházak elsődleges célja, hogy eladják termékeiket, és ezáltal profitot termeljenek. Elengedhetetlen, hogy a vásárlók könnyedén tudjanak böngészni közöttük. Ezen okból kifolyólag alakítottam ki a menüsávon egy részt, melyen a különféle italok kategóriái között könnyedén lehet navigálni. Az ábra szemléltet is néhányat a kódból a teljesség igénye nélkül. A főoldalon alapból megjelenő hamarosan érkező, új, akciós és kifutóban levő termékek vannak szerepeltetve, melyek felhívhatják a vásárlók figyelmét. Emellett ezen az oldal design és helytakarékosság szempontjából a termékek között lapozgatni lehet, így megkeresve a számunkra legszimpatikusabbat. A kategóriák oldal és a főoldal is egy-egy különálló modulban helyezkedik el. Amennyiben kiválasztunk egy számunkra szimpatikus kategóriát minden termék, amely az adott kategóriába tartozik megjelenik a képernyőnkön bármiféle rendezési logika nélkül. A fogyasztónak lehetősége nyílik abc sorrendbe, és annak fordítottjába vagy akár ár szerint növekvő, avagy csökkenő sorrendbe helyezni a termékeket. Így sokkal gördülékenyebbé válik a vásárlás. Kategória sáv a menüben a kiválasztott jelzésével: 

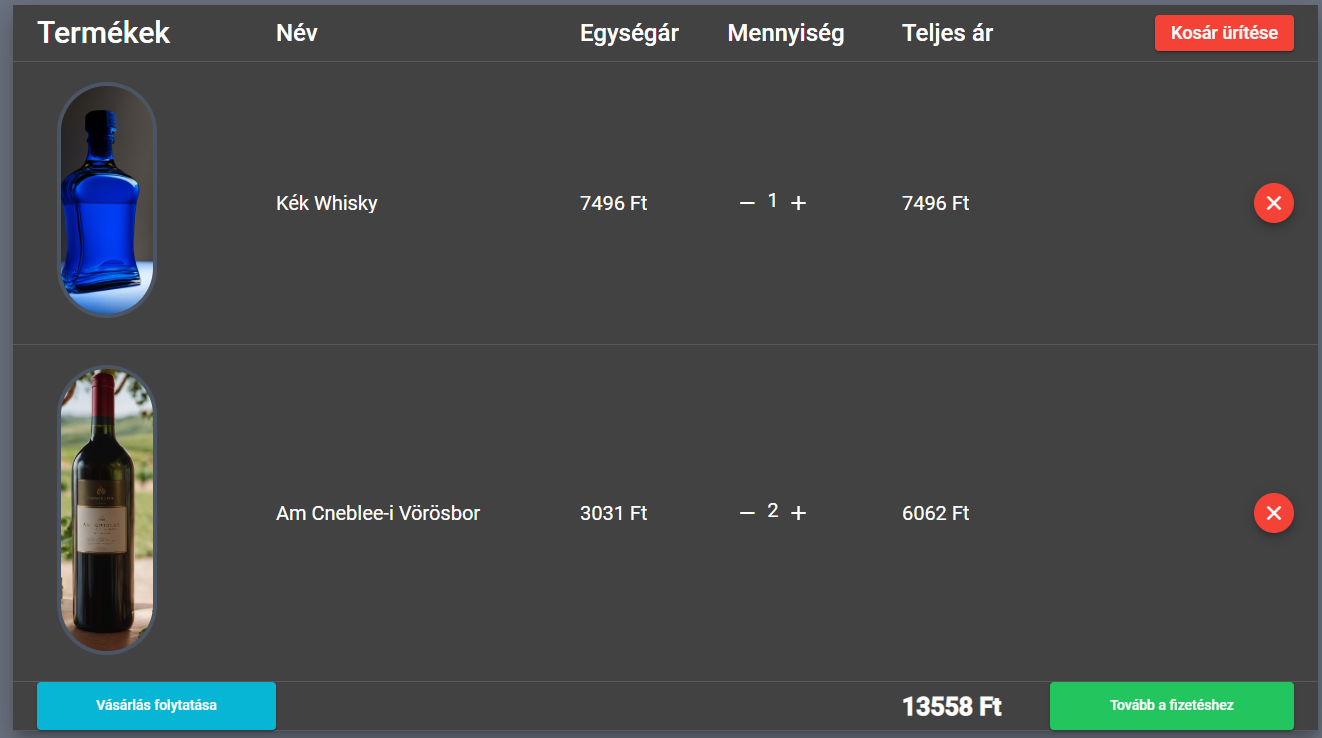
# **4.3. Termékek részletei**

Kiemelt prioritással bír, hogy az egyes termékek rendelkezzenek felhívó leírással, kedvező árral, és minden fontos információval, mely arra sarkallja a felhasználókat, hogy azonnal rendeljék meg az adott terméket, hisz mindenképp jól járnak vele. Ezen okokból kifolyólag a termékekre kattintva egy új lap tárul elénk, melyen csak a fókuszba helyezett termék képe szerepel nagyobb változatban, emellett sok érdekes információ róla, mint például egy frappáns leírás. A kiválasztott termékhez hasonló termékek kisebb képei is megjelennek ezen az oldalon, amik szintén csak könnyítik a böngészést, emellett segítik az újdonsült termékek megismerésének folyamatát. A felhasználók felé visszajelzésként egy „HAMAROSAN ELFOGY” feliratú kiemelt szöveg tárul, amennyiben már csak kevés van az adott termékből, de még az is jelzésre kerül, ha egy termék éppen nincsen raktáron. A részleteket vizsgálva van lehetősége a fogyasztónak többféle módon értékelni az adott terméket.

# **4.4. Felhasználói visszajelzések**

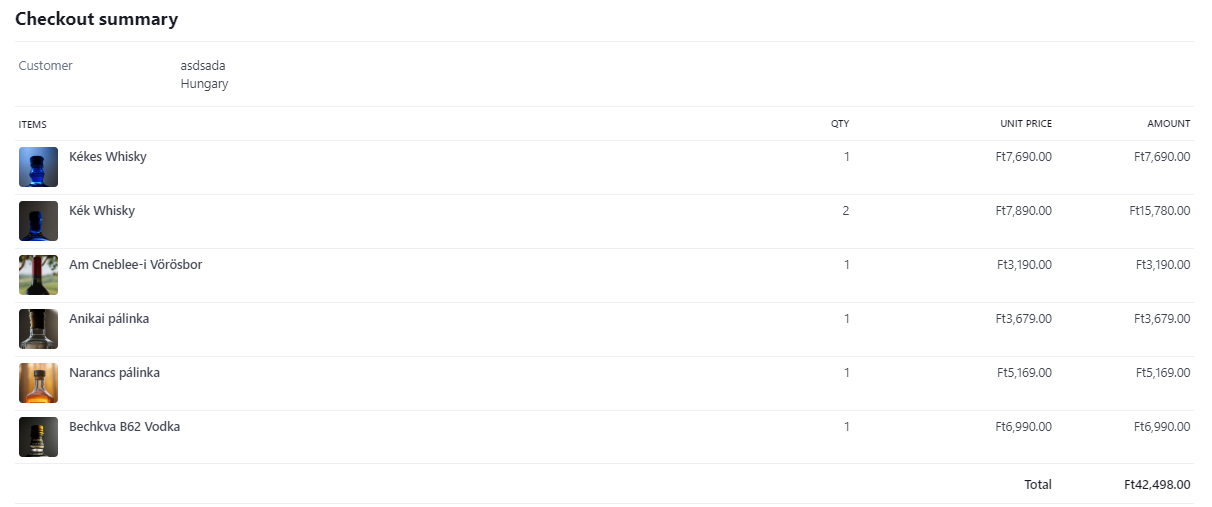
Az előbb említett felületen lehetőségük van a vásárlóknak hozzászólni az adott termékhez. Így visszajelzéseket küldhetnek minden egyes termékről a webáruház üzemeltetője felé. Ezek a kommentek szintén letárolódnak a Firebase Database-ben egy kollekcióba, mint ahogy a termékek és a felhasználók adatai. Minden felhasználó felhatalmazást kap arra, hogy a saját maga által írt hozzászólást bármikor eltávolítsa vagy frissítse. Az admin felhasználóknak jogukban áll bírálni bizonyos kommenteket, ezért törölhetik őket attól függetlenül, hogy ki a szerzője. Az aktív adminok így kulturált közösséget teremthetnek, avagy az üzemeltető szempontjából, habár átverés, de hasznos lehet a negatív visszajelzésű kommentek kiszűrése, és eltávolítása. Az ilyesfajta értékelések azért relevánsak, mert az üzemeltető láthatja, hogy melyik az a termék, amelyre nagyobb az igény, melyik az, amire nem. Természetesen nyilván tarthatjuk a szállítási mennyiségeket ezen felül, hogy még kitisztultabb képet kapjunk az adott árucikk fogyasztását illetően. Emellett a vásárlók a pozitív visszajelzéseikkel egymást is vásárlásra ösztönözhetik, ugyanis egy olyan termék mely sok pozitív megjegyzéssel van elárasztva nagyobb valószínűséggel vonzz be új vásárlókat. Másik lehetőség az értékelésre, hogy a felhasználók egy pontozási rendszerben tudják értékelni a nekik tetsző, vagy épp nem kívánatos termékeket. A skála 1-től 5-ig terjed, melyet csillagok jeleznek. Minden felhasználó minden egyes termékre csak egy értékelést nyújthat be, viszont ezt a későbbiekben lehetősége van módosítani. Az összes felhasználó egyes termékekre adott értékeléseinek átlaga a termék részleteinél megtekinthető. Kép az értékelésről és egy kommentről: 

# **4.5. Kosárkezelés**

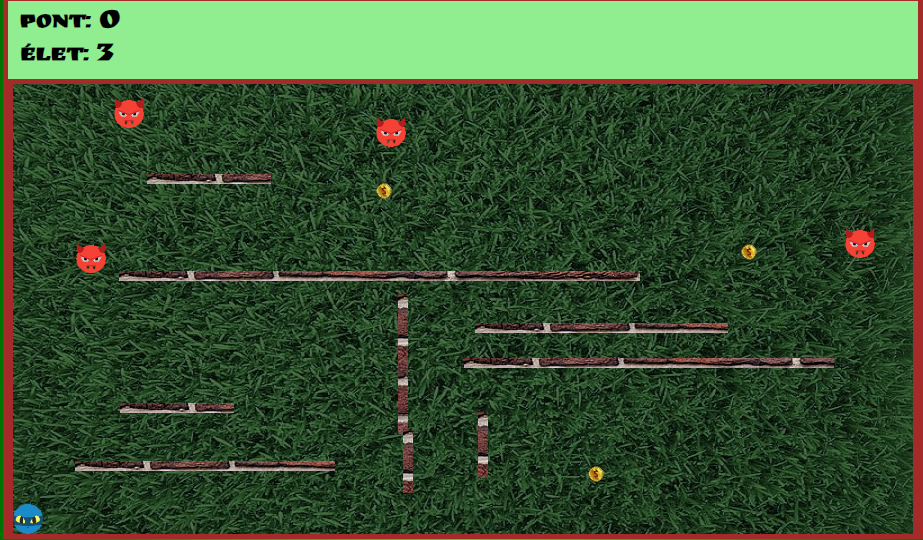
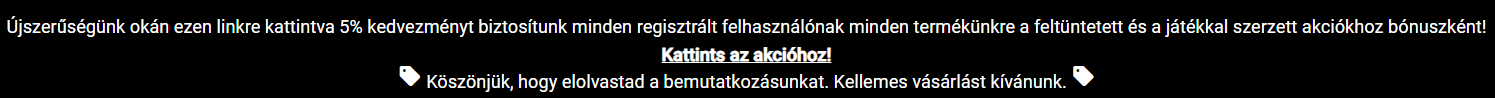
A felhasználóknak a minél kényelmesebb vásárlás érdekében több helyen is elérhetővé kell tenni a kosárba tétel lehetőségét. A főoldal, a kategóriák között való böngészés és az egyes termékekre kattintás után is lehetőség nyílik arra, hogy a kívánt darabszámú terméket elhelyezzük a kosarunkban. A sikeres kosárba helyezést egy toastr segítségével jelezzük is a felhasználók felé, mindamellett, hogy a kosár ikon a menü sávban jelzi a benne elhelyezett termékek darabszámát. Ilyenkor az elhelyezett termékeket a localStorage-ban is letárolom, hogy amennyiben a felhasználó ráfrissít a weboldalra az addig kiválasztott árucikkek ne vesszenek el a kosárból. A kosár egy önálló modul, ami mindig az adott felhasználóra vonatkozóan tárolja az éppen elhelyezett termékeket. A bevásárlókocsi ikonra kattintva elnavigálhatunk arra az oldalra, ahol a fizetést is lebonyolíthatjuk. Itt látjuk összesítve az elhelyezett árucikkek árát, és természetesen még módosíthatunk ezen. Lehetőségünk nyílik csökkenteni és növelni az elhelyezett árucikkek darabszámát, eltávolítani egy adott terméket a listából, törölni a teljes bevásárlókocsink tartalmát, visszanavigálni a főoldalra a további vásárlás érdekében, de akár már fizethetünk is egyetlen gomb megnyomását követően. Kosár kinézete a weboldalon:

# **4.6. Integrált fizetési rendszer**

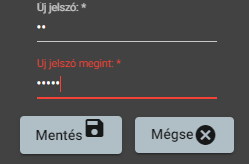
A webáruházban kizárólag bankkártyás fizetésre van lehetőség, melyet a kosár alapos megvizsgálása után érhetünk el. A Stripe fizetési rendszer beépítése teszi lehetővé a vásárlást. Mivel ez nem egy valódi webshop, ezért természetesen tesztkártya adatokkal nyílik lehetőségünk fizetni, mint például a (424242424242-es számú bankkártya). Ilyenkor egy külön apiból származó oldal tárul a szemünk elé, ahol a fizetési adataink megadása után (amik természetesen továbbra is hamis adatok, kisebb ellenőrzésekkel, mint például az email cím formátumának ellenőrzése, a kártya lejárati dátumának valószerűsége oly módon, hogy bármilyen jövőbeli dátumot tüntetünk fel) a „Fizetés” gombra kattintva visszajelzést kapunk a vásárlás sikerességét illetően. A Stripe rendszer a Firebase Cloud Functions használatával van beüzemelve, ugyanis habár az Angular 100%-ban egy frontend keretrendszer, itt szükség van a backend továbbfejlesztésére a Firebase alapvető funkcióin túl. Egy Express szerver működése teszi lehetővé azt, hogy az interneten elérhető hostolt alkalmazásban is (anélkül, hogy lokálisan bármilyen szervert működtetni kellene) megjelenjen a fizetési felület. A sikeres fizetést követően a Stripe hivatalos oldalán belépve meg is kapjuk a vásárlás adatait, melyeket továbbítani szerettünk volna. A megadott e-mail cím, a bankkártya titkosítva, a termékek képei, nevei, árai és az összár feltüntetése könnyen beazonosíthatóvá tesz egy adott rendelést azon túl, hogy természetesen az időpont is szerepel a megjelenő sorban. A képeken látható egy példa rendelés eredménye a Stripe-ból, a feltüntetett rendelési adatokkal, és egy másik példa, hogy hogyan vannak feltüntetve az egyes megrendelt termékek adatai, és a forintban értendő teljes ár is. A felhasználó természetesen ezt nem láthatja, azonban visszajelzésként egy újabb oldalra navigálódik melyen megjelenik, hogy a vásárlása sikeres volt, emellett a saját email címem, amelyre kattintva lehetősége nyílik az ügyfélnek megosztani az észrevételeit a vásárlással kapcsolatban. Ebben az esetben a kosár is kiürül, emellett a termékek darabszáma is csökken a megvásárolt mennyiségek darabszámával.



# **4.7. Kedvezmények szerzése (Beépített játék, egyszer használatos link)**

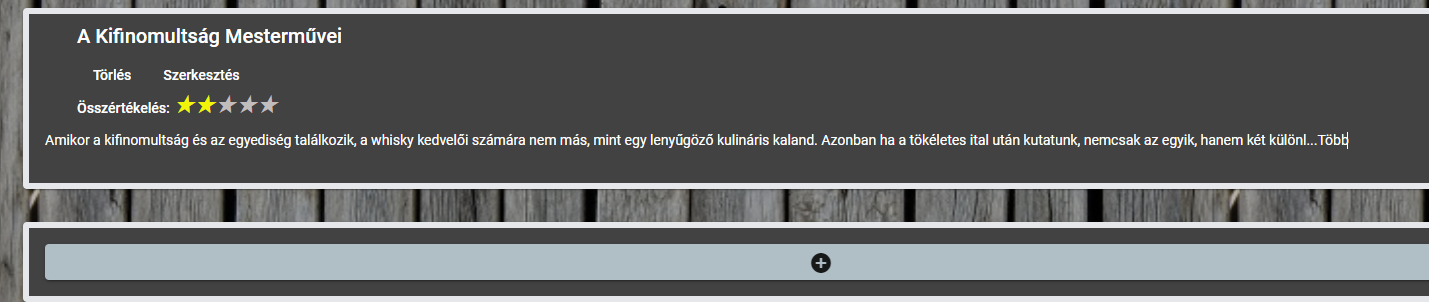
A játék egy újabb modul a rendszer teljes egészét illetően. A felhasználónak előbb be kell jelentkeznie, hogy ez a funkció elérhetővé váljon. Amennyiben egy odatévedő felhasználó rákattint a „Játék” gombra a menüsávban a jól kiépített routingnak hála a bejelentkezési oldalon találja magát, ahol egy figyelmeztető üzenet is felvillan, hogy a játék opció, csak bejelentkezett felhasználók számára érhető el. Ezen funkció egy nagyon egyszerű, a pacman mintájára készült egypályás kalandozást tesz lehetővé, ahol a feladat csupán annyi, hogy mindhárom pénzérmét felszedjük főhősünkkel. Természetesen akadályokba is ütközhetünk, melyeken nem tud áthaladni a játékos, csakúgy, mint a játéktér falának ütközve sem tudja elhagyni a pályát. Ezen falak elhelyezkedése a véletlenre van bízva, ugyanis random szám generátorok segítenek a pálya felépítésében. Ez nem minden, ugyanis gonosz ellenségek sokasága is megfertőzi a térképet, akiknek elhelyezkedése, darabszáma, sőt még a mozgásuk milyensége is teljesen randomizált folyamat. Ha szerencsések vagyunk, vagy elég türelmesek, és elégszer frissítünk rá a weboldalra akkor kaphatunk egyszerűen végig vihető pályákat is. A nehézségi fok csupán a játékos szerencséjén, avagy türelmén múlik. Minden regisztrált felhasználó 3 életet kap a játékhoz, amely életek minden 24 órában újra jóvá íródnak a fiók számláján. Ha nekiütközik egy ellenségnek, vagy ő ütközik nekünk akkor veszítünk egy életerő pontot, visszakerülünk a kiindulási pozícióba, ami a bal alsó sarok, és elveszítjük az addig már megszerzett pénzérméket, amelyek szintén random helyeken újra generálódnak a pályán. A játék akkor ér véget, ha elveszítjük az összes életpontunkat, avagy megszerezzük mind a három pénzérmét. Amennyiben elbuktunk akkor nem kapunk semmiféle kedvezményt, viszont, ha sikeresen teljesítjük a pályát, akkor 3% kedvezményre leszünk jogosultak, mely minden termékre érvényes. Nincs lehetőségünk az életerő pontokat gyűjteni, ha sokáig nem játszunk, mindig 3 lehet a maximális a számlánkon. Ehhez a folyamathoz szintén a Firebase Cloud Functions használatára volt szükség, ahol egy 24 óránként lefutó függvény automatikusan frissíti az összes fiók életerő pontját 3-ra, amennyiben kevesebb volt ennél az értéknél. A megszerzett kedvezmény minden 24 órában lenullázódik, ami arra sarkallja a vásárlókat, hogy mindennap amikor csak rendelni szeretnének újra játszanak. Mivel a fogyasztókat regisztrálásra szeretnénk sarkallni ezért van egy másik lehetőség is, ahol 5% kedvezményt szerezhetünk minden termékre. Így már összesen 8% kedvezmény kerülhet jóváírásra minden egyes termékre a regisztrált, s ügyesen játszó felhasználók javára. Az 5% kedvezmény megszerzése érdekében csupán csak a „Rólunk” fülön megtalálható linkre kell kattintanunk, amely bejelentkezetlen fogyasztók esetén a routingnak köszönhetően szintén a bejelentkezési felületre navigál. A kedvezmények biztosítása és fenntartása növelheti a vásárlói elégedettséget. A képeken látható az egyszer használatos link, és a játék végleges formái.

# **4.8. Felhasználói fiók kezelése**

A felhasználói fiókok kezelése kulcsfontosságú része egy weboldal életének, hiszen ez az interaktív felület hozza létre a személyes kapcsolatot a vásárlók és a webshop között. A „Profil” fül, mely bejelentkezést követően érhető el, egy átfogó modult kínál minden felhasználónak a fiókjuk személyre szabására és naprakészen tartására. A jelenleg használt e-mail cím, felhasználónév, vezetéknév és keresztnév mellett megtekinthetik az aznapra már megszerzett kedvezményüket is, melyet százalékban kifejezve láthatnak. Az itt elérhető információk átláthatóvá teszik a felhasználók számára a fiókjukkal kapcsolatos részleteket. A felhasználóknak lehetőségük van a kevésbé kritikus adatok, mint például a felhasználónév, vezetéknév és keresztnév módosítására, ami rugalmasságot biztosít a személyes beállításaik terén. Az említett adatok egyszerű módosítása mellett a weboldal fejlett biztonsági intézkedéseket kínál a jelszóváltoztatásra is. Bár ez egy összetettebb folyamat, de kulcsfontosságú, hogy a felhasználók számára biztosítsunk egy hatékony és biztonságos módot jelszavuk frissítésére. A profilkép feltöltése további lehetőséget kínál a felhasználóknak az egyedi azonosításra. Ezen képek bármikor módosíthatóak, mely szintén hozzájárul a felhasználói fiók naprakészen tartásában. Ezek minden komment mellett megjelennek, így a felhasználók könnyen felismerhetővé válnak a közösség számára. Ezáltal a webshop nem csupán egy online vásárlási platform, hanem egy olyan közösségi tér, ahol az azonosítást követően a blogbejegyzések és kommentek írásával a személyes élmények is megoszthatóvá válnak. A felhasználói fiókok, avagy a profilok kezelése így nem csupán egy adminisztratív funkció, hanem egy olyan eszköz, amely közelebb hozza az ügyfeleket egymáshoz és a weboldal egészéhez.

# **4.9. Bejegyzések írása**

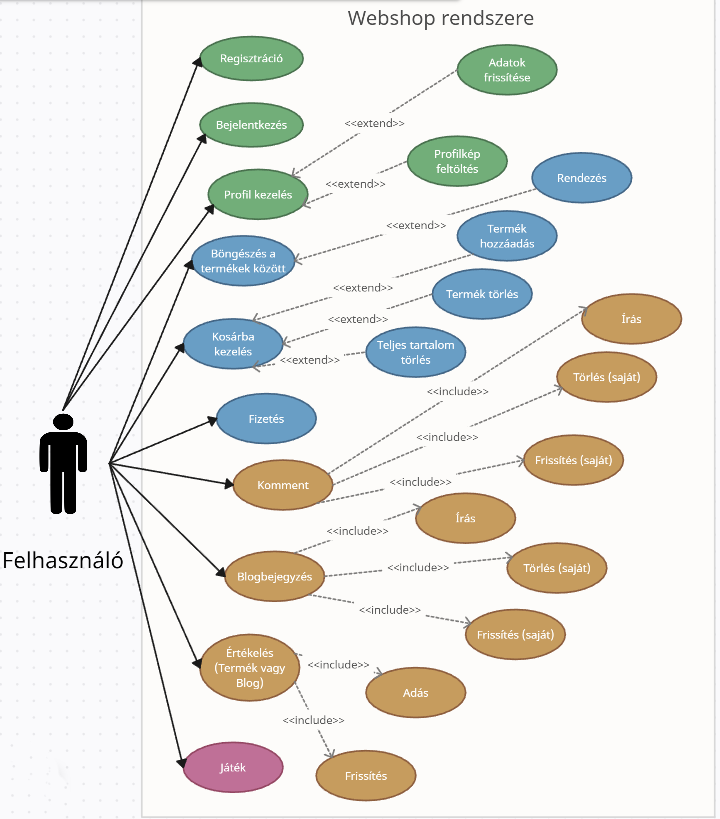
A blogbejegyzés funkció egyedülálló lehetőséget kínál a webshop regisztrált felhasználóinak, hogy ne csupán vásároljanak, de személyes tapasztalataikat, véleményüket és szakértelmüket is megosszák a közösséggel. Ez az interaktív és dinamikus funkció nem csupán kiegészíti a webshop kínálatát, de egy teljesen új dimenziót ad az online vásárlási élménynek.

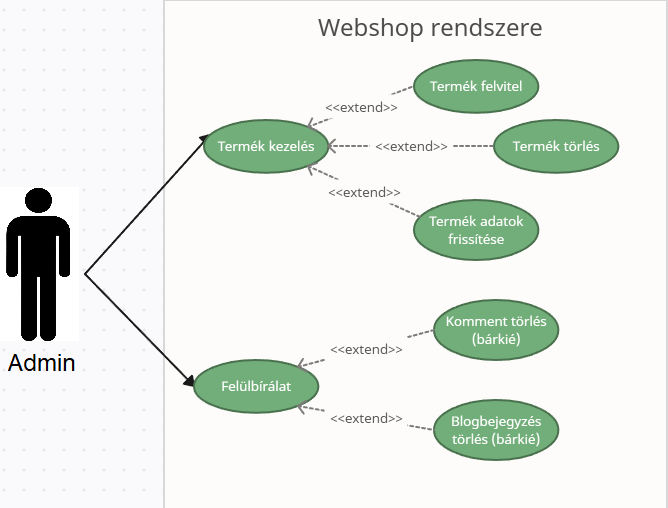
Miután a felhasználók regisztráltak, egyedi blogfelületet kapnak, ahol szabadon kifejezhetik gondolataikat és tapasztalataikat, mindemellett mások által írt blogbejegyzések sokasága között böngészhetnek. Ahogy az a termékeknél is fellelhető funkció, itt is lehetőség nyílik a mások által írt bejegyzések értékelésére szintén egy 1-től 5-ig terjedő skálán, melyet csillagok jelölnek. A blogbejegyzések írása és megosztása során a felhasználók szabadságot élveznek ugyanis értékelhetik a termékeket, megoszthatják azokkal kapcsolatos történeteiket, vagy éppen szakértői tanácsokat adhatnak a különböző italokról. A rugalmasság az egyik kulcsfontosságú jellemzője ennek a funkciónak. A felhasználók bármikor szerkeszthetik vagy frissíthetik a blogbejegyzéseiket, így mindig aktuális és releváns információkat oszthatnak meg a közösséggel. Emellett a bejegyzések akár el is távolíthatóak, így a felhasználók teljes kontrollt gyakorolhatnak a megosztott tartalom felett. A blogbejegyzés lehetősége nem csupán egy szokványos véleménynyilvánítási platform, hanem egy interaktív tudásmegosztási eszköz. Ezáltal a webshop nem csupán egy hely a termékeinek értékesítésére, hanem egy közösségi tér, ahol az ügyfelek egymás közötti kapcsolatokat építhetnek, és érdekesebbnél érdekesebb információkkal gazdagíthatják az italkultúrát és a webshop hosszú távú vonzerejét. A képen részlet látható a bejegyzések oldalról.

# **4.10. Admin funkciók**

A webshop adminisztrátorai különleges jogosultságokkal rendelkeznek, amelyek elengedhetetlenek a rendszer hatékony üzemeltetéséhez. Az adminisztrátorok számára elérhető kulcsfontosságú funkciók közé tartozik a termékek kezelése. Csakis az adminisztrátorok jogosultak új termékek hozzáadására, meglévők törlésére és azok adatainak frissítésére, biztosítva ezzel a webshop aktuális kínálatát. Ezen kívül az adminisztrátorok felelősek a kommentek és blogbejegyzések felülvizsgálatáért is. Jogosultságuk révén ellenőrizhetik a felhasználók által beküldött tartalmakat, és eltávolíthatják azokat, amelyek nem felelnek meg a webshop irányelveinek, vagy amelyek nem kívánatosak.

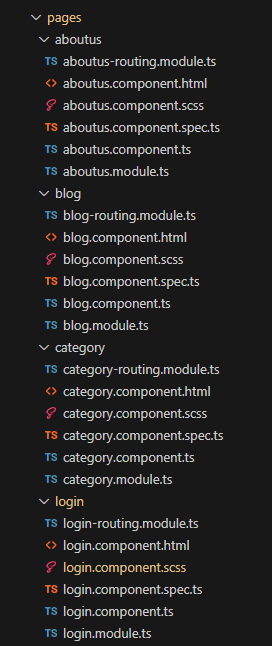
# **4.11. Use-Case diagramok**

Felhasználók használati eset kapcsolata a webshop rendszerével

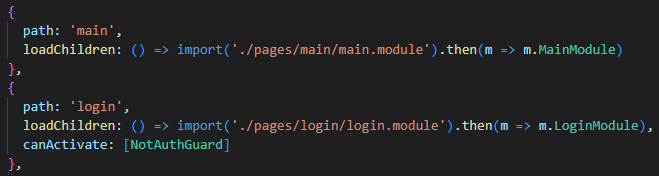
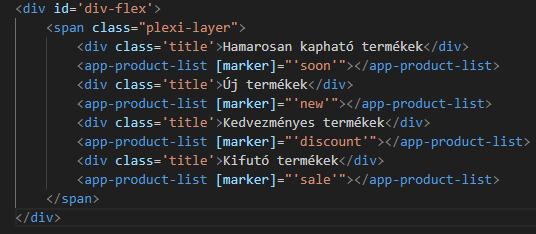
Admin felhasználók lehetséges további használati eset tevékenységeinek feltüntetése az alapvető felhasználói funkciókon túlmenően.

# **A megvalósítás lépései fontosabb kódrészletek kiemelésével**

# **5.1. Komponensek és modulok**

A komponensek és modulok hierarchikus szervezése kulcsfontosságú lépés az Angular alkalmazás megtervezésében. A komponensek tisztázottan elkülönülő egységeket képviselnek, amelyek lehetővé teszik az újra felhasználhatóságot és az egyszerűbb karbantarthatóságot. Minden komponens tartalmaz egy TypeScript fájlt a logika kezelésére, valamint HTML és SCSS (vagy más stílus) fájlokat a megjelenésért és stílusért. Modulokba szervezve ezek a komponensek még nagyobb egységeket alkotnak, ahol a modulok lehetnek általában önálló oldalakat jelenítenek meg. A modulok további tartalmakat is magukba foglalhatnak, mint például szolgáltatások, csővezetékek és más fájlok. A routing szintén elengedhetetlen részegység (egy különálló ts kiterjesztésű fájlban helyezkedik el), amely lehetővé teszi a modulok közötti hatékony átirányítást. Így könnyedén létrehozható egy Single Page applikáció. Az alkalmazás a legeneráláskor létre is hozza a gyökérmodult AppModule néven. Az ábrán néhány modul a látható a saját alkalmazásomból. Célszerű volt először megtervezni, hogy mennyi, és milyen céllal elkészülő modult hozzak létre. A tervezési folyamatban kiemelt a külső forrásból származó modulok importálásának fontossága az alkalmazás megjelenésének és működésének optimalizálása érdekében.

A teljes alkalmazás a gyökérmodult is beleértve 14 modulra bomlik szét, melyek mind különféle feladatokat hivatottak ellátni. A gyökérmodul app névre hallgat, és a többi modul felett áll, a fő feladatai közé tartozik a testvér modulok közötti navigációt megvalósítása, és a menüsáv kialakítása. A menüsáv, amennyiben telefonon, avagy kisebb méretű kijelzőn látogatjuk az alkalmazást egy hamburger menüvé alakul át. Ehhez egy külön komponens kapcsolódik a tagolhatósága végett, melyet az AppModule definiál. A képen a routing megvalósításának részlete látható:

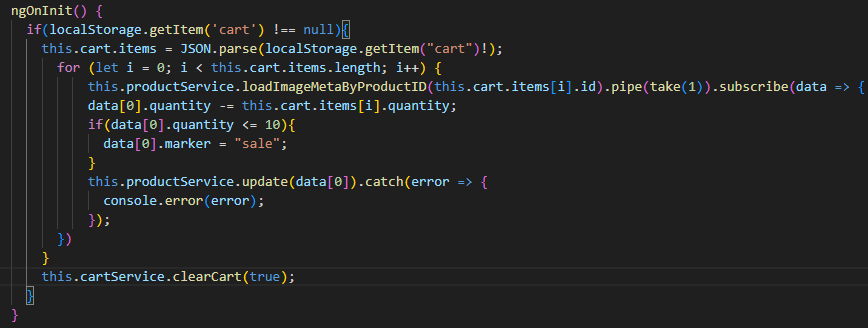
A MainModule (avagy főoldal) az áttekinthető felhasználói élmény érdekében különböző jelzésekkel ellátott termékeket kínál. A termékek jól láthatóan különböző címkékkel vannak ellátva, mint például "hamarosan érkező", "új", "akciós" vagy "kifutó". Az egyes termékekhez való hozzáférés és a kosárba helyezés lehetősége azonnal a felhasználók rendelkezésére áll a MainModule-ban, így felgyorsítva és megkönnyítve a vásárlás. A MainModule komponensekre van tagolva, ezzel optimalizálva a fejlesztést és elősegítve a karbantartást. A komponensek egyes felelősségek egy-egy sorért felelősek, ugyanis a főoldalon lényegében 4 sornyi terméket láthatunk a fent említett címkéknek megfelelően. Ez a komponensalapú megközelítés lehetővé teszi, hogy a kód tisztább, könnyebben érthető legyen, és lehetővé teszi a fejlesztők számára a funkcionalitás egyszerű módosítását vagy bővítését. Az első képen látható a MainModule csekély tartalma, melyben a komponenseknek átadott input direktíva segítségével osztjuk részekre az oldalt. A következő apró kép pedig ábrázolja, hogy a komponensekben miként kapjuk meg az input direktívában elhelyezett szöveges értéket:

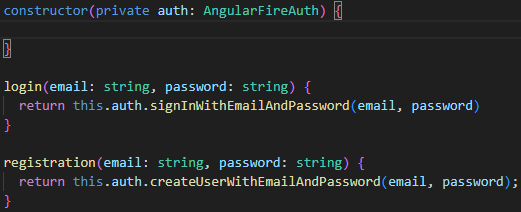


A CategoryModule szintúgy egy kifinomult megoldás, ahol az egyes termékek kategóriák szerint vannak csoportosítva. Mindössze egyetlen modulból áll, és dinamikusan még is 10 oldalnak ad otthont, mely a kategóriák számával egyezik meg, beleértve azt is amikor minden terméket megjelenítünk az oldalon. Ezen dinamikusság szintén a routingnak köszönhető, mert az url-ben azon felül, hogy megjelenik, hogy a „category” oldalon vagyunk, további részegységként fellelhető maga a kategória megnevezése, amely a termékek egy attribútumaként van eltárolva minden egyes árucikk esetén. Ezáltal a felhasználók könnyen navigálhatnak a különféle kategóriák között, és a CategoryModule segítségével gyorsan és hatékonyan megtalálhatják az érdeklődésüknek megfelelő termékeket. Ezzel a megközelítéssel a webshop rugalmasan alkalmazkodik a különböző kategóriákhoz, miközben fenntartja a könnyű kezelhetőséget és az intuitív felhasználói élményt. A felhasználónak ezeken az oldalakon is lehetősége nyílik az egyes termékekhez való hozzáférésre és az azonnali kosárba helyezésre. A böngészést nagyban megkönnyíti, hogy ár- és abc szerint lehetőségünk van rendezni az árucikkeket.

Szorosan kapcsolódik a fent említett két modulhoz a ProductModule, amely minden termékhez önálló oldalt létrehozva biztosítsa az árucikkek részleteiben való elmélyedést. A felhasználóknak lehetősége nyílik nagyobb ábrán megtekinteni az egyes termékeket, és azokhoz tartozó információkat is olvashatnak. Ezeken az oldalakon biztosított a termékek értékelése, emellett a kommentek hozzáfűzése. A gördülékeny böngészés érdekében a termékhez hasonló más termékek is fellelhetőek a mat-card-ban elhelyezett termékinformációk mellett. A „product” oldalak a kategóriához hasonló dinamikusságot hordozzák magukban.

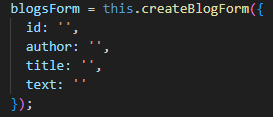
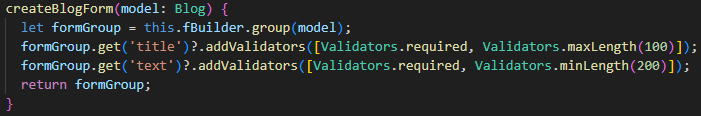
A kosár egy szintén önálló, kényelmes és felhasználóbarát modul, ahol könnyedén és átláthatóan kezelhetjük a virtuális bevásárlókosarunk tartalmát. Ez a modul lehetővé teszi, hogy a felhasználók a már kosárba helyezett tetszőleges számú terméküket egyben láthassák, átvizsgálhassák, és módosíthassák a megvásárolni kívánt darabszámot amennyiben meggondolnák magukat ezt illetően. Az oldalon belül lehetőség van a kosár tartalmának teljes törlésére, valamint az egyes termékek kosárból való eltávolítására is. A kosár oldalon minden egyes termékhez tartozó egységárak és a teljes ár is jól látható, így a vásárlók mindig tisztában lehetnek a költségeikkel. Az oldal intuitív és könnyen áttekinthető, lehetőséget biztosítva a felhasználóknak a vásárlás további folytatására vagy a fizetési folyamat elindítására. A kosár tartalmát egy erre a célra kifejlesztett modell segíti, melyet a későbbiekben fogunk részletesen megnézni.

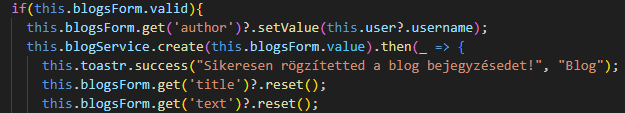
A fizetéshez kapcsolódik két kisebb modul, melyek a fizetés sikerességét jelölik, és a routingnak köszönhetően csak a kosár oldalon látható „Tovább a fizetéshez” gomb lenyomása után érhetőek el. Az egyik akkor kerül a felhasználók elé, amennyiben a fizetési kísérletük sikertelen. Ekkor semmi különösebb nem történik. A felhasználók tovább böngészhetnek a weboldalon, bővíthetik kosarukat, újabb fizetési akciót indíthatnak, és így tovább. A másik modul viszont kritikus fontossággal bír. Ugyanis, amennyiben sikeresen a vásárlás, akkor azt szintén jeleznünk kell a felhasználó felé. Ami még fontosabb, hogy ki kell ürítenünk a kosarat, és a raktáron levő árucikkek számát is módosítanunk kell a kiszállításra küldött mennyiség levonásával. Mivel a kosár tartalma a localStorage-ban van, ezért ennek a tartalmát kell törölnünk, emellett kritikus, hogy ténylegesen sehogy ne lehessen elérhető ez az oldal fizetés nélküli interakció esetén. A lent látható kód (ábra) értelmezése: Az ngOnInit() életciklusbeli függvény azonnal lefut, amikor az oldal megnyílik. A biztonság kedvéért először leellenőrizzük, hogy a localStorage-ban elhelyezett „cart” item értéke létezik-e. Amennyiben igen, akkor a JSON.parse() függvény segítségével kinyerjük az értékét a cart objektumunk items attribútumába, melynek típusa Array<CartItem>, amely azt jelenti, hogy a kosárba helyezett termékek tömbjét tartalmazhatja. Ezt követően végig iterálunk ezen a tömbön, és mivel a „CartItem”-ek nem egyenlőek az adatbázisban letárolt „Product”-okkal ezért a productService egy függvénye segítségével feliratkozunk arra, hogy lekérjük az adatbázisban található megegyező azonosítóval szereplő termékeket. A .pipe(take(1)) rész segítségünkre van abban, hogy mindig csak egyszer fusson le a lekérés. Alapvetőleg ez egy folyam, melyben a data lokális változó a termékek egy tömbjeként érhető el. Viszont mivel az ID, avagy azonosító egyedi attribútum ezért a data tömb egy egy elemű tömb lesz. Ez már az adatbázisban is letárolt terméket fogja tartalmazni, melynek módosíthatjuk a megfelelő értékeit. Ezt követően pedig a productService egy másik függvényével frissítjük az adatbázisban a megfelelő termékeket egyesével. A for ciklus lefutását követően pedig a teljes kosarat kiürítjük egy másik szolgáltatás használatával, melyben egyébként a localStorage „cart” névre hallgató elemét töröljük.

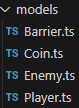
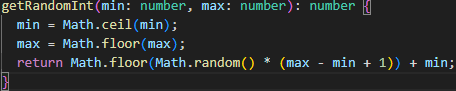
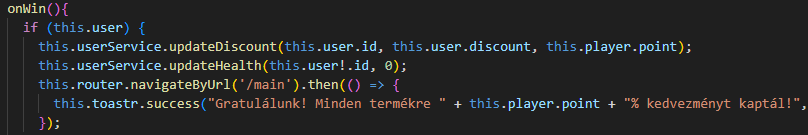
A regisztrációt és a bejelentkezést szintén önálló modulok alkotják. Mindkettő egyszerű, letisztult, könnyen kezelhető felület. A Firebase biztosította authentikáció van megvalósítva a weboldalon. Mindemellett a Firebase Database kollekciói közé tároljuk le a felhasználók regisztráció során megadott adatait. Egy különálló szolgáltatás használja az AngularFireAuth által biztosított függvényeket. Ezek teszik lehetővé, hogy az új felhasználó e-mail címe, és jelszójának hashelt formája mentésre kerüljön az Authentication fülön, avagy a már mentett értékek alapján azonosítsa a felhasználót. Ennek a szolgáltatásnak a használata mindkét modulban fellelhető a megfelelő függvényeinek meghívásával.

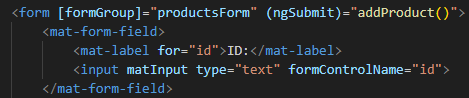
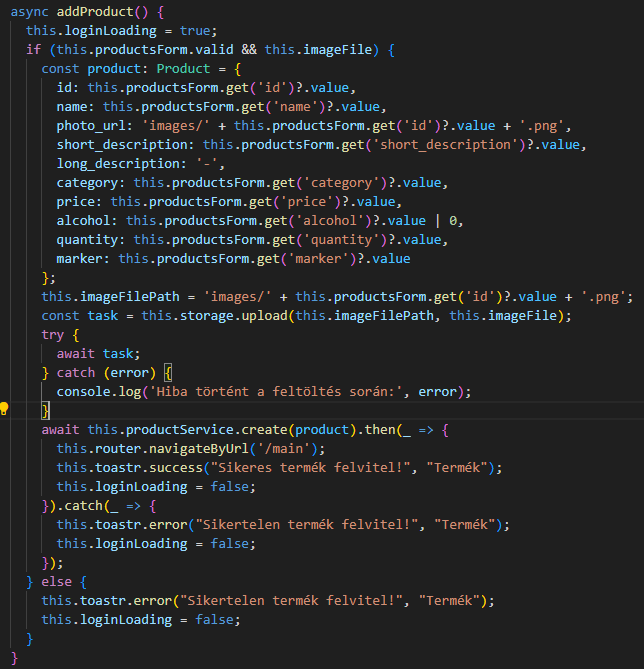
Van egy önálló modul, mely kisebb kiegészítésként funkcionál, ahol az ügyfeleknek lehetősége van közelebbről megismerni a webshop milyenségét. A menüsávban egyszerűen „Rólunk” néven jelenik meg, és egy kattintással azonnal elérhetővé válik. Ezen a felületen rövid összefoglalót és bemutatkozást találhatnak a látogatók, amely hozzájárul az üzleti történetünk és értékeink jobb megértéséhez. Az oldal alján egy link található, amelyre kattintva a regisztrált felhasználók 5%-os kedvezményre tehetnek szert. Fontos megérteni, hogy ez a kedvezmény a termékek 100%-os árára vonatkozik, függetlenül az esetlegesen már meglévő más kedvezményektől. Tehát, ha egy termék alapból 50%-os kedvezménnyel szerepel, a 5% kedvezmény ezen termék teljes árából kerül levonásra. Ezáltal biztosítjuk, hogy az ügyfeleink minden esetben részesülhessenek az extra előnyökből, és ezáltal növeljük a weboldalunk iránti elégedettségüket.

Az eddig említett modulok megtekintései nem igényelnek regisztrációt. Persze már ezek között is fellelhető olyan funkcionalitás, amelyhez szükséges a bejelentkezett fiók, mint például a kedvezményt szerző link használata. A következő modulok azonosított felhasználókat feltételeznek, ugyanis a guard-oknak hála nem érhetőek el bejelentkezés nélkül. A guard-ok speciális szolgáltatások, melyekben többféle függvényt írhatunk, meg és ezután ezeket szabadon hozzárendelhetjük a routinghoz. Ezen példában a guard canActive nevű függvényében szerepel a logika, amely bejelentkezetlen felhasználókat elnavigál a loginhoz. Elsőként vegyük figyelembe a BlogModule-t. Ez a modul lehetővé teszi a felhasználóknak, hogy blogbejegyzéseket írjanak, frissítsenek, és töröljenek. Természetesen mindenki csak a saját bejegyzése fölött bír efféle jogokkal. Az olvasás művelete viszont mások által írt bejegyzésekre is kiterjed. Emellett minden regisztrált felhasználó azonnal rendelkezik a fent említett jogok mindegyikével a belépést követően. Az admin felhasználók törlési jogot kapnak minden egyes blogbejegyzésre, a mások által készítettekre is, ezt nevezzük felülbírálási funkciónak. A blogok mindegyike egy mat-card-on helyezkedik el egy-egy külön komponensben. A modul html kódjában egy for ciklus által kerülnek egyesével átadásra a blogok adatai, mint az a képen is látható.

Hasonlóan a MainModule-hoz, itt is input direktívát használunk. A blogbejegyzés hozzáadásához a HTML-ben egy form használatával valósul meg. A TypeScript kód adja a logikáját, melyben egy belső függvény segítségével validátorokat rendelünk az egyes attribútumokhoz, ezzel szabályozva, hogy a felvitt blogbejegyzés megfeleljen az általam meghatározott kritériumoknak (tehát, hogy a blog címe maximum 100 karakter hosszú lehet, a bejegyzés tartalma pedig legalább 200 karakter hosszú kell, hogy legyen). 

Ezek után amennyiben a felhasználó rákattint a létrehozás gombra szintén egy függvény hívódik meg, amely elvégzi az adatbázisba tárolást egy szolgáltatás segítségével, üressé teszi újra a mezőket, és egy toastr segítségével jelzi a felhasználó felé egy felugró ablakban a felvitel sikerességét. A validátorok használata igencsak egyszerű, ugyanis az ellenőrzés a form-ot ellenőrizve, mintha a „valid” szó egy adattag lenne, vizsgálhatjuk, hogy megfelelt-e a megadott kritériumoknak. Ez egy logikai igaz avagy hamis értéket ad vissza.

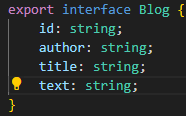
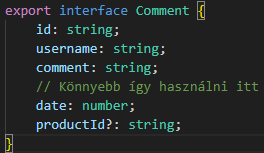
 A beépített játék is egy külön modulban helyezkedik el. A játékhoz tartozik egy önálló modellek mappa, amiben az összes felhasznált elem, és azoknak a funkcionalitása szerepel a könnyebb átláthatóság végett. Ezen objektumok vannak létrehozva a játék moduljában olyan számban amennyire szükség van. A játék teret egy canvas biztosítja, amely szélessége és magassága a kódban tetszőleges módosítható, azonban a játékos számára fixálva van. Írnom kellett egy random számokat generáló függvényt, amely elősegíti, hogy a pályán levő ellenségek, az akadályok, és a megszerezhető pénzérmék helyzete is véletlenszerű legyen. Ezenfelül a játék saját TypeScript fájljában azt határozza meg, hogy mennyi legyen az ellenségek, és az akadályok darabszáma. A belépett felhasználó adatai között szerepel a kedvezmény százaléka, amely érték 3-mal nő amennyiben a játékos sikeresen felvette az összes pénzérmét, és ezt követően rákattintott a jutalom megszerzését jelző gombra. Végül visszanavigáljuk a fogyasztót a főoldalra.

 Az utolsó modul kizárólag admin felhasználók számára elérhető. Ez az a felület, ahol gördülékenyen tudnak új terméket felvinni, mely azonnal feltöltődik az adatbázisban a megadott adatok alapján, és ezt követően meg is jelenik a megfelelő oldalakon. A hozzáadás függvénye magában rejti az új termék adatainak felvitelét a Firestore Database megfelelő kollekciójába és a Storage-ba is megfelelő módon letárolja a képet. A fájl feltöltés kapcsán letároljuk a képet egy változóban, az elérési útvonalat pedig a megadott termékazonosító alapján állítjuk be.

# **5.2. Modellek (Interfészek)**

A használandó modellek átgondolása és kialakítása kulcsfontosságú lépés a webalkalmazás tervezésekor, mivel ezek határozzák meg az alkalmazás adatstruktúráit. A modelltervezés során figyelembe vettem az alkalmazás üzleti logikáját és a kívánt funkciókat, hogy az adatkezelés hatékony és átlátható legyen.

Elsőként a „User” modellt terveztem meg, mely a megvalósítás során átesett kisebb módosításokon is. Át kellett gondolni, hogy melyek azok a releváns információk, amelyeket mindenképp el kell tárolni egy felhasználóról. Ez magában foglalja a létrehozott fiók felhasználónevét, e-mail címét, keresztnevét, vezetéknevét, jelszavát, jogosultságainak kezelését egy admin attribútumban, a profilképének url-jét. A felhasználómodellt úgy kellett megalkotni, hogy támogassa a regisztrációt, a bejelentkezést és az esetleges profilfrissítéseket. A korábban említett módosítások a következők (új attribútumok): a fiókhoz tartozó játszással felhasználható életerő pontok mennyisége, az összesen megszerzett kedvezmény százaléka, a kedvezményes link használatba vétele egy logikai attribútumban.

A „Comment” és „Blog” modellek kapcsán érdemes volt megtervezni azokat az attribútumokat, amelyek segítségével egyértelműen azonosíthatók ezek az entitások. Például egy komment esetében a szerző felhasználóneve, a komment szövege, a dátum és annak a terméknek az azonosítója, amelyhez vonatkozóan megírták. A blogmodell esetében a cím, a tartalom és a szerző felhasználónevének letárolása tűnt releváns információnak. 

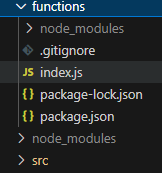
A „Rating” modellje magában foglalja az értékelt termék azonosítóját, a felhasználó által adott értékelést, mint szám érték, és magának az értékelőnek a felhasználónevét. A későbbiekben ugyanezt a modellt használtam fel, amikor a blogbejegyzések értékelését valósítottam meg. Itt a termékazonosító helyett, a bejegyzés azonosítója kerül átadásra, a többi viszont teljesen hasonlóan zajlik, mint a termékek értékelése.

Végül, a „Product” modellje a legkritikusabb fontosságú a webshop szempontjából. Ebben a modellben eltároltam a termék nevét, rövid leírását (eleinte terveztem egy hosszabb leírást is adni minden termékhez, azonban végül a főoldalt más kialakításban hoztam létre, így csak egyetlen leírás maradt benne a végleges verzióban), árát, raktári készletét, a termék képének url-jét, és egyéb releváns információkat (alkoholtartalom, kategória, valamilyen jelzés). A kategóriák attribútum nagyban segíti a termékek hatékony csoportosítását és keresését.

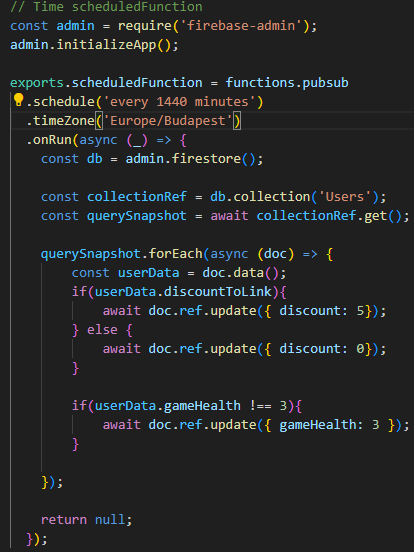
Eredetileg nem tervezett modell, de kiemelt fontossággal bír, és rendkívül megkönnyítette a munkát a „CartItem” modellje, mely a „Cart” modell egyetlen attribútumaként egy tömböt is alkot. Ebben a modellben tárolásra kerülnek a kosárba helyezett árucikkek megfelelő attribútumai. Ez az egyetlen olyan modell, mely csak helyileg van felhasználva a frontend oldalon, ugyanis csak az átláthatóságot segíti. A „Product” modell megfelelő attribútumait használja fel, amellett, hogy tartalmazza a kosárban elhelyezett mennyiséget is, mint adattag.

A modellek létrehozása során fontos, hogy azok összhangban legyenek az Angular alkalmazás tervezési mintájával és az alkalmazás üzleti logikájával. A megfelelő modellek segítenek a hatékony adatkezelésben és hozzájárulnak az alkalmazás átláthatóságához és karbantarthatóságához. A backend-en, melyet a Firebase biztosít ezen modellek mindegyike a „Cart” kivételével, ugyanis az csak helyi felhasználásban releváns egy kollekcióban szerepel, és így tárolódik le a Firebase Database funkciójának segítségével. Fontos, hogy a modellek összhangban legyen a tényleges adatbázisban létrehozott kollekciókkal.

# **5.3. Backend kódok**

Habár alapvetőleg ez egy frontend fejlesztés, mely már egy meglévő backend-et használ ki a Firebase segítségével, még is vannak olyan részek melyekhez muszáj volt egyedi kódsorok írására. A Firebase Cloud Functions nagyban hozzájárult ehhez a művelethez. A teljes frontend alkalmazás lényegi kódja az src mappán belül helyezkedik el. Ehhez a művelethez viszont ezen kívül kellett tekinteni, egy külső functions mappába tárolt index.js fájl biztosítja a backend-hez írt újdonsült funkciókat. Ezen mappa jól láthatóan egyedi node\_modules mappával és package.json fájlokkal is rendelkezik a konfiguráció érdekében.

Az első különálló függvény lényegi kódjának magyarázata: Elsőként importáltam magát a Firebase Functions modult, amely lehetővé teszi a Firebase Cloud Functions használatát. Ezt követően importáltam az Express modult, amely egy webes alkalmazáskeretrendszer. Általa tudunk egy külön álló Express szervert működtetni, ami kiszolgálja a fizetési rendszert. A CORS (Cross-Origin Resource Sharing) importálása segít a keresztezett eredetű http kérésék kezelésében. Ez azt jelenti, hogyha a frontend és a backend nem ugyanazokon a domain-eken van telepítve, de még is tudnak kommunikálni egymással (Ez nagy segítség volt a lokális tesztelésnél is, ahol különböző port-on kellett futtatni az express szervert). A Body-parser a http kérések és válaszok testének tartalmainak könnyű kezelése végett került importálásra. Az importok a require kulcsszót használják. Mindezek után következik az express alkalmazás inicializálása az app nevű változóban. Ezután ezen állítottam be néhány dolgok, hogy gördülékenyebben kezeljen bizonyos típusú adatokat, használja a CORS-t. A következő változó stripe névre hallgat, ebbe a stripe rendszer titkos kulcsának értékét tároltam le, melyet biztonsági okokból olvashatatlanná tettem. Ezt követi a meghívott post metódus, amelyben létrehozzuk, hogy hogyan nézzen ki, és miket jelenítsen meg az az oldal, ahol a fizetést lebonyolíthatjuk a bankkártya adatok megadásával. Itt látható, hogy a sikeres és a sikertelen fizetési esetén bekövetkező útvonal navigáció is. Végül az utolsó sorban látszik, hogy a függvény api néven fog megjelenni a felhőszolgáltatás függvényei között. 

 A másik függvény a játékhoz kapcsolódik. Itt szükséges a firebase admin felületének importálása és inicializálása, ugyanis nem frontend oldalon vagyunk most, tehát más úton van lehetőség lekérni, és módosítani az adatbázis egyes adatait. Jól látható, hogy a függvény a scheduledFunction névre hallgat. A nevéből adódóan látszik, hogy ez a függvény ütemezett, azaz bizonyos időközönként fut le. Még pedig 1440 percenként, azaz mindennap pontos egyszer, mindennap ugyanabban az időpontban. A teljes Users kollekciót lekérjük, végig iterálunk rajta, és módosítjuk a megfelelő értékeket.