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 is meg lesznek valósítva. 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. Emellett a megalkotott webalkalmazás meg fogja állni 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 az emberek életét. Emellett a tulajdonosok számára is előny, hiszen rengeteg felhasználóhoz el tudják juttatni termékeiket.

A hallgató feladatát képezi egy újabb webshop megalkotása, mely azt az egyediséget viseli, hogy egy kis játék alkalmazás legyen beépítve a webshop keretein belülre, aminek segítségével különböző akciókra 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. Ezen üzleti logika elősegítheti, hogy a felhasználók több mindent rendeljenek, de még is szimpatikussá tegyük számukra a rendszert.

# ***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 kedvezmények megszerzéséhez.

* **A megoldási mód:**

Első tekintetben meg kellett ismernem az Angular keretrendszert körültekintőbben. 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. A firebase cloud functions nagy segítségemre volt a fizetési rendszer beépítésekor, és a játék időhöz kötésekor. A projektet időnként a Git verziókezelő segítségével GitHubra feltöltve tárolom.

## Tartalomjegyzék

[*Feladatkiírás* 3](#_Toc138870516)

[*Tartalmi összefoglaló* 4](#_Toc138870517)

[Tartalomjegyzék 5](#_Toc138870518)

[1. Bevezetés 6](#_Toc138870519)

[1.1 Problémafelvetés, motiváció 6](#_Toc138870520)

[2. Alkalmazott eszközök, technológiák bemutatása 8](#_Toc138870521)

[2.1 Fejlesztői környezetek 8](#_Toc138870522)

[2.1.1 Visual Studio Code 8](#_Toc138870523)

[2.1.2 Böngésző fejlesztői mód 8](#_Toc138870524)

[2.2 Programozási nyelvek 9](#_Toc138870525)

[2.2.1 TypeScript 9](#_Toc138870526)

[2.2.2 HTML 10](#_Toc138870527)

[2.2.3 SCSS 10](#_Toc138870528)

[2.3 Keretrendszerek 10](#_Toc138870529)

[2.3.1 Angular 10](#_Toc138870530)

[2.4 Firebase 11](#_Toc138870531)

[3. Funkcionális specifikáció 12](#_Toc138870532)

[3.1. Bejelentkezés és regisztráció 12](#_Toc138870533)

[3.2. Böngészés a termékek között 13](#_Toc138870534)

[3.3. Termékek részletei 13](#_Toc138870535)

[3.4. Felhasználói visszajelzések 13](#_Toc138870536)

[3.5. Tervezett további funkciók 14](#_Toc138870537)

[3.5.1. Kosárkezelés 14](#_Toc138870538)

[3.5.2. Integrált fizetési rendszer 14](#_Toc138870539)

[3.5.2. Beépített játék 14](#_Toc138870540)

[3.5.3. Felhasználói fiók kezelése 15](#_Toc138870541)

[4. A megvalósítás lépései 16](#_Toc138870542)

[4.1. Komponensek és modulok 16](#_Toc138870543)

# **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 Super Mario-hoz hasonló 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. 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.

# **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**

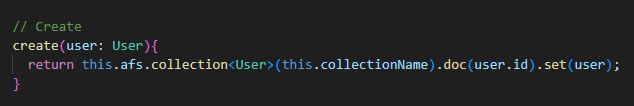
Az általam is használt nyílt forráskódú webes keretrendszer az Angular, melyet a Google fejlesztett ki, és adott ki 2010. október 20-án még Angular JS néven modern webalkalmazások fejlesztésének céljából. A célja, hogy felhasználóbarát, jól skálázható, hatékony alkalmazást fejlesszünk általa. Eleinte 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.

# **Firebase**

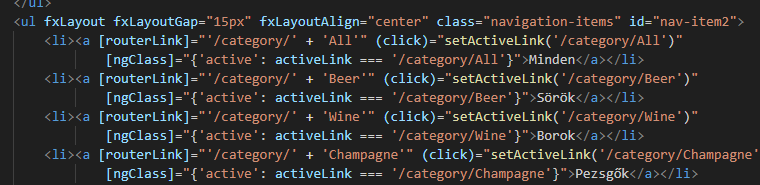
A Firebase a Google által kifejlesztett népszerű és teljeskörű fejlesztői platform. A gyors, hatékony fejlesztésben, tesztelésben, sőt az üzemeltetésben is nagy segítségére van a fejlesztőknek. Az eszközei és szolgáltatásai rengeteg segítséget nyújtanak. Idetartozik a teljesség igénye nélkül a felhasználókezelés 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.

# **Funkcionális specifikáció**

# **3.1. 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 nem tudjuk visszafejteni, szinte lehetetlen, így az alkalmazásunk biztonságosabb lesz az adathalászokkal szemben. A Firebase egy egyedi azonosítót (User UID) is rendel 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. 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-ben is megjelenik az újonnan regisztrált felhasználó azon felül, hogy az Authentication fülön is megtalálható lesz.

# **3.2. 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 későbbiekben majd kereső funkció is elérhető lesz az oldalon, mely segítségével akár egyesével is kiszűrhető lesz minden egyes termék. További rendezési tervek: ár és név alapján való rendezés.

# **3.3. 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 rendeljenek. 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. A kosárba tétel mind a részletes áttekintésnél elérhető, mind a nagyobb kiterjedésű böngészés közben.

# **3.4. 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 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 írta. Az ilyesfajta értékelések azért jók, mert az üzemeltető láthatja, hogy mi az a termék, amire nagyobb az igény, mi az, amire nem azon felül is, hogy nyilvántartja a szállítási mennyiségeket. Emellett a vásárlók a pozitív visszajelzéseikkel egymást is vásárlásra ösztönözhetik. Másik lehetőség, hogy a felhasználók egy pontozási rendszerben tudják értékelni a nekik tetsző termékeket. A skála 1-től 5-ig terjed. 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.

# **3.5. Tervezett további funkciók**

Még számos kiaknázatlan lehetőség rejlik a webáruház megalkotásában, főleg a tervezett beépített játék adja majd sokszínűségét és egyediségét. Ezek közül említek néhányat a teljesség igénye nélkül.

# **3.5.1. Kosárkezelés**

Jelenleg a felhasználó kap egy visszajelzést egy toastr segítségével arról, hogy hány darab és milyen termék került a kosarába, de valójába még a háttérben nem működik semmi. A kosárnak egy önálló modulja lesz, ami mindig az adott felhasználóra vonatkozóan tárolja, hogy miből mennyit helyezett el a megvásárolni kívánt termékek listájára. Ezen a külön oldalon majd vissza lehet vonni, ha még sem szeretne megvásárolni egy terméket, emellett a vásárolni kívánt mennyiség is korrigálható lesz.

# **3.5.2. Integrált fizetési rendszer**

A webáruházban kizárólag valamilyen bankkártyás fizetés lesz elérhető. A regisztrált felhasználóknak lehetőségük lesz eltárolni bankkártya adataikat így meggyorsítva a rendelésüket. Ezzel szemben a regisztrálatlan felhasználóknak minden egyes alkalommal meg kell adniuk az adataikat. A kosárnak fenntartott külön oldalon lesz majd lehetőség a fizetésre. Ekkor értelemszerűen szállítási hely, emellett egyéb elérhetőség is megadható lesz.

# **3.5.2. Beépített játék**

Egy külön oldalon lesz elérhető a beépített játék, mellyel a felhasználók kedvezményekhez juthatnak. Teljesen véletlenszerű lesz, hogy melyik termékre vagy termékekre kapnak akciót. Viszont az, hogy mennyit az csak is a teljesítményükön múlik. Mindennap egy felhasználó csak egyszer indíthatja el a játékot, és az akció mindösszesen 24 óráig lesz majd elérhető a számára. A regisztrálatlan felhasználóknak csak a játék demó verziójával lesz lehetőségük játszani egy kb. 30-60 másodperces időlimittel. Így kedvezményekhez nem juthatnak csak kipróbálhatják. Ez afféle reklámként szolgál, és arra buzdít, hogy regisztrálj a webáruházba.

# **3.5.3. Felhasználói fiók kezelése**

A felhasználók jelenleg csak hozzászólásokat tudnak írni, melyhez a felhasználónevük kerül azonosítás gyanánt (az email címük nem lesz publikusan látható). Viszont egy fiók a fent említett funkciókon kívül még módosítható és kezelhető is lesz. Ugyanis megtörténhet, hogy egy adott felhasználó szeretné megváltoztatni a jelszavát, a felhasználónevét, vagy esetleg új bankkártyát akar hozzáadni, a régit törölni stb. Ehhez egy külön felület lesz kialakítva az adott felhasználóra vonatkozó adatokat szerepeltetve az oldalon.

# **A megvalósítás lépései**

# **4.1. Komponensek és modulok**

Egy Angular alkalmazás esetében elengedhetetlen, hogy komponensekbe szervezzük. Ez átláthatóbb kódot eredményez, és ezáltal külön egységekbe tudjuk tárolni az összetartozó részeket. A komponensek tehát külön egységeket jelölnek, melyek lehetővé teszik az újra felhasználhatóságot. Ezek adják a felhasználói felület (UI) építőköveit. Minden komponenshez tartozik logikáért felelős TypeScript fájl és a megjelenésért és stílusért felelős HTML és SCSS fájlok. A komponensek tehát felelősek a felhasználói felület megjelenítéséért és a felhasználói interakciók kezeléséért. A komponenseket modulokba szervezhetjük, amik egy még nagyobb egységet alkotnak. Egy modul több komponensből is felépülhet, emellett szolgáltatásokat, csővezetékeket és egyéb fájlokat foglalhat magába. Az alkalmazás a legeneráláskor létre is hozza a gyökérmodult AppModule néven. Az ábrán látható néhány modul a saját alkalmazásomból.