ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

# SZAKDOLGOZAT

**OE-NIK** Hallgató neve: **Gellért Péter**
**2022** Hallgató törzskönyv száma: **T/005995/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

# SZAKDOLGOZAT FELADATLAP

| | |
|---|---|
| Hallgató neve: | **Gellért Péter** |
| Törzskönyvi száma: | T/005995/FI12904/N |
| Neptun kódja: | [redacted] |

A dolgozat címe:

**Társasjáték kölcsönző weboldal megvalósítása**
**Development of a boardgame renting webapplication**

| | |
|---|---|
| Intézményi konzulens: | Simon-Nagy Gabriella |
| Külső konzulens: | |
| Beadási határidő: | 2022. május 15. |
| A záróvizsga tárgyai: | Számítógép architektúrák<br>Big Data és üzleti intelligencia specializáció |

**A feladat**

Tervezzen és valósítson meg egy társasjáték-kölcsönző webalkalmazást, amely képes a felhasználó tevékenységéhez igazodni mind a felület elrendezésében, mind pedig az ajánlott tartalmakat tekintve. Kutassa fel és vizsgálja meg a feladat megoldásához használható technológiákat. Ismertesse a hasonló célú portálok működését és képességeit. Valósítson meg egy olyan tesztrendszert, amely a felhasználók viselkedését és preferenciáit figyelembe véve változtatja a megjelenített tartalmat, valamint készítsen egy webes adminisztrációs felületet is.

**A dolgozatnak tartalmaznia kell**:

- a feladat pontos ismertetését,
- a hasonló rendszerek működését és képességeit,
- a feladat megoldásához alkalmazható technológiák tömör bemutatását,
- a választott megoldások indoklását,
- a tervezés és megvalósítás részletes leírását,
- a tesztelés tervét és eredményeit,
- az elkészített rendszer eredményeinek részletes bemutatását és értékelését,
- a továbbfejlesztési lehetőségek számba vételét.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest

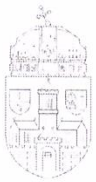
.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.14.

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# KONZULTÁCIÓS NAPLÓ

Hallgató neve: Gellért Péter

Neptun Kód: [redacted]

Tagozat: Mérnök Informatika

Telefon:

Levelezési cím (pl.: lakcím):

[redacted]

Szakdolgozat címe magyarul:

Társasjáték kölcsönző weboldal megvalósítása

Szakdolgozat címe angolul:

Development of a boardgame renting webapplication

Intézményi konzulens:

SIMON-NAGY GABRIELLA

Külső konzulens:

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2021. 09. 21. | A téma meghatározása | |
| 2. | 2021. 10. 19. | A kutatási területek áttekintése | |
| 3. | 2021. 11. 22. | Az irodalomkutatás összegzése | |
| 4. | 2021. 12. 10. | A rendszerspecifikáció véglegesítése | |

A hallgató a Szakdolgozat I. tantárgy követelményét teljesítette, beszámolóra bocsátható.

.....

Intézményi konzulens

Budapest, 2021. december 10.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# KONZULTÁCIÓS NAPLÓ

Hallgató neve: Gellért Péter

Neptun Kód: [redacted]

Tagozat: Mérnök Informatika

Telefon:

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat címe magyarul:

Társasjáték kölcsönző weboldal megvalósítása

Szakdolgozat címe angolul:

Development of a boardgame renting webapplication

Intézményi konzulens: SIMON-NAGY GABRIELLA

Külső konzulens:

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2022. 02. 23. | Megvalósítási lehetőségek áttekintése | |
| 2. | 2022. 03. 17. | Fejlesztési fázis ellenőrzése | |
| 3. | 2022. 04. 22. | Tesztelés ellenőrzése | |
| 4. | 2022. 05. 10. | Eredmények, összegzés | |

A hallgató a Szakdolgozat II. tantárgy követelményét teljesítette, védésre bocsátható.

.....

Intézményi konzulens

Budapest, 2022. május 10.

# TARTALOMJEGYZÉK

# 1. BEVEZETÉS

Magyarország társasjátékok iránti szükséglete ugrásszerűen emelkedett az elmúlt években. A társasjátékok remek családi vagy baráti kikapcsolódást biztosítanak ezenfelül a gyermekek fejlesztésében is nagyon hasznosak. A társasjátékok fejlesztik a kreativitást a szociális készségeket ezen felül megtanítanak minket veszíteni. Összességében a társasjátékokra hatalmas szükségünk van.[1]

A pozitív hatásokon kívül a jelenlegi pandémia helyzet is csak dobott a társasjátékok népszerűségén. Az emberek a bezártságot társasjátékokkal kompenzálták. A társasjáték iparág az elmúlt években évente 10%-os növekedést mutatott.[2]

A társasjátékok között ma már találunk olyan komplexitású stratégiai játékokat, amik akár több órán keresztül is lekötik az embert és csak a szabályok elmagyarázása és megértése is akár egy órát igénybe vehet. Ezek már nem gyermekeknek való 'ki nevet a végén' társasjátékok, hanem komoly gondolkozást, taktikázást igénylő játékok keresőképes felnőtteknek.

A pandémia miatt a rendelések száma is gyarapodott. Ha az ételek rendelését nézzük, akkor a COVID ideje alatt Észak Amerikában duplázódott a rendelések száma.[3] Egy ilyen növekedés egyértelműen azt mutatja, hogy az emberek szeretnek otthonról intézni dolgokat, a rendelési kedv gyarapodik.

Ezekből adódott az ötlet, hogy egy olyan kölcsönzőt hozzak létre, ahonnan az emberek egy átlagos ételrendelő vagy éppen autóbérlő oldal felületéhez hasonló weboldalon keresztül tudják megrendelni a kívánt társasjátékot. A kiválasztott társasjátékot szabadon választott időre ki lehetne bérelni, amit futárral vagy személyesen lehet el és vissza szállítani. A fizetés a weboldalon keresztül történne. A bérlés regisztrációhoz és kaucióhoz kötött, amit a társasjáték visszajuttatása és leellenőrzése után visszautalunk.

A weboldalon törekedni fogok olyan megoldások használatára, hogy a kölcsönzések számát maximalizálhassam. Ezt egyrészről a felhasználói felület optimális elrendezésével másrészről az ajánlások testreszabásával szeretném elérni. A fiatalos és modern weboldal designnal szeretném ösztönözni a vásárlókat az oldalam meglátogatásához.

Jelenlegi egyetlen piacon lévő megoldás a játszóház projekt, ami hasonlít az én ötletemhez. A különbség csak annyi, hogy míg a játszóház projekt véletlenszerű társasjátékokat küld ki a felhasználóknak és kéthetes ciklusokban cseréli azokat, az én weboldalamon pontosan ki lehet majd választani a kívánt játékot, illetve azt a kívánt ideig megtartani. [4] Lehetőség lesz arra, hogy akár már aznap elvigyék a megrendelt társasjátékot.

Célom egy olyan weboldal létrehozása, amiből későbbiekben akár egy vállalkozás nőhet ki.

# 2. IRODALOMKUTATÁS

## 2.1 Technológiák

Függetlenül attól, hogy microservice vagy monolitikus felépítésben készítjük el a programot a szervízek vagy front-end back-end közti kommunikációra szükségünk van. Ehhez rendelkezésünkre áll a REST szoftver architektúra és a SOAP protokoll. A SOAP egy nyelv és információátviteli protokoll független megoldás. Beépített hibakezeléssel rendelkezik. A SOAP nagyobb sávszélességet használ mivel a kérés sok adatot tartalmaz, így biztosítva a biztonságos kommunikációt. Formátum kezelésben a SOAP egyedül XML formátumot tud kezelni.[5] A REST-et használó alkalmazásokat RESTful alkalmazásoknak hívjuk.[6] Ennek előnye, hogy a kérések kisebb adatmennyisége miatt gyorsabb, mint a SOAP, illetve az XML-en kívül egyszerű szöveg, JSON és HTML formátumot is tud kezelni. Gyorsítótárral is rendelkezik, illetve a megtanulása és implementálása egyszerűbb. Amennyiben az alkalmazásunkat microservice architektúrában szeretnénk megvalósítani, használhatunk még KAFKA-t mint szervízek közötti kommunikációt. A KAFKA egy Apache által kifejlesztett eszköz, amivel nagy mennyiségű adatot tudunk elküldeni. A KAFKA egy magasan skálázható eszköz. Nagy előnye még, hogy rendelkezik egy tárolóval, amiben tárolja az üzenetet amennyiben a fogadó fél nem elérhető. Amint elérhetővé válik, elküldi a tárolóban lévő üzeneteket. Hátránya, hogy nincs azonnali válaszunk, így nem tudjuk, hogy az üzenetünk feldolgozásra került-e. Abban a helyzetben használhatjuk, hogyha ez nekünk nem fontos információ.[7]

Projektépítő eszközök közül Maven és Gradle között választhatok. Mindkettő eszköz megoldásban ugyan annyira lenne számomra hasznos azonban több különbség is van köztük. Az egyik legjelentősebb a performancia, amiben a Gradle toronymagasan nyer. Nagyobb projekteknél az alkalmazás fordítása akár 100-szorosan is gyorsabb lehet. [8] Ami a Maven mellett szól, hogy elterjedtebb, könnyebben tanulható és több dokumentáció és oktatóanyag áll a rendelkezésemre. Sok integrált fejlesztői környezet jobban ki tudja használni a Maven adta funkciókat mivel könnyebben használható.

Az alkalmazás fejlesztését nagyban megkönnyíti, ha virtuális gépek segítségével fejlesztek. Ennek célja egy olyan izolált környezet létrehozása, amit bárhol tudok futtatni. Jelenleg két megoldás lehet ideális számomra. Az első egy Virtuális gép létrehozása. Ennek hátránya, hogy a virtuális gép egy teljes számítógép emulációja, ami azzal jár, hogy a gazdaszámítógép memóriáját, processzorát stb. használja. Egy másik manapság elterjedt megoldás a Docker nevű program, ami a virtuális gépekhez képest egy eltérő megközelítést ad. A Docker lényege, hogy az általa létrehozott „minden container osztozik a gazda kerneljén a többi containerrel” [9], így a legnagyobb előnye, hogy nincs virtualizált hardver. A Docker egy gyorsan megtanulható és könnyen használható alkalmazás, ami sok erőforrást meg tud spórolni fejlesztés közben.

Amennyiben konténereket szeretnék használni, nem lehet kihagyni a Kubernetest, amit nagy méretű tárolók kezelésére használnak. A Kubernetes egy olyan platform, ami a konténerek kezelését segíti elő. Könnyebb lesz vele a skálázhatóság, figyeli a folyamatokat a változásokat.[11] Használatával kevesebb időt kellene fordítanom DevOps munkákra, azonban érzésem szerint az én rendszerem nem lesz akkora, hogy megérje egy ekkora erőforrás bevezetése.

Az applikációm HTTP kérésekkel fog kommunikálni a front-end és back-end között így szóba jöhet az API könnyű tesztelése. Erre az egyik legismertebb eszköz a Postman, ahol REST kéréseket tudunk elküldeni és figyelni a választ. Jó lehet performancia tesztre vagy akár API tesztre is, hogy megnézzem az egyes végpontok jó adattal térnek-e vissza.

Mivel az alkalmazásomat valószínűleg Java nyelven fogom fejleszteni ezért el kell döntenem, hogy milyen függőség befecskendező eszközt használjak. A számomra két elérhető opció a Spring Framework és a Google által kifejlesztett Guice. A Guice egy vékony függőségkezelő keretrendszer. Az előnyei közé tartozik, hogy könnyen és gyorsan fejleszthető, hátránya pedig, hogy az alkalmazásomban így különböző keretrendszereket kell keresnem. A Spring előnye, hogy egy nagyon régóta fejlesztett, hatalmas dokumentációval rendelkező keretrendszer. Amennyiben a Springet választanám, nem csak a függőség befecskendezését kapnám meg de a beépített adatbázis kezelőt is, ezen kívül a Spring Boot használatával a fejlesztési sebességen is jelentősen növelni tudnék.[12]

A fejlesztés elengedhetetlen része a verziókezelés. A verziókezelésnek rengeteg előnye van. Először is a visszaállításban nagy segítségünkre lehet hiszen egy commit alkalmával elmentjük az aktuális állapotot. Így amennyiben a programunkban olyan hibát vétünk, ami a visszaállításhoz vezet, a verziókezelő segítségével egy commit hash használatával ezt megtehetem. Ezzel jelentős munkaórát spórolhatok meg magamnak. Az egyes commitokhoz megjegyzéseket köthetünk így később visszanézhető, hogy egyes fájlokban miért történt változás. Ami jelenleg nekem nem lesz prioritás, de későbbiekben akár jól jöhet az a szinkronizáció. Amennyiben egynél több ember dolgozik egy projekten elengedhetetlen, eszköz a verziókezelés, hogy mindenki mindig a legfrissebb verziót kapja meg. [13] Verziókezelők közül a Git és az SVN a két számomra elérhető.

Fejlesztőkörnyezet választása sem elhanyagolható dolog. Egy jó fejlesztőkörnyezettel a fejlesztés sebessége jelentősen növelhető. Amennyiben notepad-et használnék nem kapnék olyan segítséget, amit egy jó fejlesztőkörnyezet tud nyújtani. A szóközöket nekem kellene beírni a függőségeket nekem kellene kézzel beimportálni, nem kapnék szintaxis jelzéseket, illetve az automata kiegészítés sem működne. Ez jelentős időt venne el a tényleges fejlesztés elől. A piacon manapság már nagyon sok fejlesztőkörnyezet elterjedt ezek közül nem könnyű a választás. A legjobban elterjedt fejlesztőkörnyezetek az IntelliJ, Eclipse, Visual Studio Code, Microsoft Visual Studio. A választás a programozói nyelv kiválasztásától is függ. [14]

A fejlesztés és telepítés megkönnyítése érdekében használhatok folyamatos integrációt és folyamatos szállítást segítő eszközöket. Ezzel a kód tesztelését tudom megkönnyíteni,

illetve, ha szeretném a kód feltolása után azonnal lefuttathatok rajta teszteket, és ha a tesztek sikeresen lefutottak a feltolt kódot azonnal ki is tehetem éles vagy teszt környezetbe. A legelterjedtebb folyamatos integráció és folyamatos szállítás eszköz a Jenkins, ezen kívül elterjedt még a Buddy, GoCD, CruiseControl.

## 2.2 Nyelvek és keretrendszerek

A webapplikáció elkészítéséhez nagyon sok technológia közül választhatok. Mivel ezek közül nagyon sokkal még nem találkoztam, ezért olyan nyelveket szeretnék választani, amiben kihívást is találok, viszont annyira már ismerem, hogy könnyen neki tudjak kezdeni. Így a backend vonalon Python, C#, Java, Kotlin, Ruby, Node.js, a frontend vonalon React.js, Angular, Node.js, Django a számomra elérhető választék.

2021-re a világon legelterjedtebb programozási nyelv a Python.[15] A Python egy egyszerűen megtanulható és könnyen olvasható, illetve írható programozási nyelv. A nyelv egyszerűsége miatt a szintaxisok könnyen elsajátíthatóak így a programozásba fektetett idő csökken. A Python egy nem típusos nyelv így nem kell azzal időt fecsérelni, hogy az egyes változóknak típust adunk. A Python ezen kívül egy nyílt forráskódú nyelv, amit ingyen lehet használni. A nyelv hatalmas belső könyvtárral rendelkezik, így nincs szükség a külső függőségekre. Amennyiben mégis olyan függőségre lenne szükségünk, ami nem található meg a belső könyvtárban a pip nevezetű csomagkezelővel gyakorlatilag bármit megtalálhatunk. A Python könnyen mozgatható különböző rendszerek között amennyiben nem használunk rendszer függő függőségeket. A sok előnyön kívül a Pythonnak vannak hátrányai is. Mivel a Python a kódot futási időben értékeli ki ezért lassú tud lenni. A könnyű programozásnak és szintaktikának másik hátránya, hogy sok memóriát használ. Amennyiben ez számunkra fontos a Python nem hatékony memória kezelésben. A másik hátránya a Pythonnak az adatbázis elérése és kezelése, ami a többi nyelvhez képest primitív így nem is használják vállalati szoftverekben, ahol sok adatbázis elérés szükséges. A futási időben kiértékelt típusosság miatt a szoftverben a legtöbbször RuntimeException-t dob, ami a program leállásához vezethet amennyiben nem kezeljük helyesen.[16]

A második leggyakrabban használatos programozási nyelv a Java.[15] A Java egy objektum orientált programozási nyelv, ami kezelésében és használatában egyszerűnek mondható. A nyelv könnyen megtanulható, valamint egyszerűen karbantartható és jó hibakereső képességű kódot lehet írni vele. A Java kódot könnyű fenntartani és fejleszteni mivel hardver független. A Java egy könnyen olvasható programozási nyelv, ami nagyban hasonlít az ember által írt szövegre, így egy programozásban nem jártas személy is könnyen megérheti. A Java további előnye, a beépített szemetes. Amikor a fordító úgy véli, hogy egy objektumra már nem lesz szükségünk, törli a memóriából. Azt sem szabad kifelejteni, hogy a Java egy folyamatosan karbantartott és fejlesztett nyelv. Amennyiben az egyik verzióban kijön egy hiba azt a következőben gyorsan tudják orvosolni. A Javaban támogatott a többszállúság, így a programunk jobban ki tudja használni a processzor által szolgáltatott szálakat. Adatbáziskezelésben a Java remekel. A JPA segítségével egy remek lehetőséget kapunk adatbáziskezeléshez. Tesztelés

szempontjából mind unit tesztet mind integrációs teszteket is támogat a nyelv. A Javahoz gyakorlatilag bármilyen könyvtárat be tud szerezni a Maven vagy Gradle segítségével. Ennek ellenére a Javanak is vannak hátrányai. C és C++-hoz képest jelentősen lassabb és több memóriát fogyaszt. A beépített szemetes miatt is tud performancia probléma fellépni, mivel, ha éppen kitörölte, ami nekünk kellett volna az objektumot újra létre kell hoznia.[17]

Szintén a JVM-en futó programozási nyelv a Kotlin. Manapság mobil fejlesztésre nagyon elterjedt, ami számomra is igen hasznos lehet amennyiben egy applikációt is szeretnék fejleszteni a weboldal mellett. Nagy előnye, hogy a Java-val kéz a kézben is használható, így nem kell feltétlenül az egész applikációt Kotlinban írni. Könnyen megtanulható, hisz szintaktikában nagyon hasonlít a Java nyelvre. Megbízható hisz már lassan 10 éve fejlesztik és használják. Egyik nagy hátránya, hogy viszonylag kevesen használják. Ezzel együtt jár, hogy kevesebb oktatóvideó, lecke található meg róla az interneten, illetve, ha bármit meg kell keresnünk nem lesz olyan egyszerű dolgunk, mint egy hagyományosnak mondható nyelvvel. Ez nagyban le tudja lassítani a fejlesztés sebességét.[18]

A C# egy szintén gyakran használt programozási nyelv. A legismertebb vele használt keretrendszer a .NET. A kettő manapság kéz a kézben jár ezért én is együtt fogok hivatkozni rájuk. A .NET 2002-ben a C# megjelenésekor jelent meg. Előnye, hogy egy típusos, objektum orientált programozási nyelv. Szintén rendelkezik saját memóriakezeléssel így a felhasználónak/programozónak ezzel nem kell foglalkoznia. Könnyen fejleszthető, jól olvasható kódot lehet írni vele, ezen kívül a sok könyvtár segítségével sok funkciót egyszerűen lehet fejleszteni. További előnye, hogy nem számít milyen platformon futtatjuk és fejlesztjük a kódot, elég, ha a rendszeren megtalálható a .NET keretrendszer.[19] A saját fejlesztői környezete, a Visual Studio, is egy nagy előny ugyanis egy gyors, könnyen megtanulható és hatékony platformot fejlesztett a Microsoft. A .NET nagyon jól használja ki a gyorsítótárat, amit a fejlesztő méretre szabhat így skálázhatja a programot. Hátránya, hogy egy Microsoft által fejlesztett és karbantartott nyelv, így bármi változás a Microsoftnál befolyásolhatja a nyelv fejlődését is. A fejlesztés licenszekhez kötődhet így megeshet, hogy komoly pénzeket kell kifizetni egy vállalati szoftverért. További hátránya, hogy egyes új kiadásoknál sok probléma tud kijönni, amit viszonylag lassan és nehézkesen javítanak. Amennyiben .NET-et használnék, figyelnem kell, hogy a legújabb verzióra ne azonnal frissítsek, az esetleges hibák ne jöjjenek ki a programban.[20]

Ruby programozási nyelv egy 1995-ben megalkotott nyelv. A mára elhíresült Ruby on Rails keretrendszer 2004-ben készült el, és azóta a Ruby és a Ruby on Rails kéz a kézben fejlődnek. A Ruby on Rails egy web fejlesztő eszköz, ami mára az egyik legelterjedtebb a piacon. Nagyon gyorsan lehet vele fejleszteni, egy MVP-t vagyis egy Minimal viable product-ot talán a leggyorsabban a Ruby segítségével lehet elkészíteni. Működésében egy MVC architektúrát követ, azaz a model-view-controller felépítést. A Ruby egyik nagy előnye, hogy nagyon nagy és aktív fejlesztői közössége van. Ha bármi fellép fejlesztés közben azt egy gyors kereséssel meg lehet találni. Ezek mellett egy könnyen megtanulható nyelvről van szó, amit a kezdőknek előszeretettel ajánlanak. Ennek ellenére

a Rubynak is megvannak a hátrányai. Egy ezek közül a performancia. Mivel nagyon gyorsan és könnyen lehet vele fejleszteni ennek egy hátránya lesz, hogy teljesítményben nem jeleskedik. Nagyon sok mindent elvesz a felhasználótól, így az optimalizációra nincs lehetőség. Másik hátránya a folyamatos fejlődés. Egyértelműen nem lehet a hátrányok közé sorolni azonban amikor egy frissítéssel a saját applikációdat ilyen nagy mértékben meg kell változtatnod az egy kezdő programozónak nehézkes. Ezen kívül a legtöbb könyvtár is folyamatos fejlődésben van, újabb és újabb verziók jönnek ki, amikben szintén egyes frissítésekor olyan változtatások lehetnek, amit nehéz kiszúrni, azonban az applikáció működését befolyásolhatja.[21]

Az Angular egy nagyon elterjedt webapplikáció fejlesztő keretrendszer. Eredetileg Angular 1 néven megjelent framework javascriptet használt azonban az Angular 2 megjelenésével átváltottak TypeScriptre. Az Angular egy MVC azaz model-view-controller architektúrát használó keretrendszer. Támogatja a függőségek befecskendezését így nagyban javít a kód és program megbízhatóságán. A keretrendszer TypeScript használata miatt típusos nyelvet kapunk, ami Javascript kódra fordul. A Typescript miatt egy biztonságosabb, fordítási időben kijövő típusos hibáktól mentes kódot kapunk. Segítségével időt spórolhatunk meg, mivel nem futási időben jönnek ki a hibák, mint Javascriptben. Hátrányai közé tartozik, hogy egy kezdő sokkal lassabban tudja megtanulni a nyelvet React vagy Vue keretrendszerhez képest.[22]

Manapság nagyon sokan használják továbbá a ReactJS nyílt forráskódú JavaScript könyvtárat. Célja felhasználói felületek gyors és egyszerű létrehozása. Nagy előnye, hogy gyors megtanulást ígér és könnyű használatot. Dinamikus felhasználói felületek fejlesztésére sokat használják ugyanis a könyvtár itt nagyban segít. Teljesítményben is jeleskedik mivel virtuális DOM-ot használ. Ezen kívül könnyen tesztelhető kódot tudunk vele írni. Hátrányai közé tartozik, hogy dokumentációban nem jeleskedik, illetve, hogy az MVC modellből kizárólag a view rész fejlesztésére alkalmas, bármi másra muszáj más technológiát használni.[23]

Vue.js egy JavaScript keretrendszer, amit webes felületek és egy-oldalas alkalmazások fejlesztésére használnak. Ezen kívül mobil applikációk fejlesztésére is kiválóan alkalmas az Electron keretrendszerrel párosítva. Előnyei közé tartozik a mérete. A letöltött könyvtár összesen 18 kilobájt helyet foglal. A komponenseket újra tudjuk használni a kód több részén, nem kell mindenhova újra megírni. Ez nagyban segíti a kód olvashatóságát is, mivel nem elszórva léteznek a komponenseink, hanem fixen egy helyen. Ennek további előnye a jó tesztelhetőség. A Vue.js egyik hátrányai közé tartozik, hogy mivel Kínában nagyon elterjedt ezért nagyon sok anyag, tanulnivaló, információ, dokumentáció kínai nyelven íródott. További hátránya, hogy flexibilitásban, nagyobb projektek használatában kevésbé előnyös. Mivel egy viszonylag friss keretrendszer így az elérhető anyagok is limitáltak.[24]

Egy további nagyon gyakran használt eszköz a Node.JS. A NodeJS egy nyílt forráskódú JavaScript futás idejű környezet. A nagy előnye a valós idejű feldolgozásban rejlik. Ebben a részben kifejezetten jól teljesít a többi javascriptet használó eszköz vagy keretrendszerhez képest. Könnyű megtanulni és fejleszteni benne, az applikáció válasz

idejét nagyban javíthatja. A Quick Cache miatt a töltési idők is nagyban csökkennek, illetve platformközi applikációk fejlesztésében jeleskedik. Hátrányai, hogy amennyiben nehéz számítási feladatokat végez, nagyban romlik a performancia. Aszinkron programozási modell miatt viszonylag nehéz fenntartani a kódot, illetve mivel sok könyvtárt nem támogat így vigyázni kell, hogy akkor ne használjuk, ha a nem támogatott könyvtárak közül már használunk egyet. Mivel kevés az igazi NodeJS expert ezért támogatásban is vannak hiányosságai.[25]

Django keretrendszer egy Pythonban megírt front-end fejlesztő keretrendszer, amit előszeretettel használnak nagy streaminget használó szolgáltatók, mint például a youtube, netflix, instagram. Mivel a keretrendszer Pythonban íródott ezért hasonlóan egyszerű a használata. Könnyen megtanulható és használható. A keretrendszer MTV (model, template, view) architektúrát használ, ami nagyban meggyorsítja a fájlátvitelt. Ezen kívül nagy előnye a skálázhatóság és a biztonság. Hátránya, hogy monolitikus, azaz tartalmaz néhány előre definiált változót és fájlt, amit ismernünk kell mielőtt nekiállnánk a fejlesztésnek. Nem ajánlják kisebb projektek fejlesztésére, mivel nagy erőforrást elvesz a keretrendszer és egy kisebb projekt ezt kevesebb erőforrásból is meg tudná oldani.[26]

## 2.3 Architektúra

A webalkalmazást többféle architektúrában is meg lehet valósítani. Az egyik megoldás a monolitikus architektúra. Ebben az esetben egy szervízben van megoldva a program, nincsen logikailag szétválasztva. Előnye, hogy a fejlesztése jóval egyszerűbb, mint egy microservice architektúrának, a tesztelése is jóval könnyebb, illetve a használatban is megkönnyíti a dolgunkat. Ezzel szemben nagy hátránya, hogy nehezen skálázható, megbízhatóságban nem megfelelő ugyanis, ha egy része leáll az egész rendszerünk leáll, illetve a folyamatos telepítés nagy erőforrást igényel. A másik megoldás a microservice architektúra. Ez manapság egy nagyon elterjedt megoldás mivel a monolitikus architektúra problémáira ad megoldást. A legnagyobb előrelépés a megbízhatóságban tapasztalható, ugyanis, ha az egyik szerviz leáll attól még a többi szerviz működhet probléma mentesen. Az egész programunk nem áll le ilyen esetekben. Fejlesztési sebességben is tapasztalhatunk javulást, ugyanis egy esetleges új csapattagnak nem kell az egész programkódot megértenie, elég, ha a saját szervízét megérti. Rugalmasságban is nagy az előrelépés. Egy microservice architektúrába gyakorlatilag bármikor beilleszthetünk egy új szervízt, míg ez egy monolitikus rendszerben nagyon sok időt és energiát tud felemészteni.[27]

## 2.4 Perzisztencia

A hagyományos relációs adatbázisok már több mint 40 éve jelen vannak. Ez nagy fokú biztonságot ad a fejlesztőnek, mivel tudni lehet, hogy megbízható, amit használnak. A relációs adatbázisokban táblákat tárolunk, amikben fix séma áll rendelkezésünkre. Ennek megváltoztatása nagyon költséges tud lenni egy már futó programban. Ezek a táblák SQL (Structured Query Language) segítségével kezelik az adatokat miközben garantálják az ACID[28] elveket. Hátrányai közé sorolják, hogy nehezen vagy egyáltalán nem

skálázható, ha egy tábla túl nagyra nőne akkor nehéz vele a táblát több kisebbre bontani, illetve a tranzakció száma nem haladhat meg pár ezer tranzakciót másodpercenként.

A NoSQL (Not Only SQL) az utóbbi időben megjelent adatbázis kezelő technológia. Legfőképp a skálázhatósága miatt szokták használni, illetve NoSQL segítségével másodpercenként több millió tranzakciót is tudunk kezelni. Felépítésében is megváltozott ugyanis itt nem strukturált adatok vannak jelen, hanem legtöbbször kulcs érték párok vagy JSON dokumentumok. A NoSQL nem tudja garantálni az ACID elveket cserébe a fokozatos konzisztencia modellt használja.[29][30][31]

A feladatomban szeretnék tárolni relációs, illetve NoSQL adatbázisban könnyebben megoldható adatstruktúrákat. A felhasználók kezelése és tárolása például egy tipikus relációs adatbázisban megoldható feladat míg keresés felgyorsítása és termékek tárolása jobban megoldható gráf adatbázisok használatával.

A relációs adatbázisok felépítésben és használhatóságban rendkívül hasonlítanak így nem tartom fontosnak ezek részletes összehasonítását. A szakdolgozatban PostgreSQL, MySQL vagy Oracle Database relációs adatbázisait fogom használni.

A NoSQL adatbázis ezzel szemben egy komplexebb választási feladat mivel több fajta létezik belőle, amik alapjaiban különböznek. Négy elterjedt változatot különböztetünk meg.

A kulcs-érték típusú adatbázis egy kulcs-érték párokból álló adatszerkezetben építi fel az adatbázist. A kulcsokat ismerjük, azzal tudjuk lekérni az értékeket. Számomra ez egy kevésbé hasznos adatbázis lenne. Ilyen például a Redis vagy az Oracle NoSQL.

A dokumentum-adatbázis szintén kulcs-érték párokat tárol azonban ezeket gyűjteménybe szervezi. Itt már a dokumentum bármely attribútum alapján lekérdezhető. Ilyen adatbázis például a MongoDB vagy a MarkLogic.

Az oszlopalapú adatbázis hatékonyan tárolja az adatokat és hasznos abban az esetben amennyiben egy oszlopon szeretnénk egy lekérdezést elvégezni. Ritkán használt adatok esetén hatékony. Oszlop alapú adatbázis például a Cassandra vagy a Hadoop.

A gráf alapú adatbázis csomópontokból és élekből felépülő adatbázis, ahol az adatok egymással kapcsolatban állnak. Összetettebb kapcsolatok egyszerű tárolását teszi lehetővé, illetve ajánlásokat kifejezetten jól tud nekünk adni.[32] Ilyen például az ArangoDB, Neo4j.

## 2.5 Szerver

A szerver kiválasztása két lehetséges megoldás közül választható. Az egyik a cloud megoldás, amikor a felhőben futtatjuk a szerverünket, a másik megoldás a saját szerver kiépítése.

A két megoldásnak külön-külön megvannak a saját előnyei és hátrányai így egyértelműen nehéz kijelenteni, hogy az egyik jobb lenne, mint a másik.

A felhő alapú szerver legnagyobb előnye az egyszerűség. Akár pár kattintással létre tudunk hozni egy ilyen szervert, ami onnantól kezdve a felhőben fut. A cloud megoldások további nagy előnye a szerverek elérhetősége. Például az AWS EC2 cloud szerviz 99.99%-os elérhetőséget garantál[33] ami összesen heti 1 perc elérhetőség veszteséget jelent.[34] Ezek mellé viszont nagy hátránya az ár. Egy on-premise megoldáshoz képest sokszoros befektetést jelent.

A saját, helyszíni szerver előnye, hogy közel van a szerver. Ez biztonsági szempontból sok cégnek nagyon fontos lehet. Ez akár előny akár hátrány is tud lenni ugyanis egy cloud megoldásnál a biztonságot a szerver biztosítja, nem kell erre saját megoldást kitalálni. A helyszíni szerverek hátránya, hogy rendszergazda kell az üzemeltetéséhez és kiépítéséhez. A rajta futó programokat nekünk kell folyamatosan frissíteni és napra készen tartani, nekünk kell a hardware-el foglalkozni, kiépíteni, javítani, bővíteni. Összességében úgy gondolom, hogy több vele a munka, mint amennyivel drágább a felhő megoldás.

A szervereket tudjuk futtatni többféle hardware-en. Nekem erre rendelkezésemre áll egy Raspberry Pi ami a saját 2 gigabájtos memóriájával és 1.2 GHZ-es processzorával, elegendő lesz arra, hogy a szakdolgozat készítése alatt futtassa a szerverem.

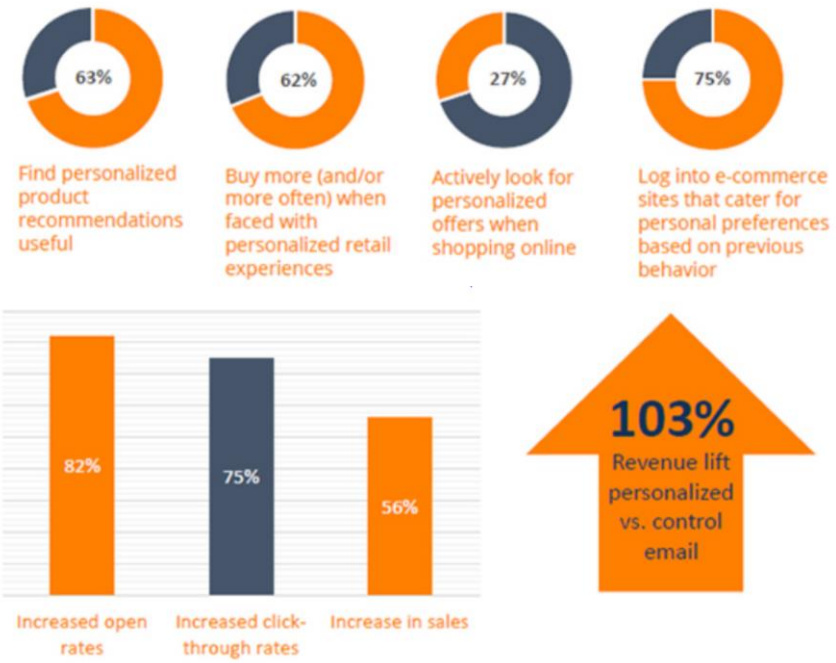
## 2.6 Portál motorok

Az egyik legjobban elterjedt portál motor a Wordpress. A Wordpress sablonok alapján pillanatok alatt elkészít az ember számára egy kész weboldalt. A Wordpress hátránya viszont, hogy nem rendelkezik olyan funkcionalitással, ami nekem lényeges lenne. Az ajánló rendszert nem tartalmazza.

## 2.7 Ajánló rendszer

Manapság az egyik leggyakoribb jelenség az online vásárlásban a személyre szabott ajánlások. Ezek segítségével a felhasználó az érdeklődési körébe tartozó tárgyakat vagy témákat látja, a számára elhanyagolható fontosságú elemek helyett. Ez egy hatalmas iparággá fejlődte ki magát az utóbbi években. Természetesen a piac is reagált erre a lehetőségre és manapság több olyan cég is található, akik kifejezetten ezzel az ágazattal foglalkoznak, például a Magyarországon létrehozott Emarsys Kft. Az ő portfóliójuk alapján egy átfogó ajánlási rendszert kaphat a felhasználó. Mind a weboldalon személyre szabott ajánlatokkal, mind email kampányokkal. A cég bevétele legutolsó adatok alapján meghaladta a 4 milliárd forintot.

Természetesen ez az ügyfélnek is megéri hiszen ahogy a 2.1 ábrán is láthatjuk a bevétel akár 100%-ban is megugorhat a hagyományos email marketinghez képest.




*2.1 ábra Hagyományos email kampány a személyre szabott kampánnyal szemben*

Az ajánlásokkal a közösségi médiában is gyakorta találkozhatunk. Ezek az oldalak a számunkra olyan tartalmakat fogják mutatni nekünk, amivel tudják, hogy értékes percekig akár órákig az oldalon tudnak minket tartani.

Az ajánlási rendszerek gráf NoSQL adatbázisban tárolják a felhasználókról szerzett adatokat. Ezek a gráfok a felhasználó egy-egy érdeklődési körét tartalmazzák, ami alapján kikövetkeztethető, hogy mi lesz az, ami legközelebb fel fogja kelteni a figyelmét.

A leggyakoribb gráf adatbázis ennek használatára a Neo4J azonban a többi gráf adatbázissal is hasonló eredményeket tudunk elérni. Kifejezett ajánlórendszereket nem találtam az interneten, ezzel vagy cégek foglalkoznak, vagy pedig gyakorlatilag a nulláról felépítést, egy oktatóanyagon keresztül.

Célom egy olyan ajánlási rendszer létrehozása lesz, ami a felhasználónak személyre szóló ajánlásokat tud mutatni. Négy féle ajánlás lesz elérhető: személyre szabott, hasonlók által kedvelt, nézd meg még egyszer és a próbálj ki valami újat. Ezeket mind saját logikával fogom felépíteni az adatbázisomban így harmadik féltől nem fogok függeni.

# 3. SPECIFIKÁCIÓ

## 3.1 Kinézet

### 3.1.1 Főoldal

A bejelentkezett, illetve nem bejelentkezett felhasználók a főoldalon azonos tartalmat tudnak fogyasztani. Az egyetlen különbség, hogy a bejelentkezett felhasználók a profiljuk gombját fogják látni a jobb felső sarokban, míg egy ismeretlen felhasználó a bejelentkezés és regisztrációs űrlapot tudja elérni ott.

A weboldal, illetve cég LOGO-ja a bal felső sarokban lesz megtalálható. Amennyiben a felhasználó erre rányom bármely ablakban a főoldalra fogja visszavezetni.

A fő menüpontok a Felfedezés, amely a felfedezés oldalra navigál, a Kategóriák, amely a kategóriák oldalra navigál, illetve az Árak, amely a kölcsönzés árairól információt adó oldalt tudja elérni.

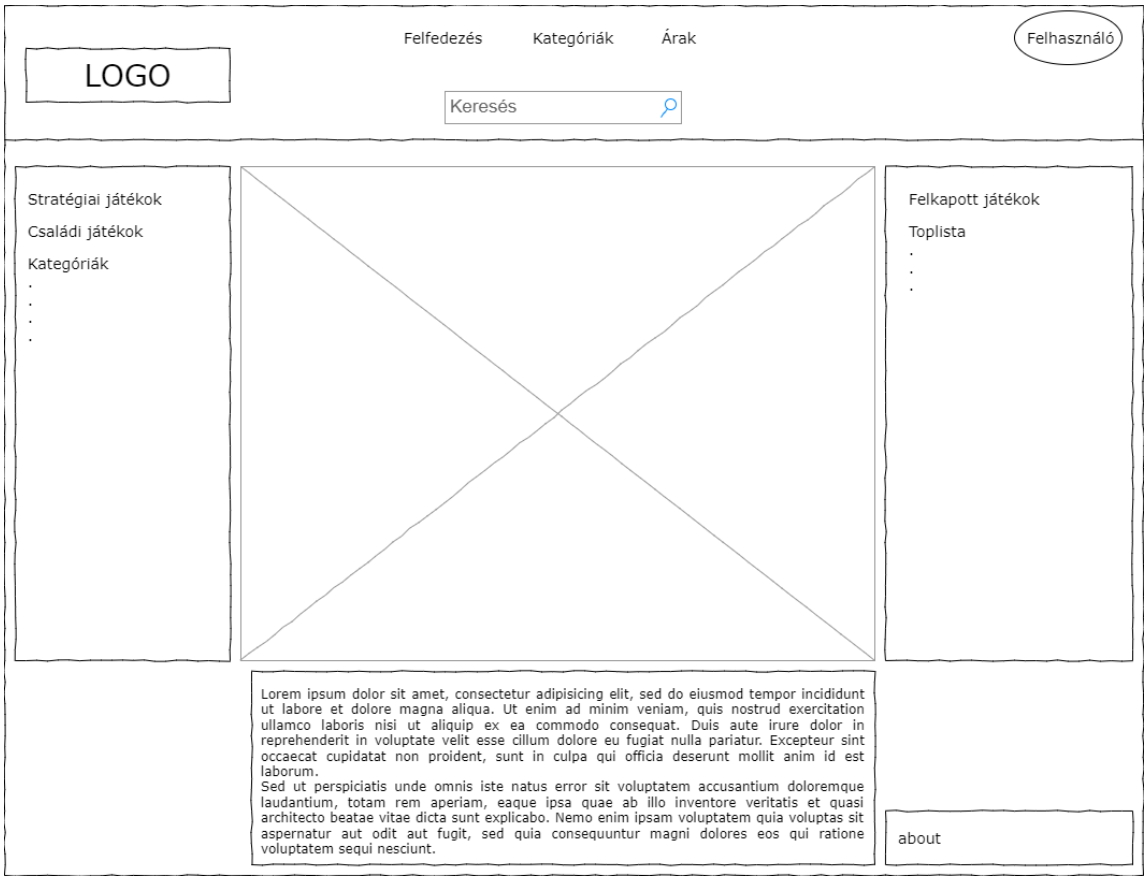
A Keresés mezőre kattintva, előjön a keresési űrlap.

A főoldal bal oldalán a játékok kategóriái, illetve főbb a társasjátékkal kapcsolatos szűrőket tartalmazza.

A weboldal tartalma társasjátékok hírei, társasjáték ajánlások, és a weboldal újdonságait fogja tartalmazni.

A jobb oldalon található részben, a Toplistát, illetve fontosabb szűrési feltételeket lehet elérni.

Az oldal alján a kapcsolat menüpont található, amely előhozza az elérhetőségeket.

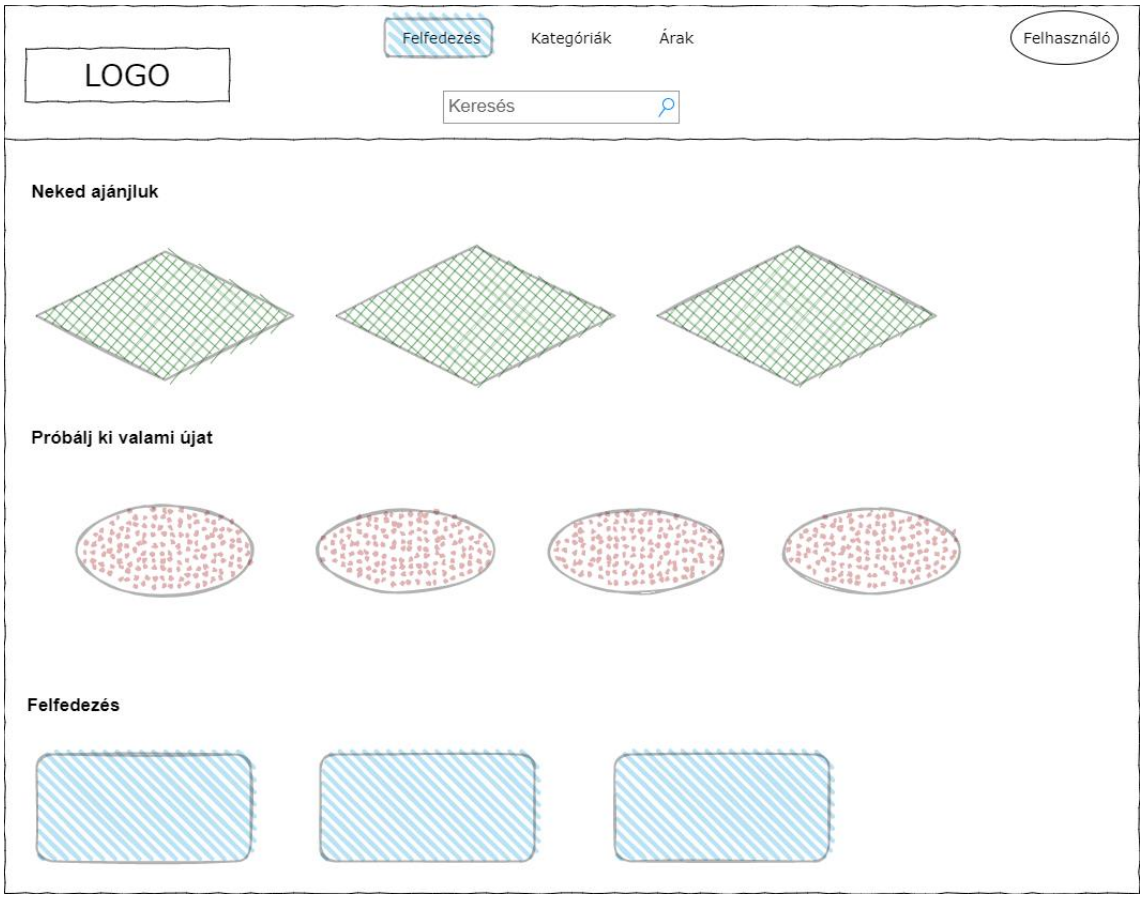



*3.1 ábra Főoldal drótkeret rajza*

### 3.1.2 Felfedezés

A felfedezés fülön a nem bejelentkezett felhasználók egyedül a Felfedezés kategóriát tudják megtekinteni. Ebben a kategóriában a leggyakrabban bérelt játékok, illetve az újdonságok lesznek megtalálhatóak.

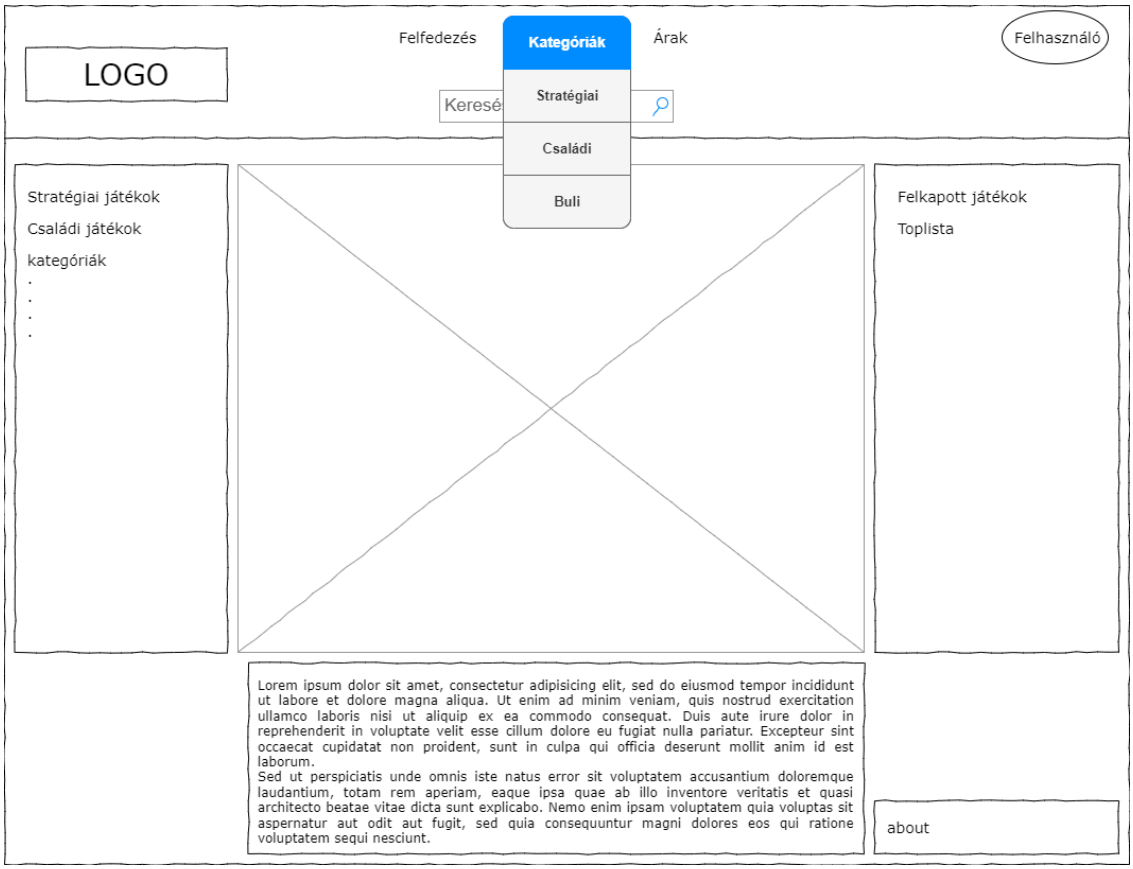
Amennyiben a felhasználó már be van jelentkezve, azon kívül, hogy a felfedezés kategóriát megtekintheti, kettő csak számára elérhető kategóriát is láthat. A neked ajánljuk kategória alatt a rendelései alapján hozzá illő társasjátékok lesznek megmutatva. A Próbálj ki valami újat kategóriában, olyan társasjátékokat fog találni, amire látszólag nem számíthat, hiszen érdeklődési köre alapján, nem neki valóak. Ebben a kategóriában amennyiben rákattint egy játékra, azonnal bekerül az adatbázisba, hogy milyen társasjátékok azok, amik a komfort zónáján kívül is érdeklik.




*3.2 ábra Felfedezés drótkeret rajza*

### 3.1.3 Kategóriák

A kategóriák fül alatt nem teszünk különbséget bejelentkezett, illetve ismeretlen felhasználó között. A Kategóriák egy lenyitható fül, amiben az elérhető kategóriák közül válogathat a felhasználó. Amennyiben kijelöl egyet az oldal azon kategóriába tartozó társasjátékokat fogja megmutatni neki.




*3.3 ábra Kategóriák drótkeret rajza*

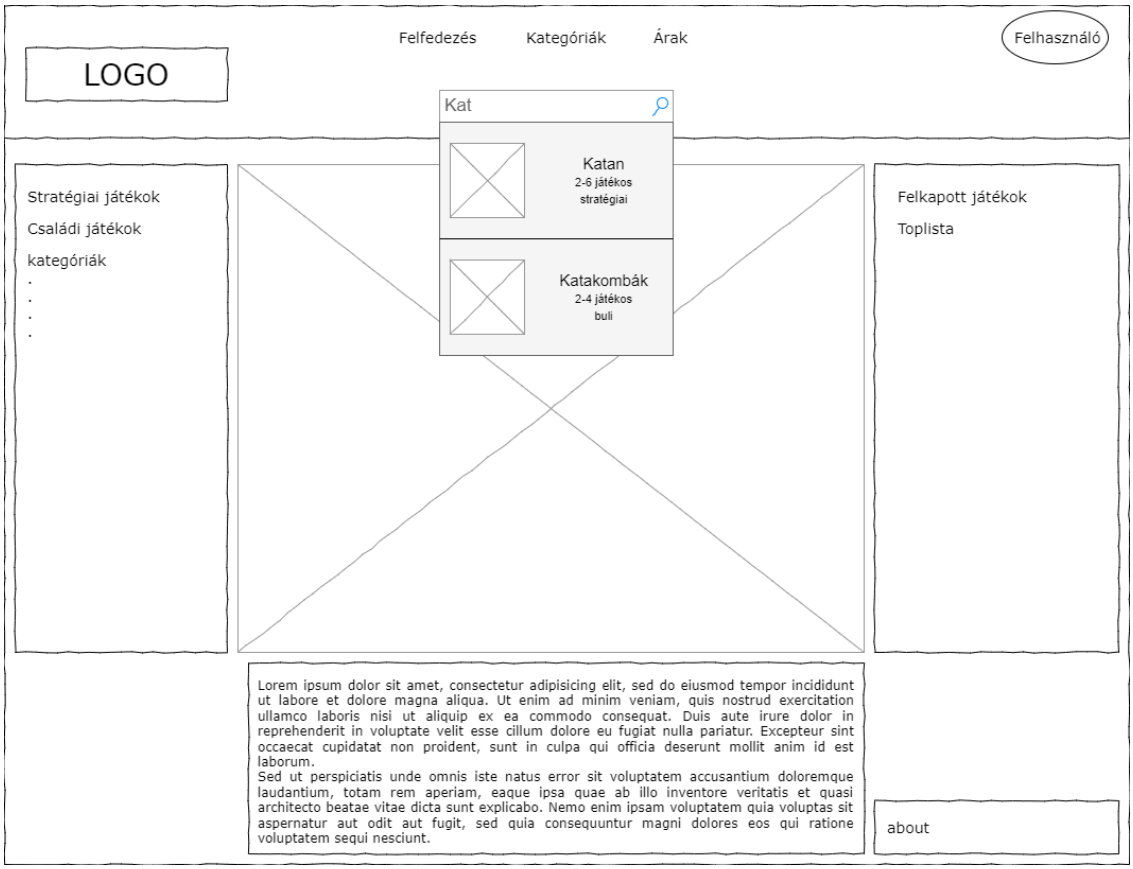
### 3.1.4 Árak

Az Árak szintén egy olyan oldal, ahol nem teszünk különbséget regisztrált és ismeretlen felhasználó között. A weboldalon ez egy statikus oldalként fog megjelenni, ugyanis itt kizárólag a bérlések részletes árazását tudja elolvasni a felhasználó.


*3.4 ábra Árak drótkeret rajza*

### 3.1.5 Keresés

A keresés mezőre kattintva előjönnek a keresési javaslatok, amik a leggyakoribb keresések, illetve a bejelentkezett felhasználók érdeklődési körének megfelelő társasjátékok. Amennyiben a keresés számára nem megfelelő, át tud navigálni a részletes keresés oldalra, ahol a név alapú keresésen kívül, kiadás éve, játékosok száma, kategóriák, játékidő alapján is tud keresni.



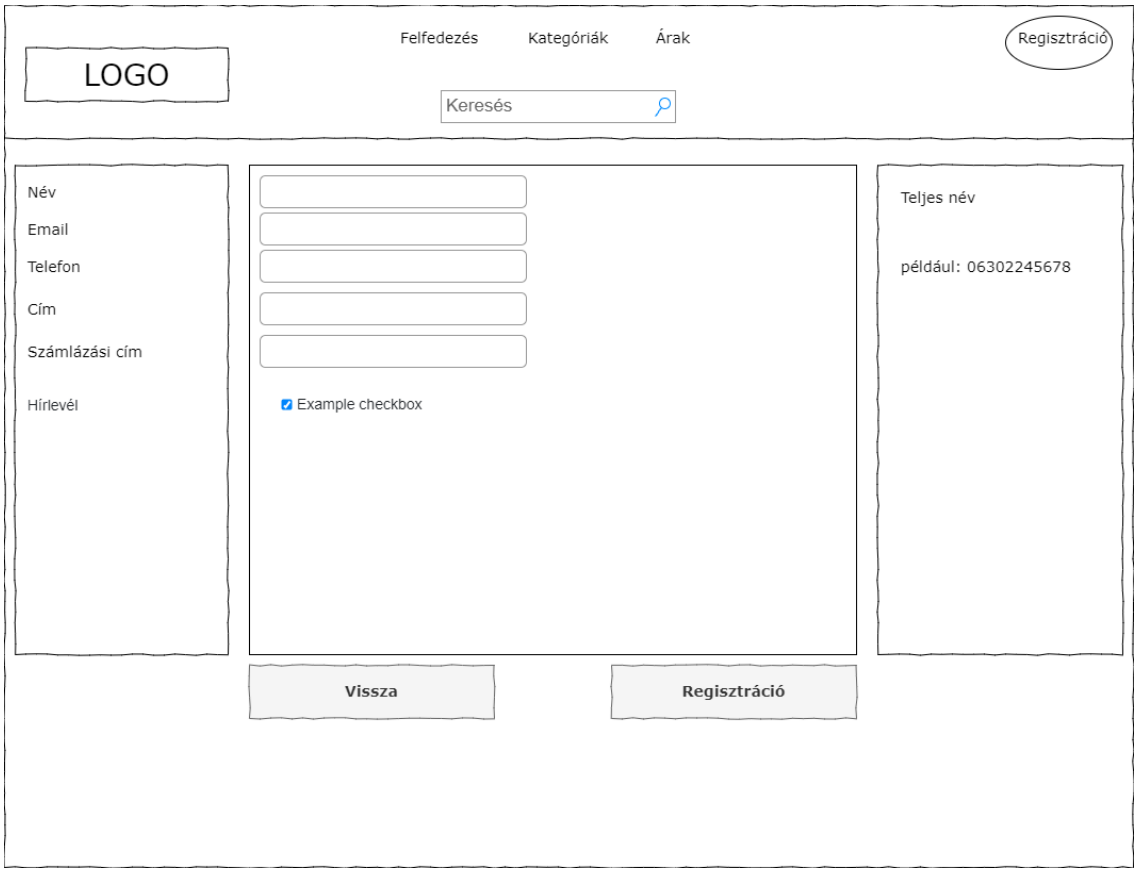

*3.5 ábra Keresés drótkeret rajza*

### 3.1.6 Regisztráció

A regisztráció fül a be nem jelentkezett felhasználóknak elérhető. Ezen a fülön előjön egy regisztrációs űrlap, ahol az adatok megadásával új felhasználói fiókot hozhat létre a felhasználó.

Az oldalon egyértelműen látszódik, hogy melyik mezőbe milyen adatot kell írni, ezen felül a mezőtől jobbra példákkal fogom szemléltetni a mező tartalmának lehetséges kitöltését.

A regisztráció az e-mail visszaigazolásához kötött, amelyet a regisztráció gombra kattintva egy információs oldal is a felhasználónak jelez. Amennyiben visszaigazolta a regisztrációt, a felhasználó be tud jelentkezni.

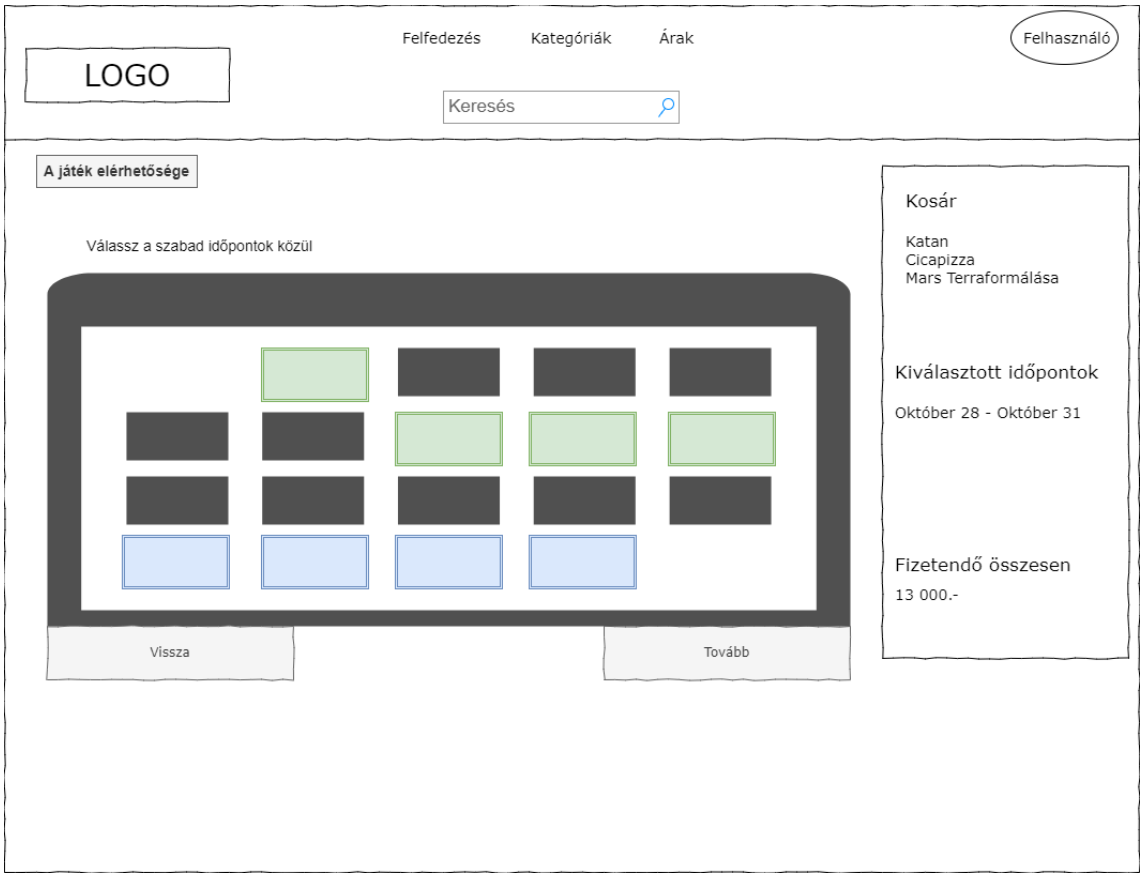



*3.6 ábra Regisztráció drótkeret rajza*

### 3.1.7 Bérlés

A bérlés oldalon egy intuitív jól olvasható naptár fog megjelenni, ahol egyértelműen leolvasható lesz a szabad időpontok helye, illetve kattintással ki tudja választani a felhasználó, hogy ő mikor szeretné kibérelni a társasjátékot. A jobb oldalon egy összefoglaló lesz látható a rendelés információival, többek között egy-egy időpont kiválasztásával a bérlés ára dinamikusan fog változni. A bérlés ára a bérlés hosszától, illetve a társasjátékok minőségi besorolásától fog függeni.

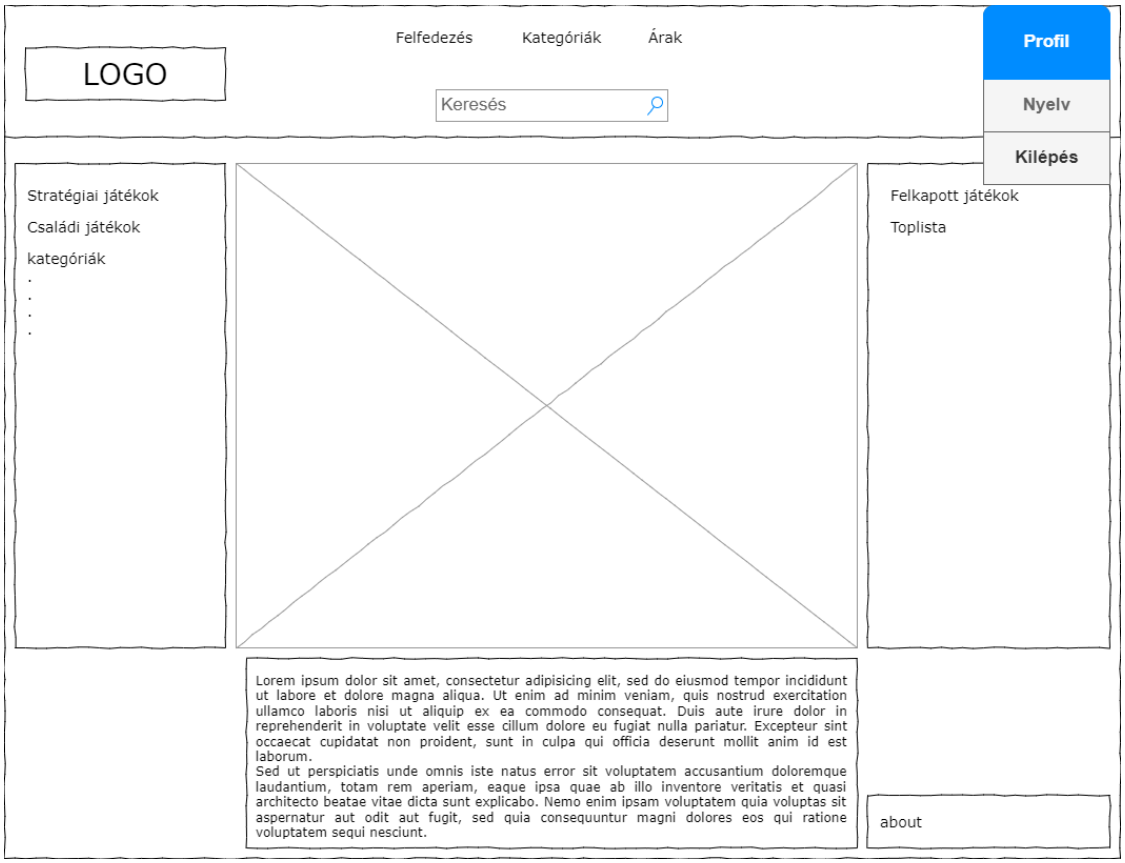
Természetesen az oldal csak bejelentkezett felhasználóknak lesz elérhető.




*3.7 ábra Bérlés drótkeret rajza*

### 3.1.8 Felhasználói menü

A felhasználó menü egy nagyon egyszerű menüpont, amely a bejelentkezett felhasználóknak lesz elérhető, amennyiben a jobb felső sarokban rányomnak a profiljukra. A három menüpont a profil, a nyelv, illetve a kijelentkezés gomb közül választhatnak. A Profil gomb megnyomásával a felhasználó oldalra tud navigálni a felhasználó, a Nyelv gombra kattintva ki tudja választani a kívánt nyelvet, a Kijelentkezés gombra kattintva pedig bejelentkezett felhasználóból, ismeretlen felhasználóként folytathatja a böngészést.

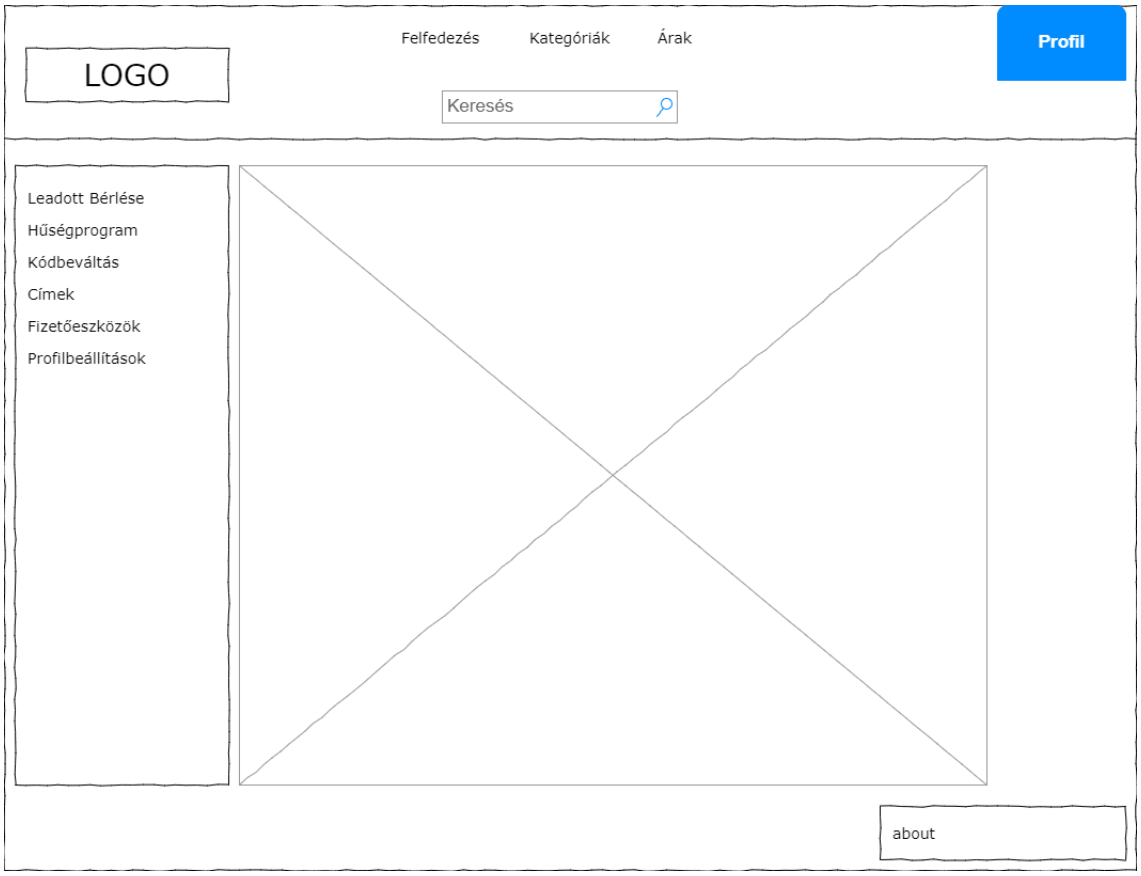



*3.8 ábra Felhasználói fül drótkeret rajza*

### 3.1.9 Felhasználói oldal

A felhasználó oldalon a bejelentkezett felhasználók a saját adataikat tudják megtekinteni, illetve módosítani. A bal oldalon található menüsorból tudja kiválasztani a kívánt menüpontot. A profilbeállítások menüpont alatt található információkat csak a jelszó megerősítése után tudja módosítani a felhasználó.

A beállítások egy listában fognak megjelenni, az információ nevével a bal oldalon, illetve tőle jobbra egy módosítható területen az információ tartalmával.




*3.9 ábra Felhasználó oldal drótkeret rajza*

## 3.2 Funkciók

A weboldal funkciójában nagyon hasonlítani fog egy hétköznapi ételrendelő vagy gépkocsi kölcsönző oldalhoz. Funkciójában is ezekhez fog legjobban hasonlítani.

### 3.2.1 Regisztráció

A felhasználóknak lehetősége lesz egy regisztrációs űrlap kitöltésére, amivel létrehozhatnak egy saját felhasználói fiókot. A rendelés leadása regisztrációhoz kötött lesz, míg a weboldal megtekintése és az ajánlatok átnézése mindenki számára nyitott.

### 3.2.2 Felhasználói fiók

A regisztrált felhasználók beléphetnek a saját fiókjukban, ahol a profiljuk összegzését fogják találni. Ezen kívül több menüpont közül választhatnak az oldalon.

A Leadott Bérlések menüpont alatt a már lejárt bérléseiket tekinthetik meg. Ugyan itt, ha szeretnék, a rendelést újra leadhatják, ekkor ugyan azok a társasjátékok kerülnek a kosárba, mint a kiválasztott rendelésben.

A Hűségprogram menüpont alatt az elérhető hűségpontjait láthatják a felhasználók. Ez elsősorban a sokat rendelő vendégeknek fog kedvezni, hiszen nagyobb kedvezményekre is tudják váltani a rendelésekből bejövő hűségpontjaikat.

A Kódbeváltás menüpont alatt a promóciós kódokat tudják érvényesíteni a felhasználók.

A Címek menüpontban a felhasználó kiszállítási címeket tud megadni és be tudja állítani az elsődleges kiszállítási címet. Az elsődleges kiszállítási cím fog megjelenni a rendelésnél alapértelmezetten.

A Fizetőeszközök menüpontban az eltárolt bankkártya adatokat tudja megtekinteni a felhasználó. Bár alapvetően harmadik fél által üzemeltetett fizetési módok lesznek, a felhasználónak lesz lehetősége a belső rendszeren belül is fizetnie. Az elmentett kártyáit itt tudja megtekinteni, törölni.

A Profilbeállítások menüpontban a profilhoz tartozó alapvető információkat tud megváltoztatni és megtekinteni a felhasználó. A jelszóváltás, névváltoztatás, telefonszám vagy emailcím változtatás mind itt lesz megtalálható. Bármiféle módosítás ezen az oldalon csak a jelszó megerősítésével lesz elérhető.

### 3.2.3 Felfedezés

Mind a nem regisztrált mind a regisztrált felhasználók használhatják a felfedezés oldalt azonban más-más tartalommal fog megjelenni nekik. A regisztrált felhasználók saját ajánlásokat fognak kapni profiljuk alapján. Ezen kívül olyan ajánlásokat is fog kapni, ami számára egy új kategóriában lévő társasjáték. Mindezen kívül a top társasjátékok is megtalálhatóak lesznek ezen az oldalon és ajánlások az oldalon megtalálható játékok közül. Akik nem regisztrált felhasználók, elsősorban a személyes ajánlást és az új játékok ajánlását nem fogják látni.

### 3.2.4 Kategóriák

A társasjátékokat kategóriákra lehet bontani. Egy társasjátéknak akár több kategóriája is lehet. A kategóriák fül alatt válogatni lehet a különböző kategóriában megtalálható társasjátékok közül. A kategóriák a következők lehetnek:

- Stratégiai
- Kaland
- Blöff
- Kártyajáték
- Kockajáték
- Fantázia
- Horror
- Orvosi
- Vallási
- Gyerekjátékok

- Kvízjátékok
- Állatos játékok
- Gazdasági
- Humoros
- Partijátékok
- Politikai Játékok
- Puzzle
- Sport
- Utazás
- Videójáték
- Túlélő

### 3.2.5 Keresés

A felhasználó használhatja a beépített kereső funkciót. Ezzel tudja a leggyorsabban megtalálni a kívánt társasjátékát. A kereső nem csak a keresett játékot tudja felajánlani, hanem az ahhoz hasonló játékokat is. A gyorskeresésen kívül, egy keresési oldal is rendelkezésre áll, ahol kategóriára, játékosszámra vagy játékidőre is lehet szűrni.

### 3.2.6 Bérlés

A regisztrált felhasználó miután kiválasztotta a kívánt társasjátékát a kosár megtekintése után, időpontot tud foglalni a rendszeren keresztül. Az időpont foglalás kétféleképpen is lehetséges, a foglalás megkezdése előtt, ekkor a felhasználó csak az általa beállított idősávban elérhető társasjátékokat fogja látni. A második esetben a játékok kiválasztása után is lehetősége nyílik az időpont foglalásra, ekkor csak azokat a szabad helyeket fogja látni, ami a megadott időpontban az összes kosárban lévő társasjátékra érvényes.

## 3.3 Adat gyűjtés

A felhasználók minden kereséséről és társasjáték megtekintéséről értesíteni fogja a rendszert. Minden egyes kereséssel pontosabb és pontosabb képet tudunk kialakítani a felhasználóról. Amennyiben elég terméket tekintett meg, a felfedezések oldalon már számára ismert, kedvelt vagy éppen számára ismeretlen, de más hasonló érdeklődésű körű felhasználó által megtekintett társasjáték ajánlatokat kaphat. A regisztrált felhasználók a felhasználási feltételekben elfogadják, hogy ezen adatokat gyűjtjük róluk. A felhasználók, nem lesznek egyértelműen beazonosíthatóak az érdeklődési körök alapján. Amennyiben egy nem regisztrált felhasználó használja az oldalt, ugyan ezt a funkcionalitást cookie-k segítségével szeretném elérni.

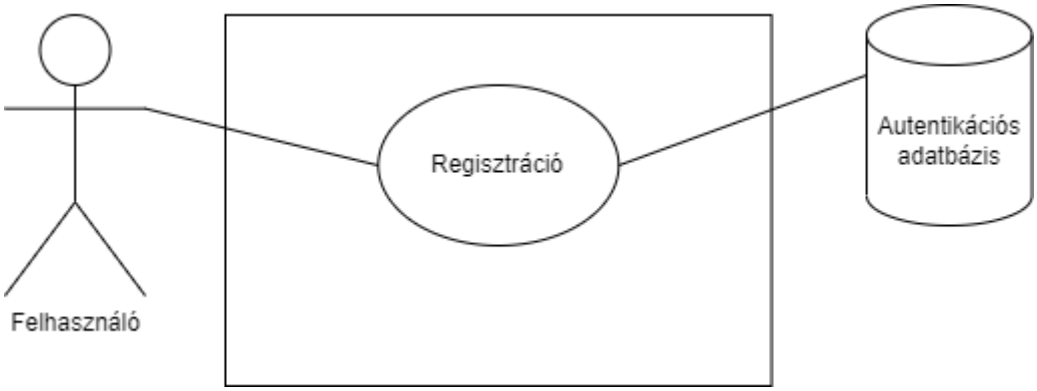
## 3.4 Jogi Háttér

Az Európai Parlament és Tanácsadás 2016/679 Rendelete[35] alapján a felhasználók adatait különös körültekintéssel kell kezelni. Amennyiben egy felhasználó törli a

regisztrációját, az adatai 60 napon belül törlésre kerülnek. Ha ezt külön kérvényezi akkor 30 napon belül köteles vagyok adatait törölni.

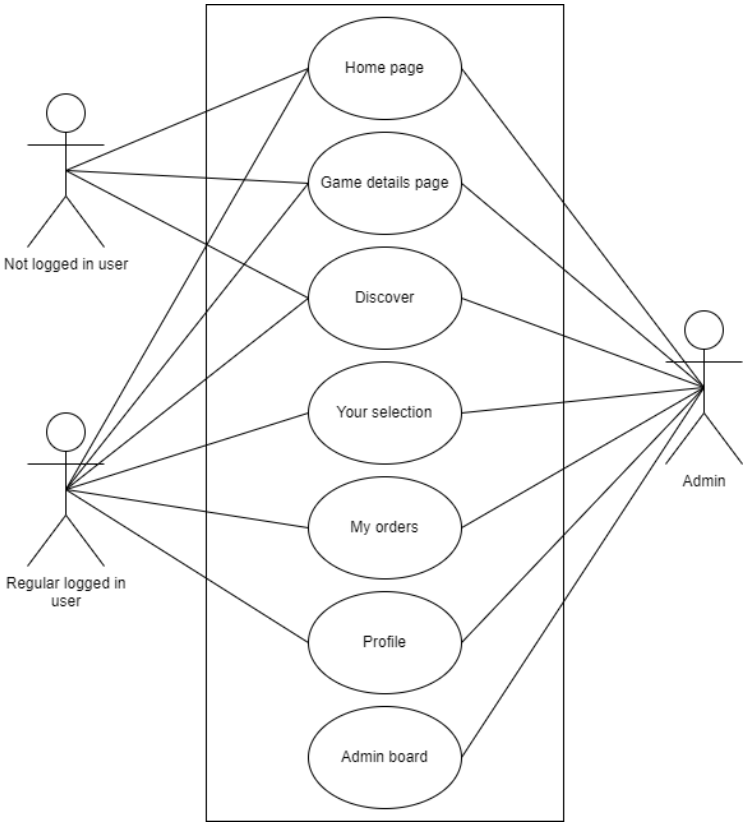
## 3.5 Use-case

### 3.5.1 Felhasználó regisztráció



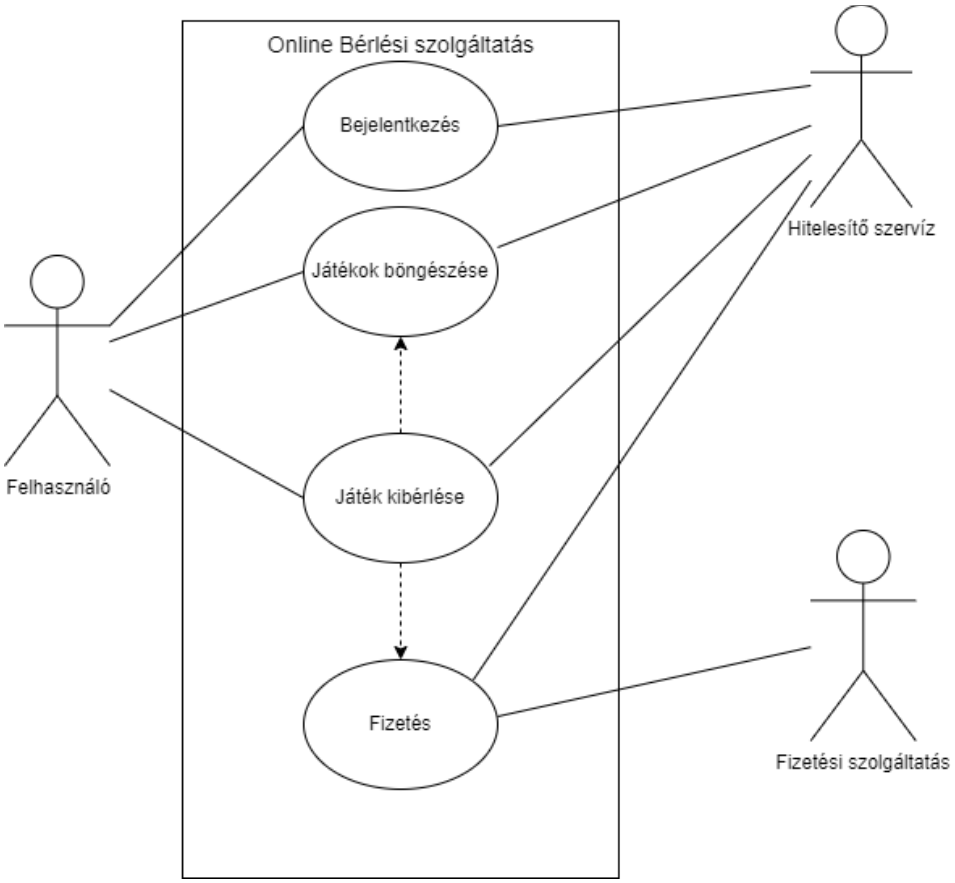

*3.4 ábra felhasználó regisztráció*

### 3.5.2 Felhasználók által elért oldalak




*3.5 ábra felhasználók által elért oldalak*

### 3.5.3 Játék bérlés




*3.6 ábra játék bérlése*

# 4. RENDSZERTERV

## 4.1 Eszközök

A webapplikáció elkészítéséhez egy erre alkalmas és elég erős számítógépre van szükségem. Jelenlegi számítógépem minden igényt kielégítő hardverrel rendelkezik így ebbe nem kell erőforrást invesztálnom. Ezen kívül a szerver futtatására egy Raspberry Pi 2 Model B áll rendelkezésemre, amely keveset fogyaszt és éppen elegendő a számítási kapacitása, hogy a számomra fontos adatbázis lekérdezéseket gyorsan és pontosan kiszolgálja. A fejlesztés a Jetbrains által fejlesztett IntelliJ Idea Community Edition-ben fog történni, illetve a frontend munkák Visual Studio Code-ban.

## 4.2 Irodalomkutatás alapján választott technológiák

Nyelvek tekintetében a back-end vonalon a Java-t választottam, egy jelenleg nagyon elterjedt és felkapott nyelvet. A Java verziói közül a legutóbbi LTS-t, azaz a hosszútávú támogatottságot élvező verziót, preferálom, ami a Java 17. Front-end szempontjából a React-ra esett a választásom. Legfőképpen az egyszerűsége fogott meg és a gyors tanulhatósága.

Architektúra szempontjából monolitikus rendszert szeretnék kiépíteni, úgy érzem, hogy egy webapplikáció számára jelenlegi helyzetemben túlzás lenne a microservice architektúra.

A Java mellett az egyik legelterjedtebb keretrendszert is mindenképp szeretném bevezetni mégpedig a Spring keretrendszert. Eleinte a Spring Boot számomra elég lesz és ahogy bővül úgy fogom a szükséges függőségként hozzáadni.

A modulok és a front-end back-end közötti kommunikációt API hívásokkal fogom megoldani, amik REST-en keresztül fognak történni. A REST egy könnyen integrálható, egyszerűen használható architektúra.

Mivel API hívások fognak futni a szervízemen belül így ezekre a könnyű tesztelhetőség érdekében Postman-t fogok használni. A Postmannal magam hívhatom meg a végpontokat így a hibakeresést felgyorsíthatom.

Függőség kezelésére a Maven-re esett a választásom. Számomra egy jobban átlátható és könnyebben fejleszthető eszköz, mint a Gradle.

Az adatbázisok, mint már említettem egy Raspberry-n fognak futni Ubuntu Server alatt. Erre a Szerverre szeretnék egy Dockert is telepíteni, hogy az adatbázisok könnyen kezelhetőek és mozgathatóak legyenek.

Adatbázisként egy relációs és egy NoSQL adatbázisra esett a választásom. Relációs adatbázisnak, amiben a felhasználók adatait fogom tárolni PostgreSQL-t fogok használni. Míg az ajánlások elkészítésére és a keresésekre ArangoDB lesz a segítségemre.

További eszköz, amit szeretnék bevezetni az a Jenkins. Eleinte el tud venni időt azonban hosszú távon egy nagyon hasznos eszköznek tartom. Szintén tudom futtatni a Raspberry-n így bármikor elérhető számomra. A szoftver tisztaságára tekintettel ez egy nagyon fontos lépés.

## 4.3 Verziókezelő

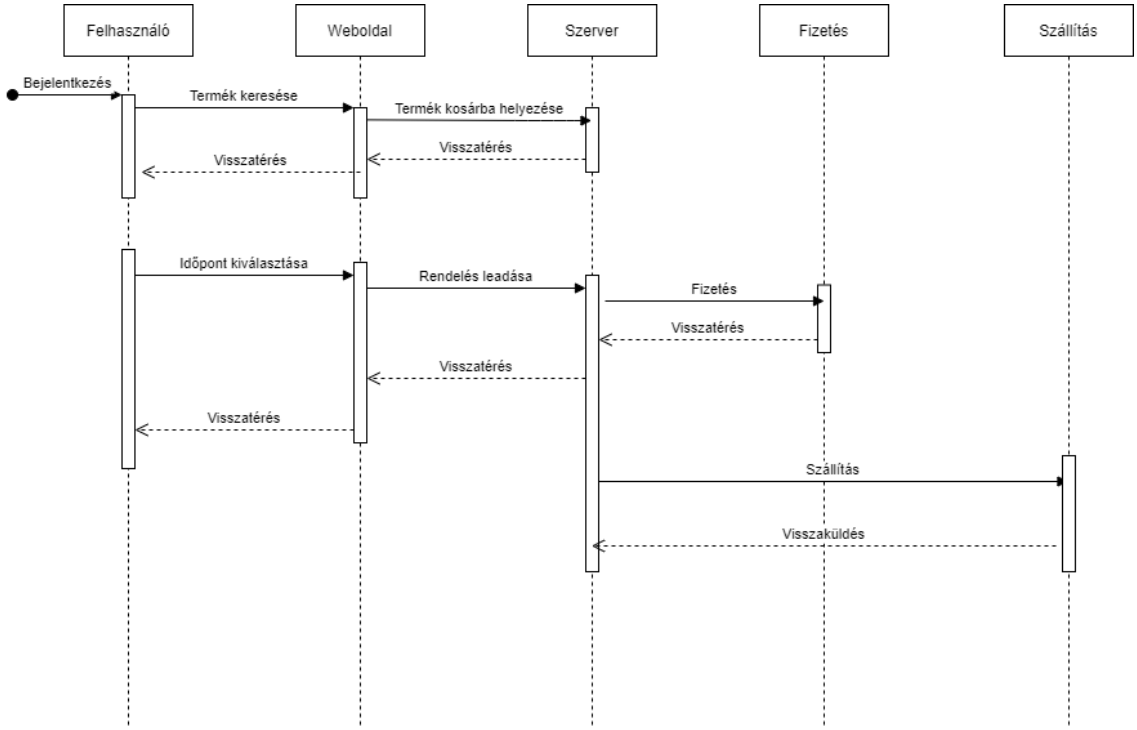
A két verziókezelő közül egyértelműen a Git-et fogom használni. A repository-m GitHub-on lesz elérhető így bármikor tudom fejleszteni másik számítógépről, illetve megoszthatom a kódom másokkal.

## 4.4 Harmadik féltől származó funkcionalitások

Az egyik legfontosabb funkcionalitás a fizetés lesz. Az esetemben a Barion tűnt jobb megoldásnak az olcsóbb kezelési költségük miatt, ez nagyjából 1% körüli összeg tranzakciónkként. A Barion egy kiszervezett API-t biztosít, aminek a segítségével beintegrálhatom a fizetést a weboldalamra.

## 4.5 Szekvencia Diagram

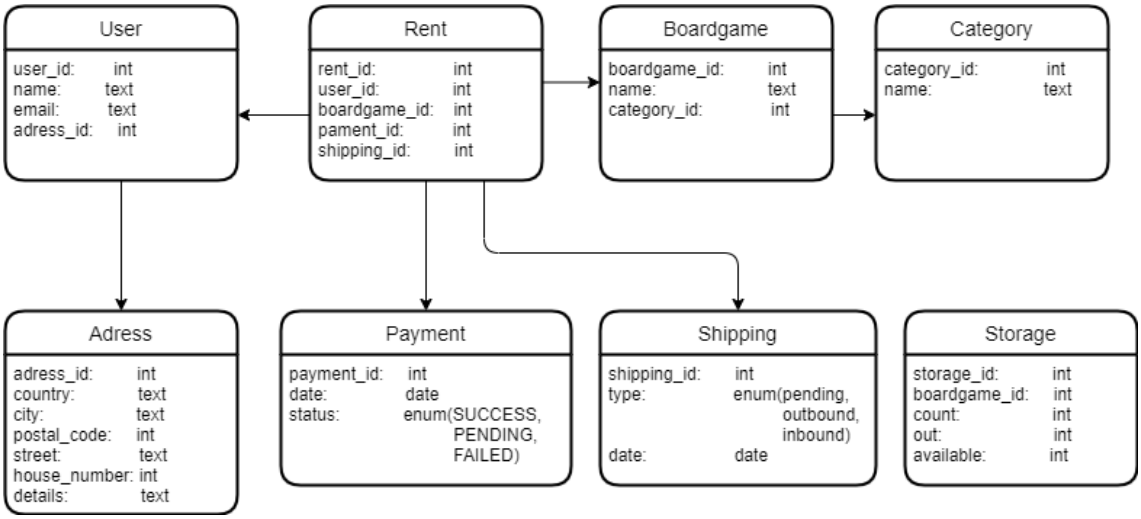
Egy tipikus rendelési folyamat ábrázolása szekvencia diagramon




*4.1 ábra rendelés szekvencia diagramja*

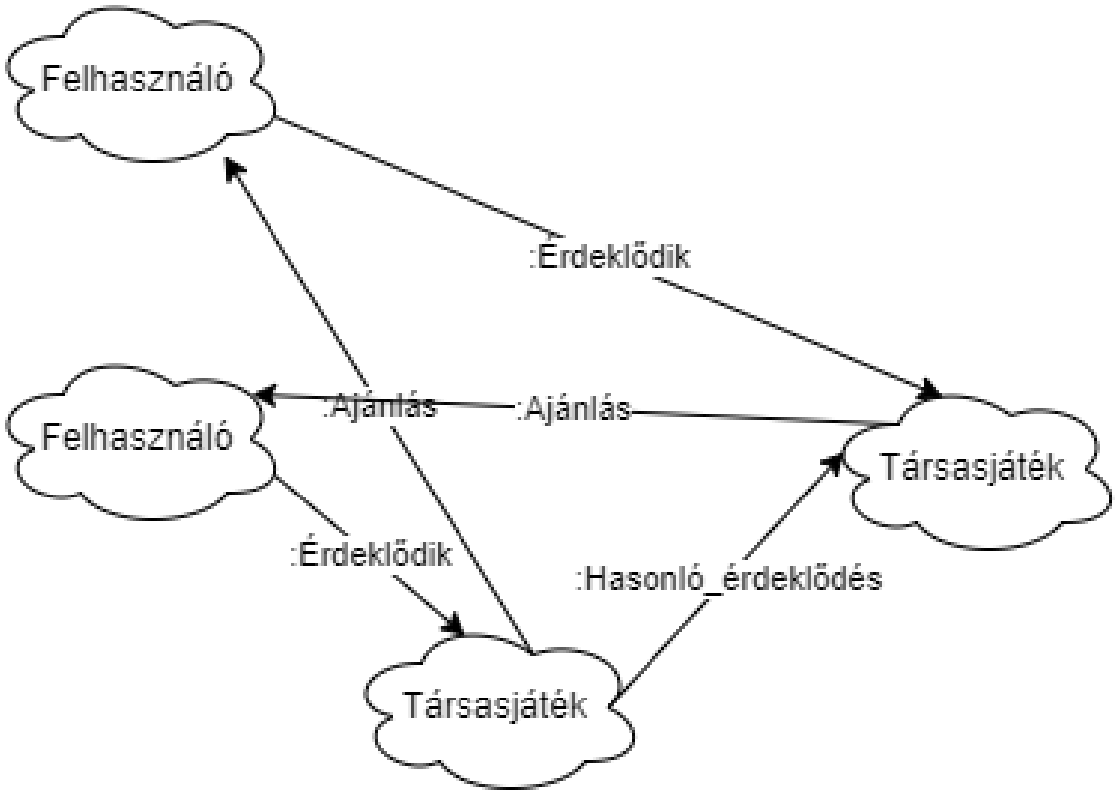
## 4.6 Adatbázisszerkezet

Az adatbázis relációs és NoSQL adatbázis kombinációjával fogom megoldani. A relációs adatbázis a felhasználók tárolására lesz használva. Ennek a szerkezete az alábbi adatbázis diagramon látszódik:




*4.2 ábra relációs adatbázis szerkezet*

Az NoSQL tábla a társasjátékok tárolására lesz használatos. Ebben az esetben egy gráf adatbázist szeretnék elkészíteni. A Gráf adatbázis segíteni fog a keresések, illetve az ajánlások pontosításában.




*4.3 ábra NoSQL adatbázis gráf felépítés terve*

## 4.7 Ajánlás algoritmusa

Az algoritmus a felhasználók adatait fogja felhasználni és az adatok elemzése alapján tud ajánlásokat adni. A felhasználóról érdeklődési köröket tudunk felállítani. Ez a kereséseiből, a megnézett társasjátékok kategóriájából, illetve a vele hasonló felhasználók érdeklődési köréből tevődik össze. Számomra a legfontosabb, hogy a felhasználó a kedvenc kategóriájának a játékait böngészhesse.

Az algoritmus nem egy harmadik fél által tervezett API lesz, hanem én fogom megírni, egy külön service fog ezzel foglalkozni. Az ArangoDB gráf adatbázisa pedig kitűnő segítség lesz ugyanis ezzel eltárolhatom a kapcsolatokat a felhasználók és a társasjátékok között.

A felhasználó többféle társasjátékot kedvelhet. Ezt kétféleképpen tudja kifejezni számomra: Megrendel egy társasjátok vagy rányom egy társasjáték adatlapjára. Amennyiben ezt megteszi, létrejön egy él a felhasználó és a társasjáték között. Ebből az adatból két dolgot nyerek: mely társasjátékot kedveli direktben, milyen kategóriát kedvel. Amennyiben elég adatot nyertem ki egy nagyon kiterjedt és hasznos ajánló rendszert tudok nyújtani a felhasználónak, ahol megtalálhatja a számára kedvelt társasjátékokat és kategóriákat.

A négy féle ajánlást az alábbiakban ismertetem.

### 4.7.1 Neked ajánlott

Ebben az egységben a felhasználó a kategóriák alapján fog ajánlásokat kapni. Amennyiben van egy társasjáték, amit már úgy tárolok, hogy kedvel, annak a társasjátéknak a kategóriái alapján fog ajánlásokat kapni. Így amennyiben a stratégiai játékokat kedveli, ebben a részben stratégiai játékok fognak megjelenni nekik.

### 4.7.2 Hozzád hasonlók által kedvelt

Ebben az ajánlásban összegyűjtöm, hogy kik azok a felhasználók, akik velem egy társasjátékot kedvelnek. Amennyiben ez megvan, elkérem ezeknek a felhasználóknak a kedvelt játékait. Így ebben az ajánlásban megeshet, hogy míg én csak stratégiai játékokat néztem, mégis kaland társasjáték ajánlásokat fogok kapni.

### 4.7.3 Próbálj ki valami újat

Ebben az ajánlásban olyan társasjátékok lesznek megtalálhatóak, amik hozzám szorosan nem kapcsolhatóak, így gyakorlatilag az egész adatbázisból válogathatunk.

### 4.7.4 Látogasd meg újra

A látogasd meg újra fülön eltárolom, hogy a felhasználó milyen társasjátékok iránt érdeklődött direktben és feldobom őket újra, ezzel is ösztönözve a társasjáték megrendelését.

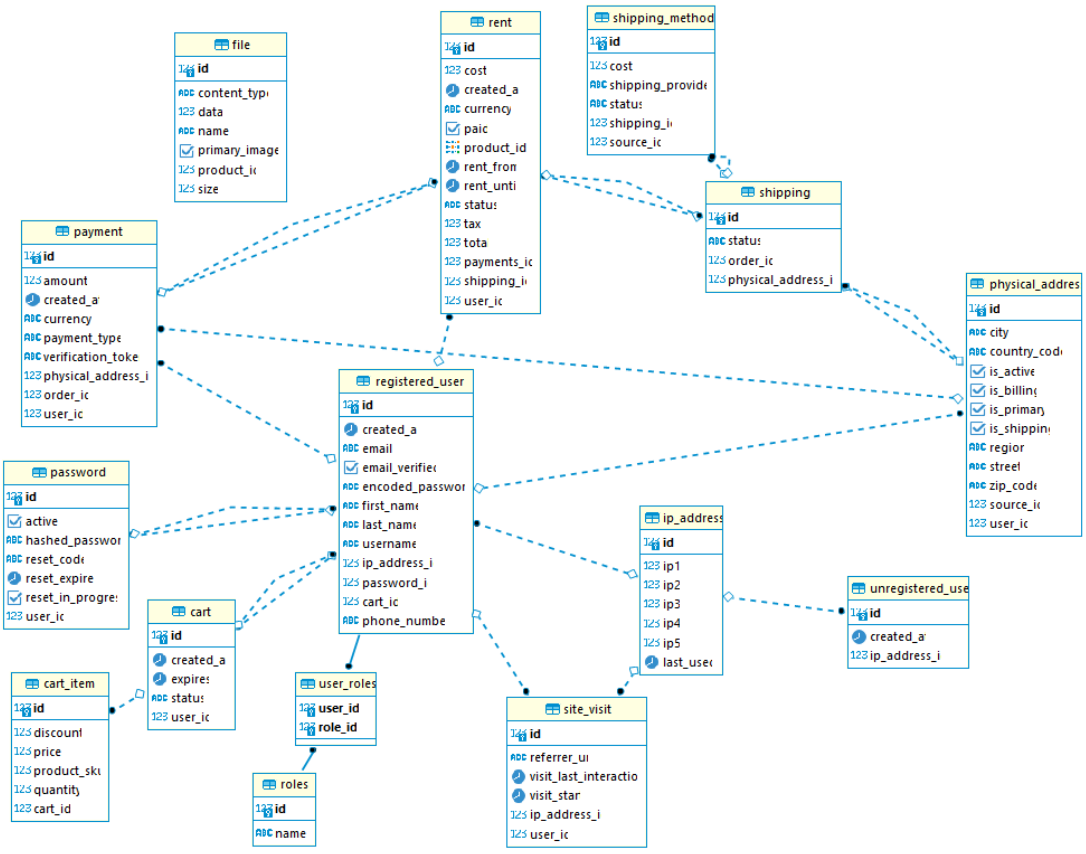
# 5. MEGVALÓSÍTÁS

## 5.1 Fejlesztés

Az applikáció megvalósítása a tervezési fázis után kezdődhetett el. Mivel a technológiák és eszközök előre meg voltak határozva számomra ezért ezek beszerzésével és telepítésével kezdtem.

A tényleges fejlesztést az alsóbb rétegen kezdtem. Itt egy Spring Boot applikációt hoztam létre, aminek elkészítése az egyik legegyszerűbb, hiszen léteznek erre kifejezett eszközök, amik legenerálják nekünk az alapbeállításokat. Én is egy ilyen eszközt használtam, a neve spring initializr.

Miután a programom le tudott futtatni egy Hello World! példa programot elkezdhettem mélyebben a fejlesztést. Az első dolgom az adatbázis létrehozása volt, amit a tervezési részben már nagyobb részben megterveztem. A PostgreSQL és ArangoDB applikációhoz csatolása után elkészítettem a modelleket és a hozzájuk tartozó repository-kat. A végleges adatbázis modell olyan adatokat is tartalmaz, amik előzetes tervezésnél nem merültek fel számomra. A modellt az 5.2 ábrán szemléltetem.

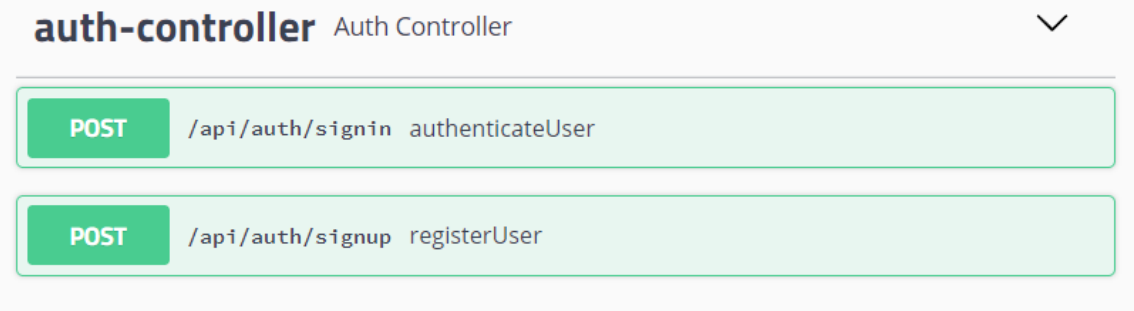



*5.1 ábra adatbázis modell*

A modell elkészítése után Controllereket hoztam létre, amik Rest API-n keresztül tudnak kommunikálni a külvilággal. A controllerek elsődleges célja az adatbázis adatokkal való feltöltése volt, később pedig a weboldal kéréseinek kiszolgálása. Swagger dokumentáció importálása után látványos API dokumentációt kaptam. Így a következő végpontokat hoztam létre

### 5.1.1 Hitelesítés

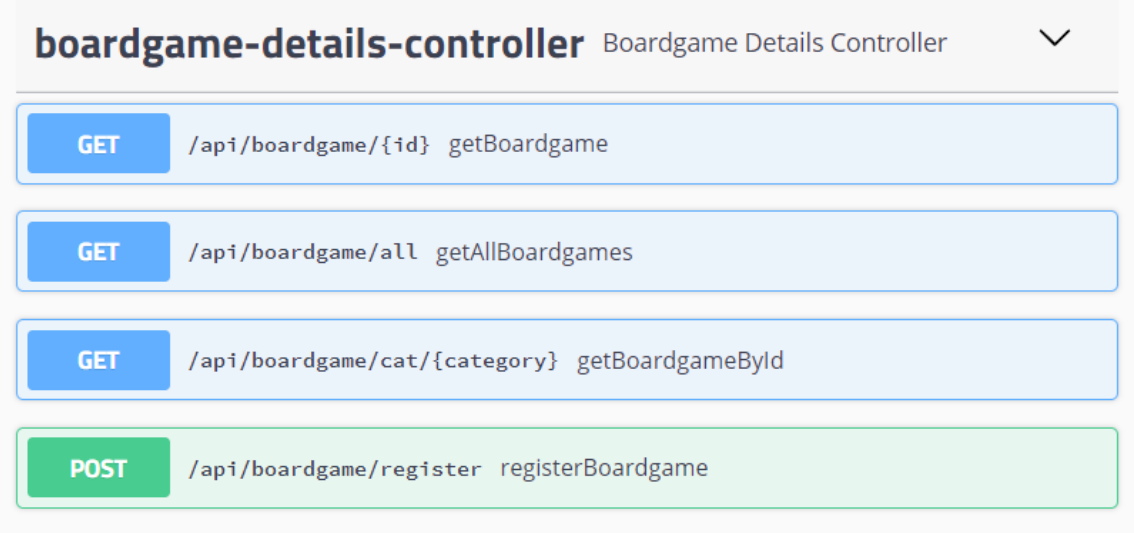
A felhasználói hitelesítés 2 API-ból tevődik össze, ami az 5.2 ábrán található. Ez a controller kezeli a felhasználók regisztrációját és bejelentkezését.




*5.2 ábra felhasználói hitelesítés végpontok*

### 5.1.2 Társasjáték részletek és feltöltés

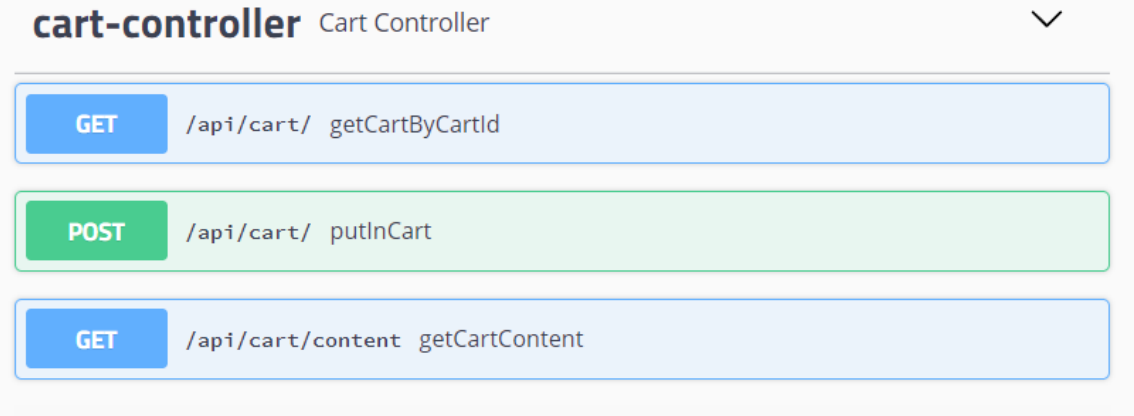
Az egyik fő controller ami a társasjátékok lekérését, beregisztrálását (feltöltését) és kategóriák szerinti lekérését támogatja. A végpontokat az 5.3 ábrán szemléltetem.




*5.3 ábra társasjáték részletek végpontok*

### 5.1.3 Kosár kezelés

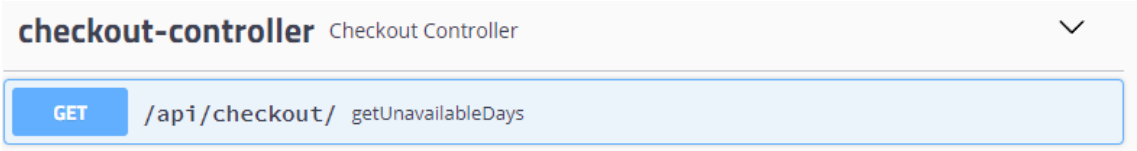
Ebben a controllerben a kosár és hozzá tartozó végpontokat kezelem. A három API a felhasználóhoz tartozó kosár, a kosár ID alapján való lekérése és a kosárba tevést támogatja. A végpontokat az 5.4 ábrán szemléltetem.




*5.4 ábra kosár kezelés végpontok*

### 5.1.4 Pénztár

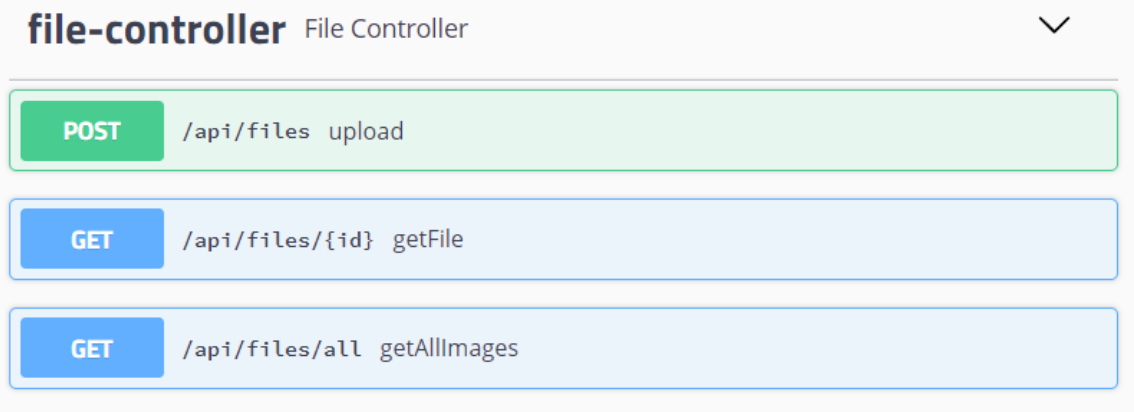
A pénztár és rendelés megkezdéséhez szükséges adatokat kérem le ebben a controllerben. A végpont jelenleg a társasjátékokhoz tartozó nem elérhető napokkal tér vissza. Ezzel tudom megjeleníteni a pénztár második pontjában lévő naptárban az elérhetetlen napokat. Az API-t az 5.5 ábrán szemléltetem.




*5.5 ábra elérhetetlen napok végpontja*

### 5.1.5 File management

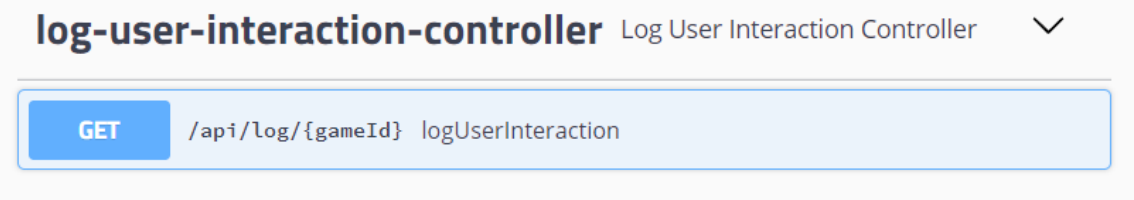
Ahogy a nevéből is kikövetkeztethető itt a file-ok és azok kezelésével foglalkozom. A programom jelenleg nem tartalmaz sok file-t önállóan, így ez inkább később lesz jobban kihasználva. Az 5.6-os ábrán található 3 végpont közül jelenleg a getAllImages van aktív használatban.




*5.6 ábra file kezelés végpontok*

### 5.1.6 Felhasználói interakciók

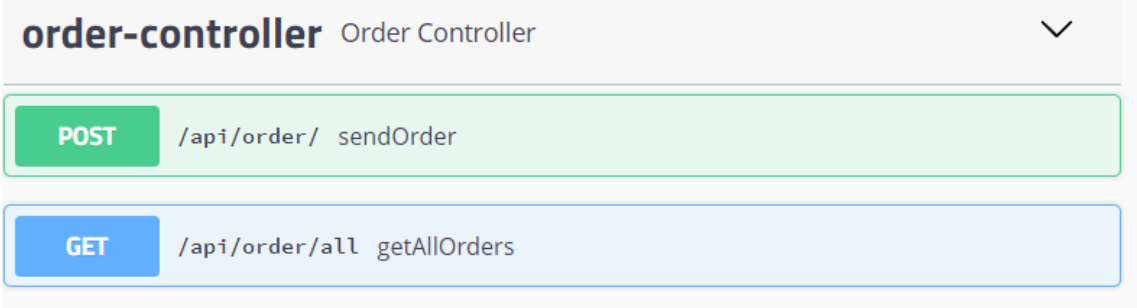
A felhasználó interakciók controllerben a felhasználó kattintásait mentem el. Ez gyakorlatban annyit jelent, hogy ha rányom egy társasjátékra akkor a gráf adatbázisomban létrehozom köztük a szükséges élt. A végpontokat az 5.7 ábrán szemléltetem.




*5.7 ábra felhasználó interakciók logolása végpont*

### 5.1.7 Rendelés vezérlés

A rendelések kezelését két API végzi, amiket az 5.8 ábrán szemléltetek. A két API a rendelés leadását, illetve a rendelések lekérését tudja kezelni.




*5.8 ábra rendelés kezelése végpontok*

### 5.1.8 Ajánlások

Az ajánlások Controller az ajánló rendszert szolgálja ki megfelelő adatokkal. Minden egyes ajánló rendszerhez külön API tartozik, ami a megfelelő társasjátékokkal tér vissza. Az végpontokat az 5.9 ábra mutatja.

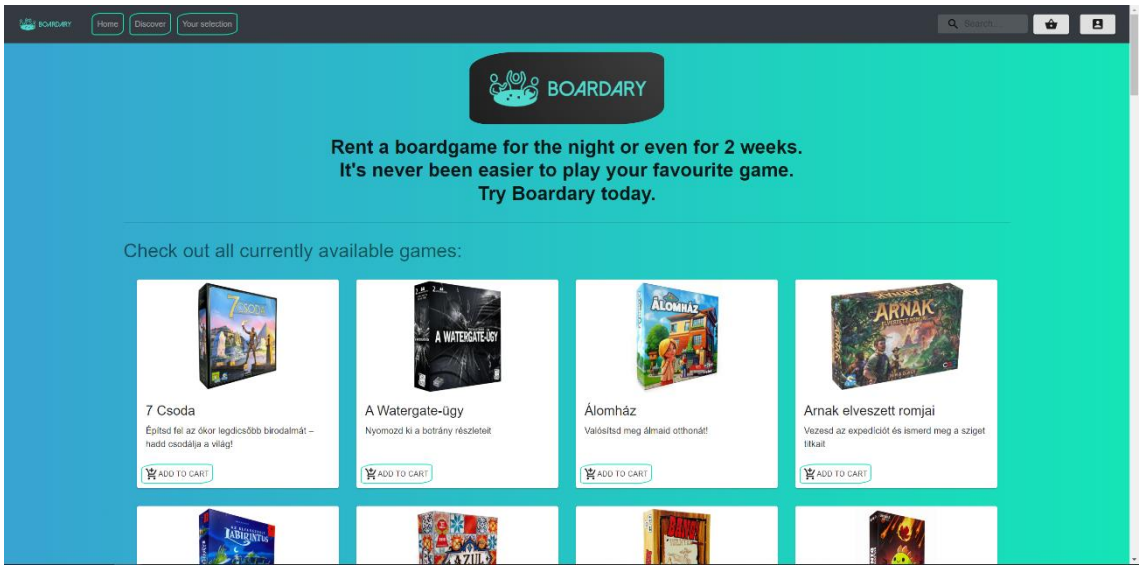

*5.9 ábra ajánlások kezelése végpontok*

## 5.2 Kinézet

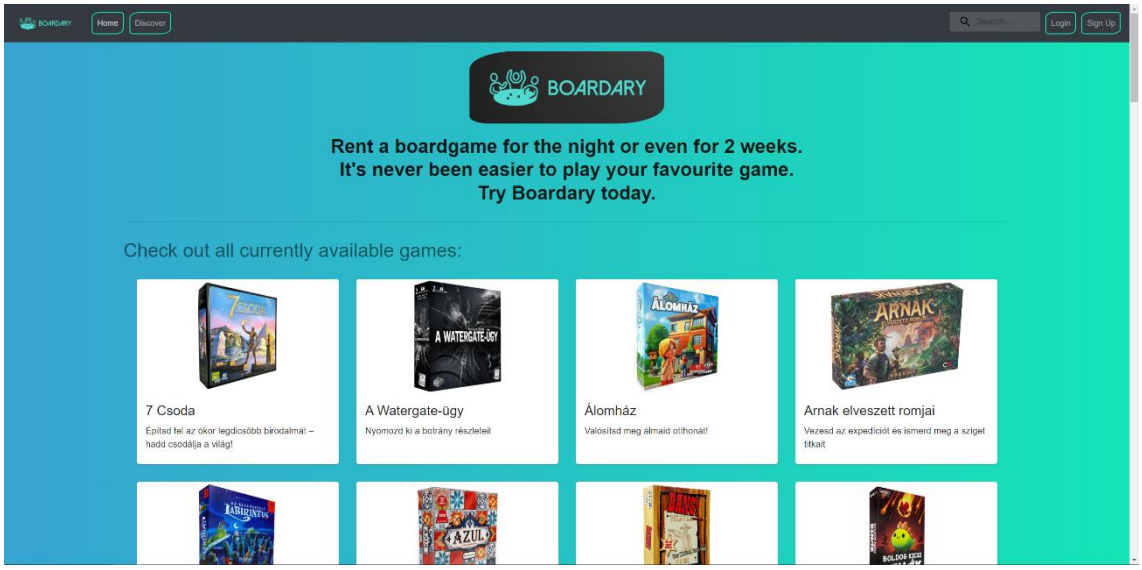
Az applikációm megjelenési részét próbáltam minél modernebbre átláthatóbbra és használhatóbbra készíteni. A Weboldal a tervezésben lerajzolt tervekhez képest enyhén változott ezeket a változásokat a későbbiekben fogom szemléltetni. A programom ReactJS használatával készült. A komponenseim nagy része material-ui használatával készült, ami egy Google által kifejlesztett csomag, különböző megjelenésekre.

### 5.2.1 Főoldal

A Főoldal letisztult megjelenést kapott, ahol a társasjátékok listázva vannak. Ezeket a társasjátékokat mind a bejelentkezett mind a látogató felhasználók is láthatják. Az 5.10 és 5.11 ábrán jól látható, hogy a különbség köztük, hogy míg egy regisztrált felhasználó, már a kosarába rakhatja a társasjátékot ezt egy látogató nem tudja megtenni, hiányzik neki a gomb.




*5.10 ábra főoldal bejelentkezett felhasználóval*




*5.11 ábra főoldal kijelentkezett felhasználóval*

### 5.2.2 Fejléc

A fejléc három féle formát tud felvenni attól függően, hogy be van-e jelentkezve a felhasználó, illetve, hogy a bejelentkezett felhasználó rendelkezik-e admin jogokkal. Az admin jogokkal rendelkező fejléc az 5.12 míg az ennek hiányában lévő felhasználó fejléce az 5.13 ábrán látható. A be nem jelentkezett felhasználó fejléce abban különbözik, hogy számára a személyes ajánlások sem jelennek meg. Ez az 5.14 ábrán látható.

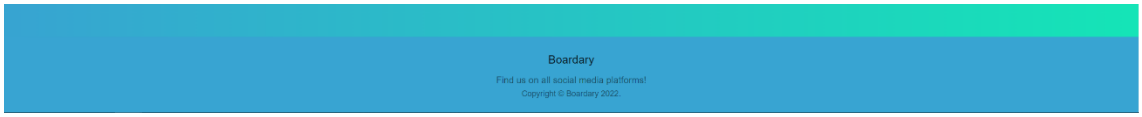

*5.12 ábra admin jogokkal rendelkező felhasználó fejléce*


*5.13 ábra admin jogokkal nem rendelkező felhasználó fejléce*


*5.14 ábra be nem jelentkezett felhasználó fejléce*

### 5.2.3 Lábléc

A weboldal lábléccel van ellátva, ami minden oldalon látható. A lábléc változtatható, igény esetén közösségi média gombokkal, különböző navigátorokkal ellátható. A lábléc az 5.15 ábrán látható.

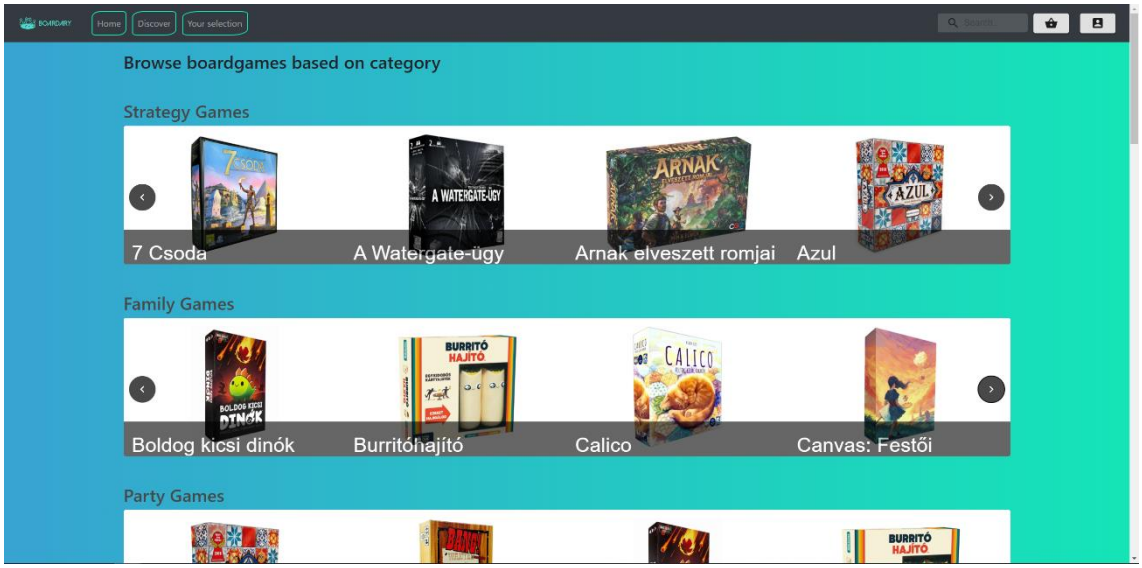



*5.15 ábra lábléc*

### 5.2.4 Felfedezés

A felfedezés fül szintén egy közös oldal, ami minden felhasználónak ugyan azt a tartalmat mutatja. A fülön kategóriákra bontva lehet böngészni a játékok között. A kategóriák sorrendben: Stratégia, Családi, Parti, Kártya, Nevetés, Páros, Egyéni, Fantázia, Kaland játékok. Ezt a fület minden felhasználó eléri, akár bejelentkezett akár nem. Az 5.16 ábrán látható, hogy jelenleg egy bejelentkezett felhasználóval böngészem.

A kategóriákban, több játék is megtalálható ezeket a jobb és baloldalt is megtalálható navigátorokkal lehet léptetni.




*5.16 ábra felfedezés oldal*

### 5.2.5 Társasjáték részletek

További közös fül minden felhasználónak, a társasjáték saját leírása és adatai oldal. Az oldalon a játékok bérlésének részletes leírása olvasható, hogy milyen kategóriákba tartozik, illetve egy rövid leírás róla. Ezeket az adatok az 5.17 ábrán mutatom be. Ahogy az 5.18 ábrán is látszik, amennyiben elérhető, egy YouTube játékbemutató is be van ágyazva a leírásba.

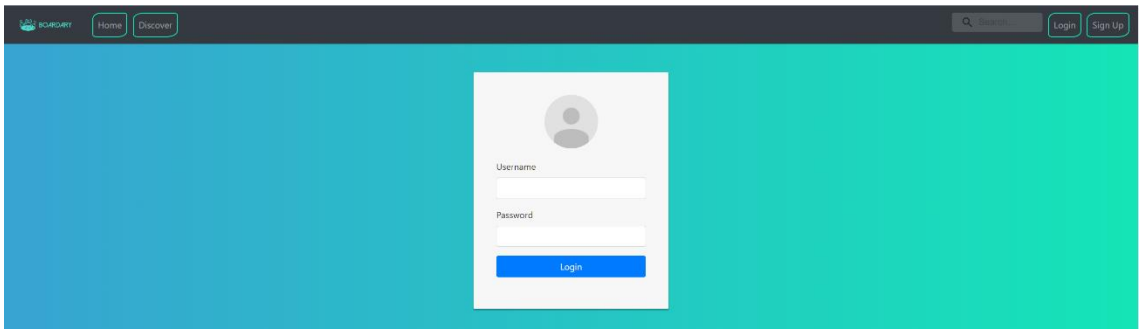

*5.17 ábra társasjáték részletei oldal*


*5.18 ábra társasjáték részletei leírás és beágyazott bemutató videó*

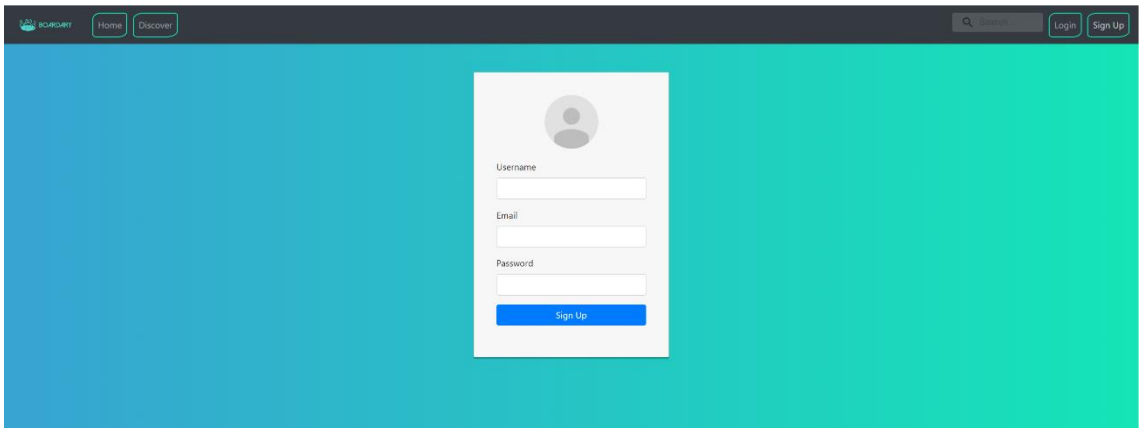
### 5.2.6 Regisztráció, bejelentkezés

Amennyiben a felhasználó regisztrálni szeretne ezt nagyon egyszerűen megteheti. Egy email, egy felhasználónév és egy jelszó szükséges hozzá. Próbáltam minél egyszerűbb regisztrációs folyamatot létrehozni, ezzel is ösztönözve a felhasználók regisztrációs kedvét. A regisztrációs űrlap az 5.10 ábrán látható.

Amennyiben a felhasználónak már van fiókja be tud jelentkezni amire egy külön felület áll rendelkezésre. Ez a felület az 5.19 ábrán tekinthető meg. Rossz bejelentkezési adatok megadása esetén a felhasználó erről egy figyelmeztetés láthat.



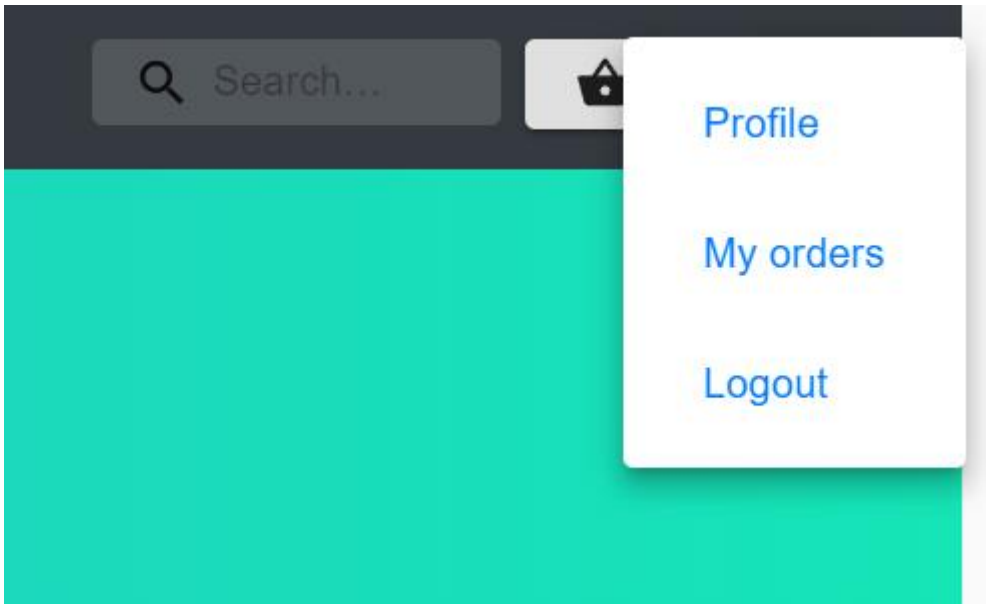

*5.19 ábra bejelentkezés űrlap*




*5.20 ábra regisztrációs űrlap*

### 5.2.7 Profil fül

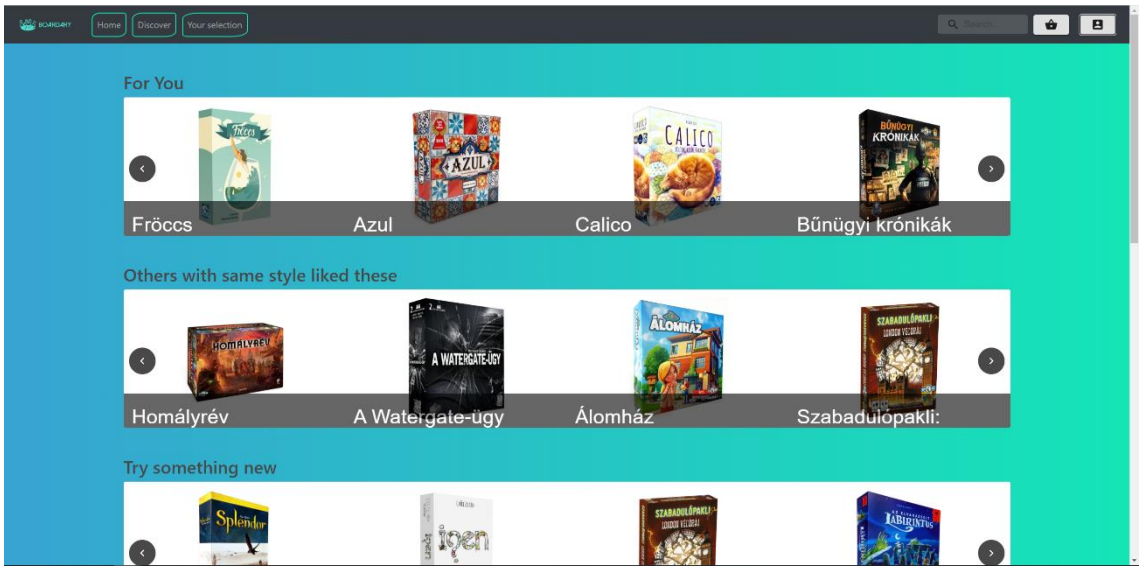
A bejelentkezett felhasználók a jobb felső sarokban lenyithatják a profil fület, amiben 3 opció közül választhatnak. A saját megrendeléseiket nézhetik meg, a saját profiljukat és kijelentkezhetnek. A lenyíló fület az 5.21 ábrán mutatom be.




*5.21 ábra profil lenyíló fül*

### 5.2.8 Személyre szóló ajánlások

A személyre szóló ajánlások oldal, nagyon hasonló felépítésű, mint a felfedezés oldal. Itt a 4 személyre szabott ajánlás rendszert láthatják a felhasználók. A felületet az 5.22 ábrán szemléltetem. Az ajánlások logikáját az 5.4-es fejezetben fejtem ki bővebben.

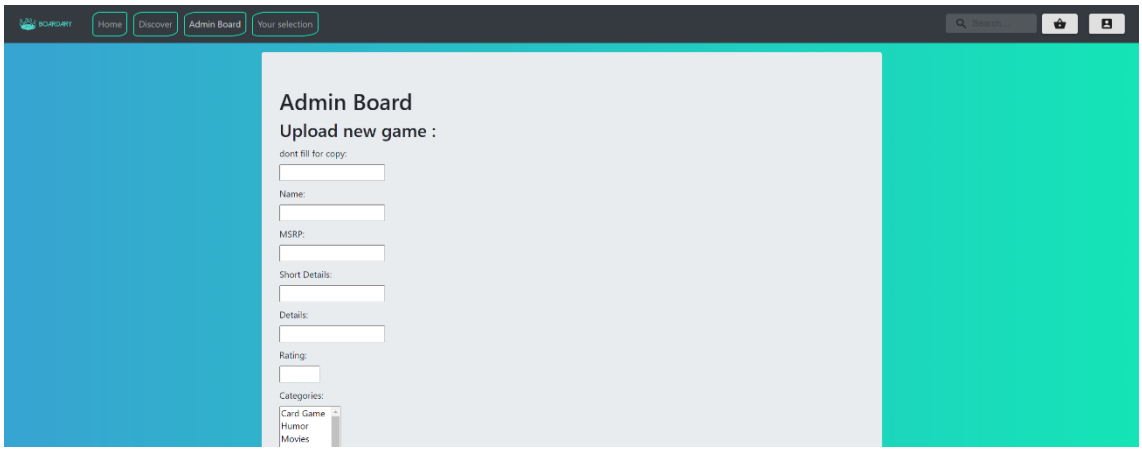



*5.22 ábra ajánlások oldal*

### 5.2.9 Admin oldal

Az admin az egyetlen oldal, ahol a kinézet nem volt számomra mérvadó, ahogy az az 5.23-as ábrán is látszik. A játék feltöltésnél minden olyan infót meg kell adni, ami a

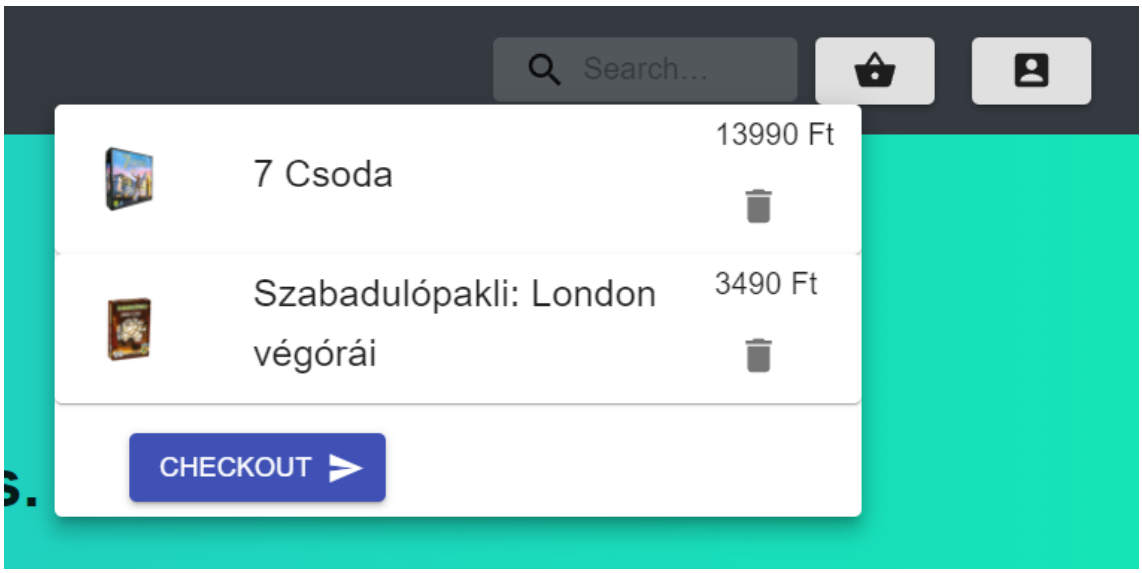
játékok listázásához és leírásához szükséges. Itt több képet is lehetőségünk van feltölteni 1-1 társasjátékhoz.




*5.23 ábra admin oldal*

## 5.3 Rendelés

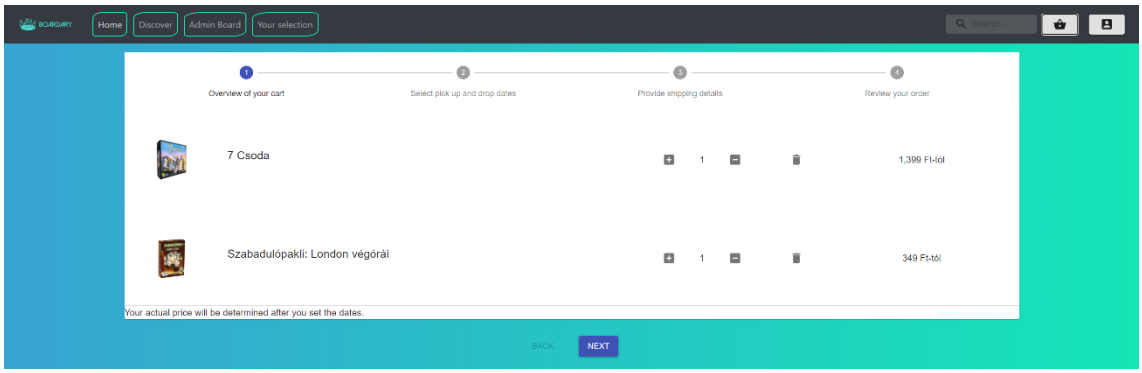
Rendelést csak bejelentkezett felhasználó tud leadni. A rendelés menete, egy társasjáték kosárba való behelyezésével kezdődik. A kosár tartalmának megjelenítése az 5.24 ábrán látható.




*5.24 ábra kosár fül*

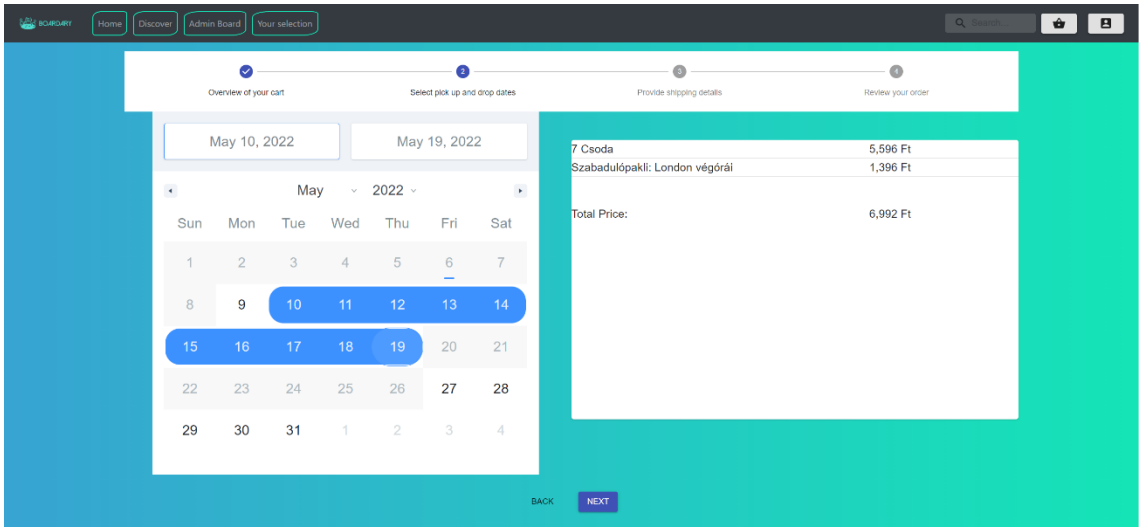
Ez után a Checkout gombra kattintva érheti el a pénztár első lépését. A Pénztár 4 lépésből áll, amiről egy fejléc ad információt.

Az első lépés a kosár átnézése, ahol a kosárba tett társasjátékokat nézheti át a felhasználó. Egy társasjátékot kitörölhet, illetve akár többet is rakhat be belőle a kosarába. Ezt az 5.25 ábrán szemléltetem.



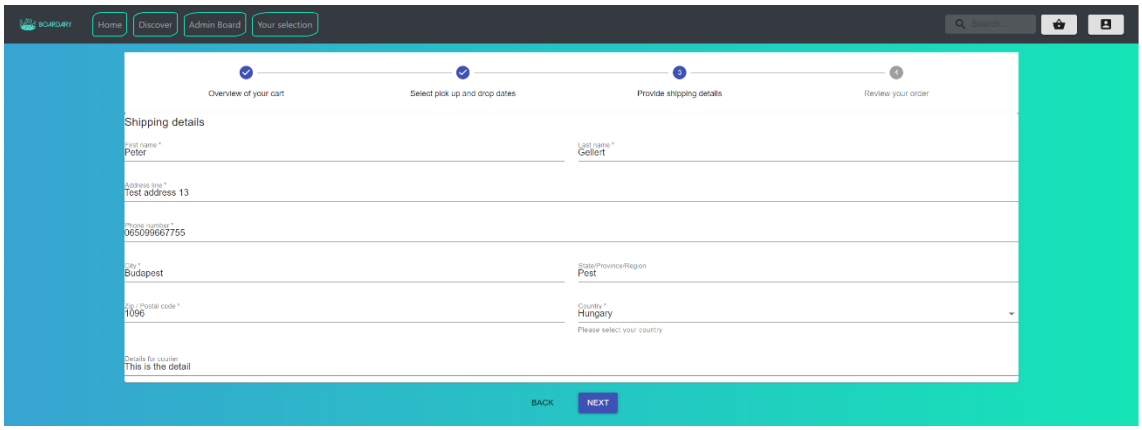

*5.25 ábra kosár tartalma*

A második lépésben tudja a felhasználó a számára szimpatikus időt kiválasztani. Ahogy az 5.26-os ábrán is látszik, az időpontok korlátozottak, csak az elérhető időpontok közül tud választani a felhasználó. Az időpont kiválasztásával a termékek ára is változik. Az árazás a termék leírás oldalán található részletesebben. A leghosszabb lehetséges bérlési idő 2 hét, azaz 14 nap. Amennyiben nem jelöl ki dátumot, automatikusan az aznapi napra lesz a megrendelés beállítva.



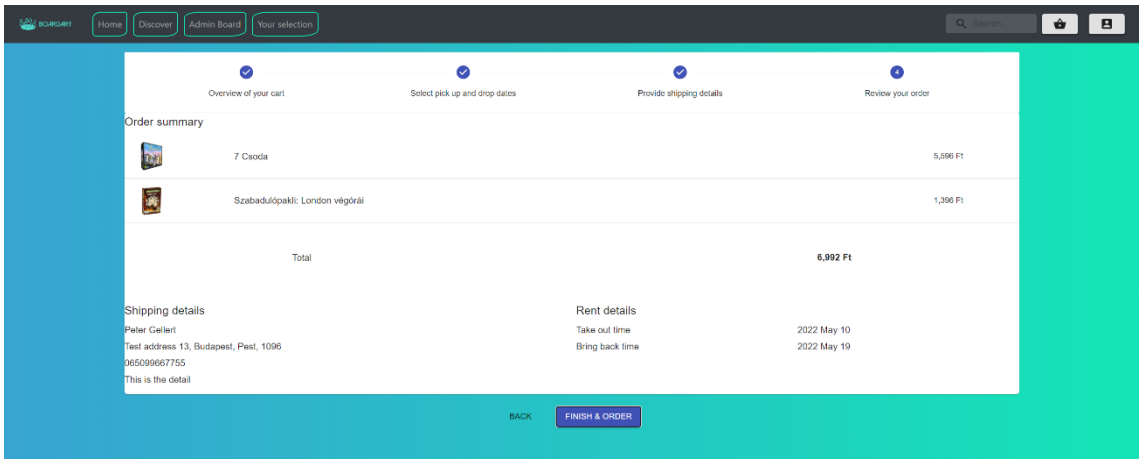

*5.26 ábra dátum választó*

A következő lépésben adhatja meg a szállítási információkat a felhasználó. Amennyiben már volt rendelése, ez az oldal számára kitöltődik a már leadott rendelés szállítási információi alapján. Egy ilyen oldalt láthatunk az 5.27-es ábrán. Amennyiben a felhasználó helytelen formátumú adatokat ad meg, a rendszer nem engedi az utolsó oldalra.

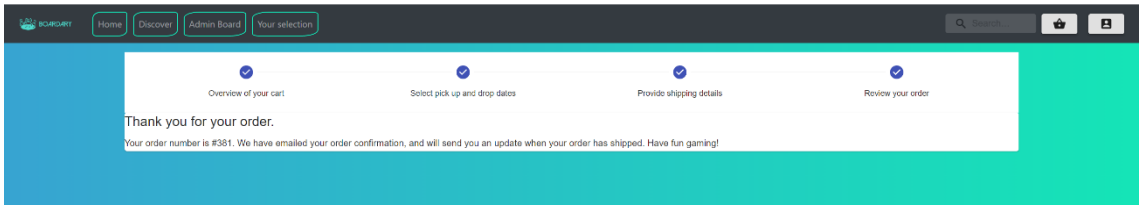



*5.27 ábra szállítási adatok*

Az utolsó lépés a rendelés átnézése és leadása. A fizetés jelenleg egy mock API-t hív, amivel a felhasználónak nincs interakciója. Az 5.28-asábrán látható átnézés során a felhasználó bármikor visszaléphet és változtathat az adatokon, társasjátékokon. A rendelés leadása gombra kattintva a felhasználó egy visszaigazolást láthat, ami tartalmazza a rendelés számát amire később hivatkozhat. Ez az 5.29 ábrán található.




*5.28 ábra rendelés összegzése*




*5.29 ábra rendelés visszaigazolása*

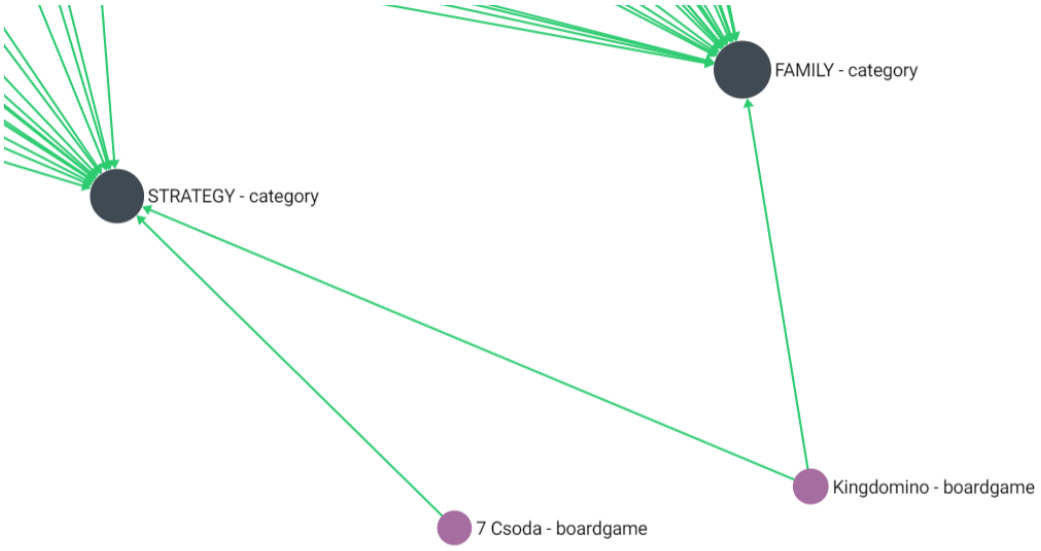
Rendelés leadása után a felhasználó a saját rendelések fülön megtekintheti a rendelése állapotát, a fizetés státuszát, illetve szállítással kapcsolatos információkat szerezhet. A táblázatban a legfontosabb információk találhatóak meg. Az oldal az 5.30-as ábrán tekinthető meg.


*5.30 ábra rendelések átnézése*

## 5.4 Ajánlások

A személyre szóló ajánlásokat gráf adatbázis segítségével számítom ki. A négy ajánlás rendszer mindegyike különböző algoritmussal számolja ki a számunkra megfelelő játékokat.

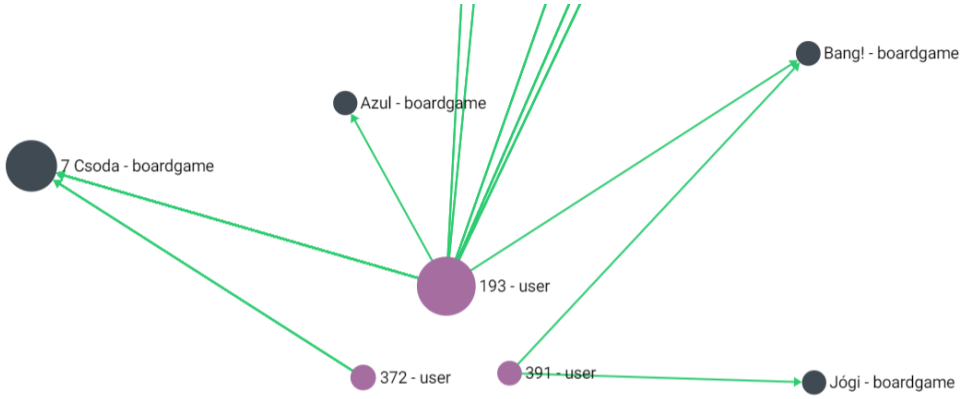
A játékok létrehozásuk (admin felület regisztráció) idejében egy társasjáték_kategória éllel kapcsolódnak a hozzájuk csatolható kategóriákhoz, ahogy az 5.30-es ábra is mutatja.




*5.31 ábra társasjáték_kategória él*

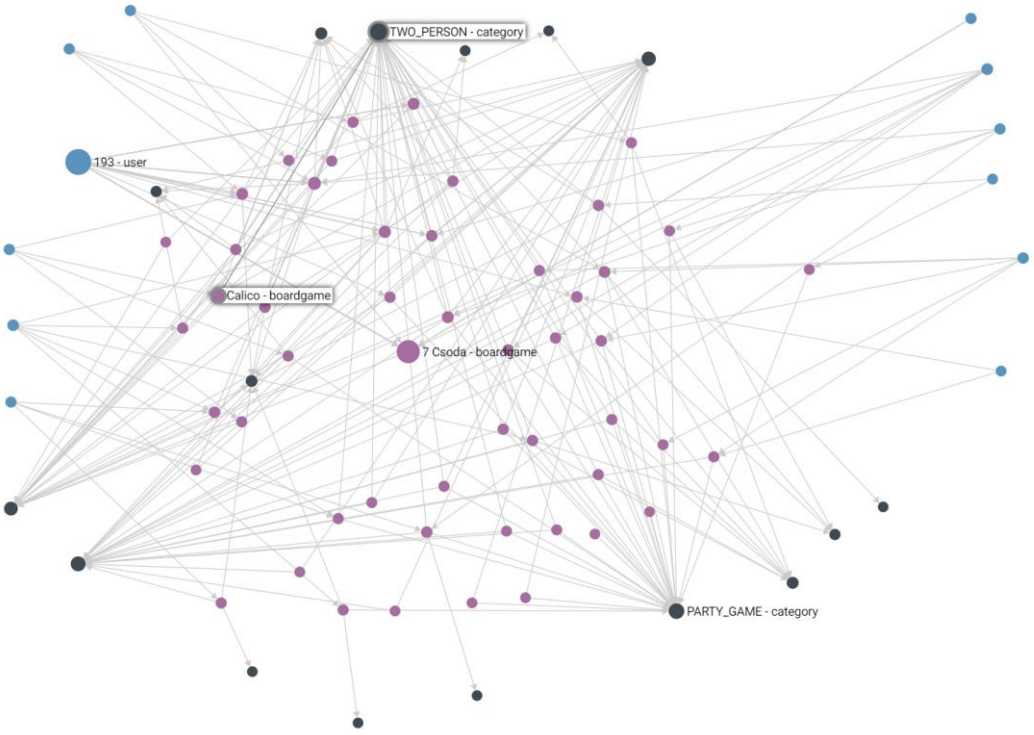
Amikor a felhasználó rákattint egy társasjátékra ez elmentődik egy felhasználó_társasjáték él képében az adatbázisban. Ez az él a társasjáték és a felhasználó

között jön létre így teljes képet kapunk arról, hogy egy felhasználó közvetlen milyen társasjátékokat kedvel. A létrejött élek segítségével később ki tudom számolni a felhasználó és a társasjátékok közötti kapcsolatokat. A létrejött élek az 5.32 ábrán láthatók.




*5.32 ábra felhasználó_társasjáték él*

Miután ezzel megvagyunk, gyakorlatilag egy teljes gráfot fel tudunk rajzolni. Itt a kapcsolatok összeadódnak és egyben látszódnak. Az 5.33-as ábrán láthatjuk, hogy néz ki 12 felhasználónak az interakciója 50 társasjátékkal.




*5.33 ábra felhasználó-játék-kategória gráf*

### 5.4.1 Neked ajánlott

A neked ajánlott társasjátékok esetén a *felhasználó_társasjáték* élekből indulunk ki és ezeknek a társasjátékoknak megkeressük a kategóriáit. Miután ezeket a kategóriákat megtaláltuk a kategóriához tartozó társasjátékokat keressük meg és listázzuk ki a felhasználónak. Így a felhasználó, ha stratégiai játékokat kedvel ebben a részben főleg stratégiai játékokkal fog találkozni.

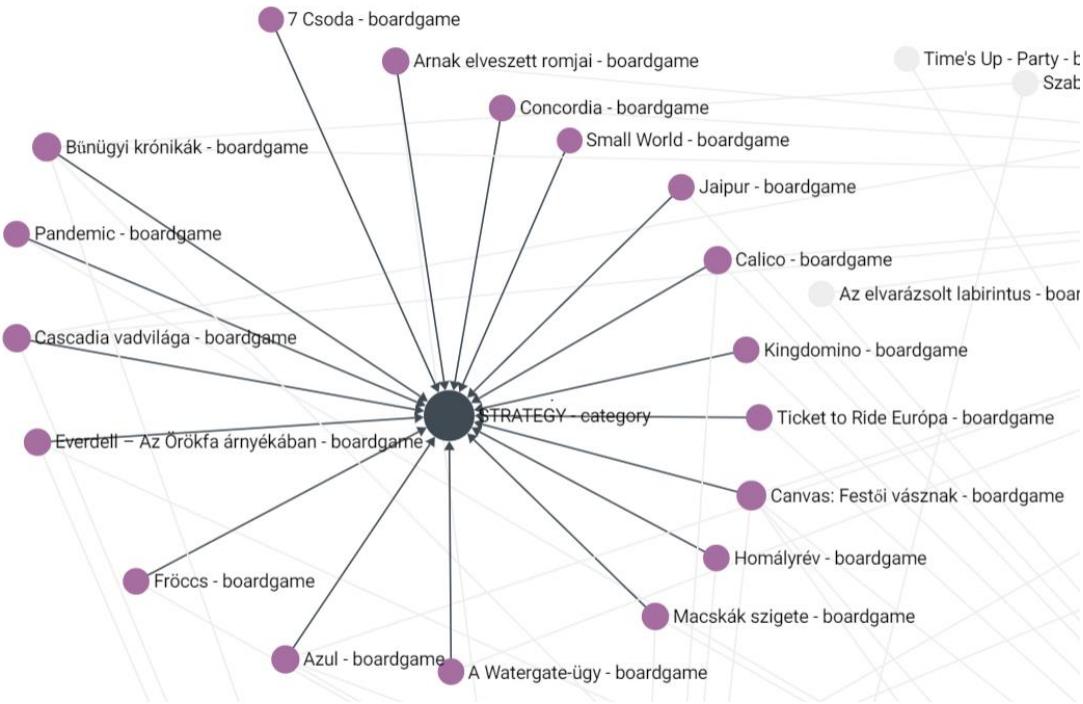
Az ajánlás kiszámítását az 5.34-es ábrán látható query hívásával oldottam meg. Mivel a gráf ebben az esetben már adott volt, így nem volt más dolgom, mint gráf műveletek elvégezni.

```
@Query("FOR v IN 3 ANY @userNodeId GRAPH 'user_boardgame_category' " +
        "OPTIONS {uniqueVertices: \"global\", order: \"bfs\"} RETURN v")
List<BoardgameNode> forYouRecommendation(@Param("userNodeId") String userNodeId);
```

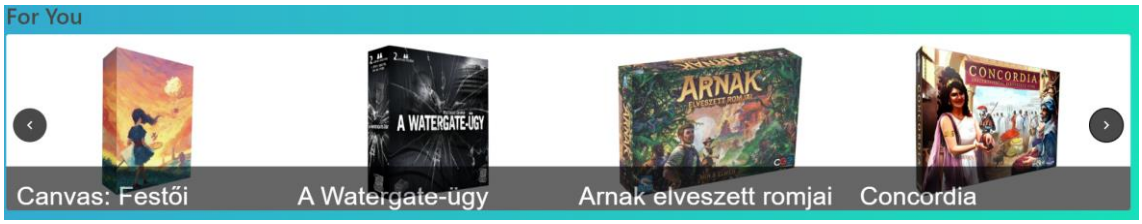
*5.34 ábra neked ajánlott query*

A query szélességi bejárással elkezdi megnézni, hogy a számomra kedvelt társasjátékhoz tartozó kategóriákhoz, milyen társasjátékok tartoznak.

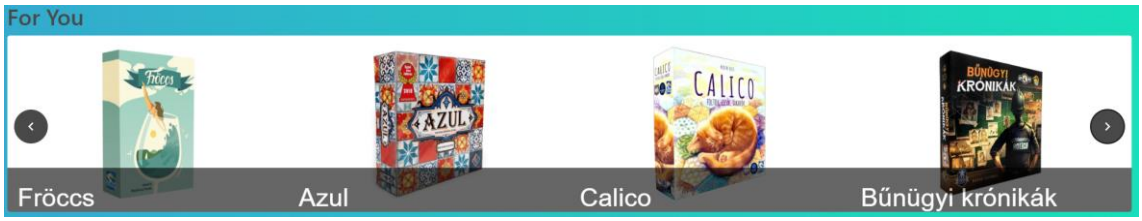
Egy példán bemutatva, friss felhasználóval, a *7 Csoda* társasjátékra való kattintással, az ajánlott társasjátékok kizárólag stratégiai játékok lesznek. A Stratégia éleit az 5.35-ős ábrán szemléltetem. A megjelent társasjátékok pedig az 5.36,5.37 és 5.38-as ábrán láthatók. Jól látni, hogy csak a Stratégia játékok közül kaptam ajánlásokat.



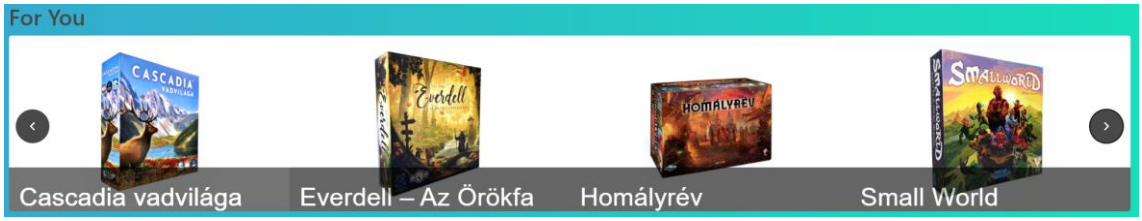

*5.35 ábra Stratégia kategória kapcsolatai*




*5.36 ábra nekem ajánlott első oldal*




*5.37 ábra nekem ajánlott második oldal*




*5.38 ábra nekem ajánlott harmadik oldal*

### 5.4.2 Hozzád hasonlók által kedvelt

A hozzád hasonlók által kedvelt ajánlás a *felhasználó_társasjáték* élen kezdi a keresést. Itt megnézzük, hogy a felhasználó milyen társasjátékokat kedvel ezután kikeresem, hogy azt a társasjátékot mely más felhasználók kedvelik. Miután ezek a felhasználók megvannak az ők *felhasználó_társasjáték* éleiket nézem és ezen társasjátékokat tudom megmutatni az eredeti felhasználómnak. Ezzel, ha valaki mondjuk kedvel egy *7 Csoda* társasjátékot, lehetséges, hogy ebben a részben robbanó macskákat fog kapni, mellette egy Homályrévvel. A megoldáshoz írt query az 5.39-es ábrán látható.

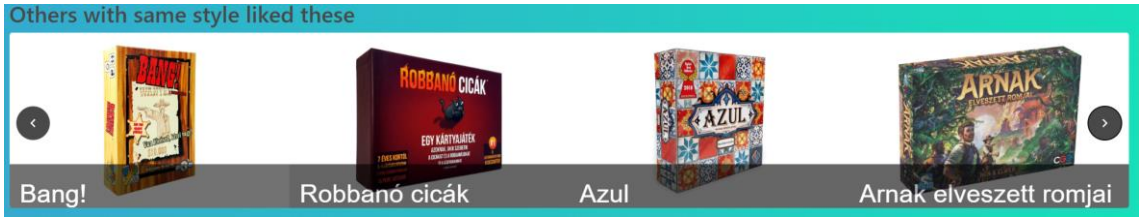
```
@Query("FOR v IN 3 ANY @userNodeId GRAPH 'user_boardgame_category' " +
        "OPTIONS {uniqueVertices: \"global\", order: \"bfs\", edgeCollections: 'user_boardgame_edge'} RETURN v")
List<BoardgameNode> othersLikedRecommendation(@Param("userNodeId") String userNodeId);
```

*5.39 ábra hozzád hasonlók által kedvelt játékok query*

A query működésében nagyon hasonló az előző megoldáshoz azonban itt fontos volt, hogy csakis a *felhasználó_társasjáték* éleken dolgozzon. Így kaphatjuk meg a 3 messze lévő csomópontokat a gráfban, amik a számunkra elvárt eredményt adják.

Például:

A felhasználó a *7 Csoda* társasjátékra nyomott, az ajánlásban a többi felhasználó által kedvelt játékok is megtalálhatóak, mint például a kártyajátékok, mint ezt az 5.40-es ábrán is láthatjuk.




*5.40 ábra hozzám hasonlók által kedvelt ajánlások*

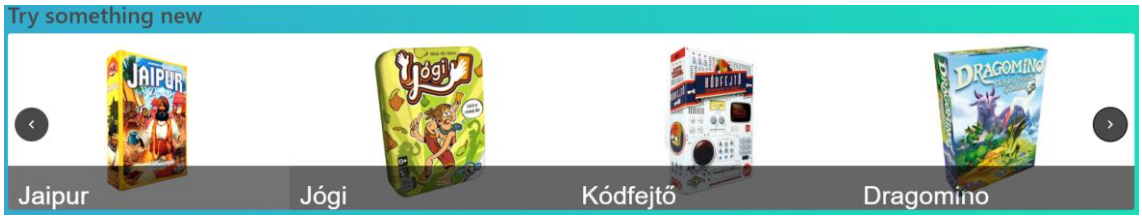
### 5.4.3 Próbálj ki valami újat

A próbálj ki valami újat egy teljesen különálló része az ajánlórendszernek. Itt a felhasználó közvetlen kapcsolatait nem vesszük figyelembe, gyakorlatilag egy random rendezés alapján elkérjük a társasjátékokat és azoknak egy részét adjuk vissza. A query az 5.41-es ábrán látható.

```
@Query("FOR v IN boardgame SORT RAND() LIMIT 6 RETURN v")
List<BoardgameNode> tryNewRecommendation();
```

*5.41 ábra próbálj ki valami újat query*

A felületen és az 5.42 ábrán jól látható, hogy ha a felhasználó továbbra is csak stratégiai játékokra kíváncsi, itt ettől függetlenül véletlenszerűen is kaphat társasjáték ajánlásokat.




*5.42 ábra új játékok ajánlása*

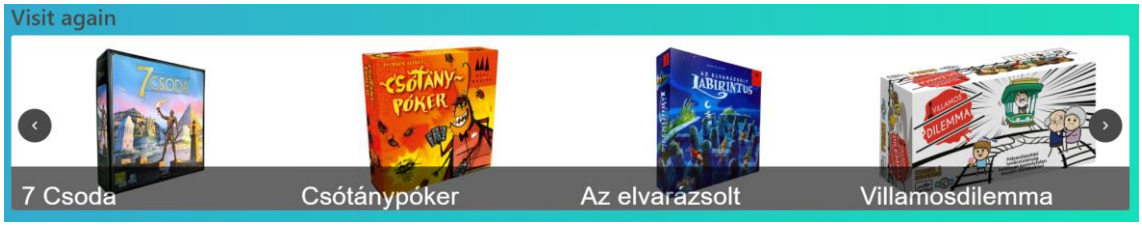
### 5.4.4 Látogasd meg újra

A látogasd meg újra részben ismét a *felhasználó_társasjáték* éleket vesszük figyelembe. Itt a felhasználótól 1 távolságra lévő csomópontok, amik nekem számítanak. Ami változás és fontos, hogy itt csak a kimenő éleket nézzük, így a bejövő éleket nem veszem figyelembe. A query az 5.43 ábrán látható.

```
@Query("FOR v IN 1 OUTBOUND @userNodeId GRAPH 'user_boardgame_category' " +
        "OPTIONS {uniqueVertices: \"global\", order: \"bfs\"} RETURN v")
List<BoardgameNode> visitAgainRecommendation(@Param("userNodeId") String userNodeId);
```

*5.43 ábra látogasd meg újra query*

Egy példán könnyen lehet szemléltetni, ahol a felhasználóval rányomtam 3 másik társasjátékra a *7 Csodán* kívül. Ezeket a játékokat most már a felületen is látjuk. Az 5.44-es ábrán szemléltetem.




*5.44 ábra látogasd meg újra ajánlások*

# 6. TESZTELÉS

A programom unit és integrációs tesztekkel lett lefedve, amivel az alapvető működést teszteltem le. A bizniszlogikára unit teszteket írtam míg a controllertől-adatbázisig tartó teszteket integrációs tesztekkel oldottam meg. Ezen felül, egy funkcionális tesztelést is eszközöltem, aminek a segítségével a weboldal felületéről tudtam az alapvető hibákat kiküszöbölni.

## 6.1 Tesztelési terv

A funkcionális teszteket egy táblázatban gyűjtöttem össze, hogy könnyebb legyen később követni és visszatérni rá. A táblázat a 6.1-es ábrán látható.

| Típus | Elvárt működés |
|---|---|
| Funkcionális | Nem bejelentkezett felhasználó nem látja a your selection fület, sem a profil és kosár ikont |
| Funkcionális | Nem bejelentkezett felhasználó nem látja a főoldalon a kosárba rakás funkciót |
| Funkcionális | A discover fülön lévő játékok a hozzájuk tartozó kategória fülben vannak listázva |
| Funkcionális | Üres kosár 'empty shopping bag' felirattal jelenik meg |
| Funkcionális | Rossz jelszó beírása esetén hiba bejelentkezéskor |
| Funkcionális | Túl rövid felhasználónév beírásakor hiba a regisztrációkor |
| Funkcionális | Helytelen email beírásakor hiba regisztrációkor |
| Funkcionális | Túl rövid jelszó beírásakor hiba regisztrációkor |
| Funkcionális | Létező email regisztrációjakor hiba |
| Funkcionális | Nem bejelentkezett felhasználó ne lássa a következő oldalakat: My orders, Profile, Your selection, Checkout |
| Funkcionális | Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Visit again |

| | |
|---|---|
| Funkcionális | Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: For You |
| Funkcionális | Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Others with same style liked these |
| Funkcionális | Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Try something new |
| Funkcionális | Admin boardot csak admin joggal rendelkező user nyithassa meg |
| Funkcionális | A leadott rendelés megtalálható a rendeléseim fül alatt, a megfelelő társasjátékokkal, státuszban és ID-val |

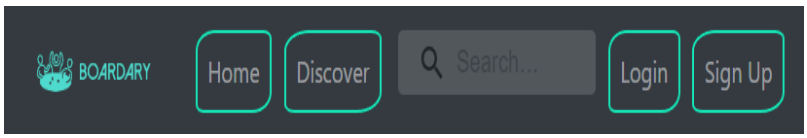
*6.1 táblázat Funkcionális tesztesetek*

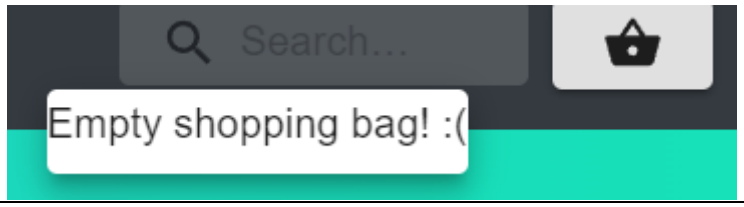
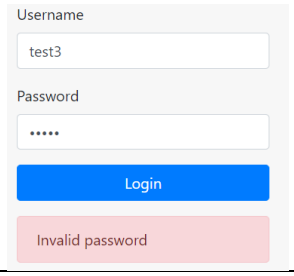
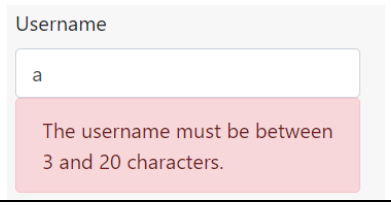
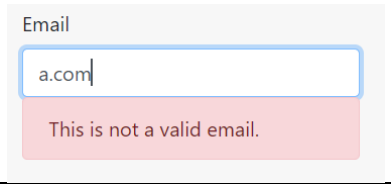
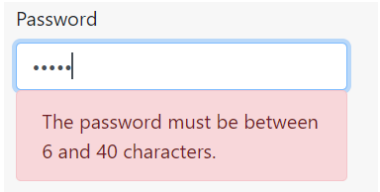
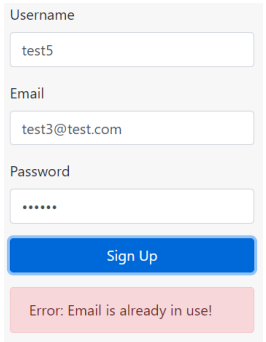
## 6.2 Teszt futások

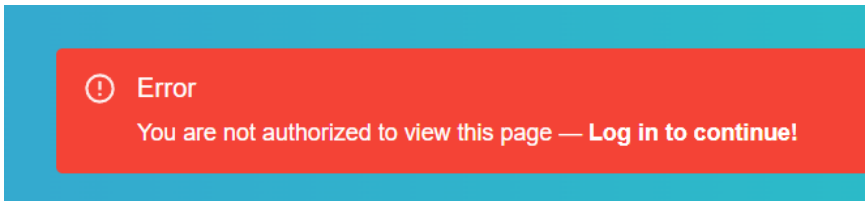
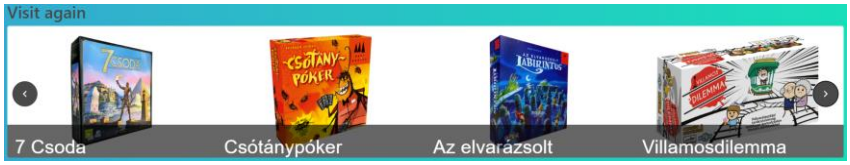
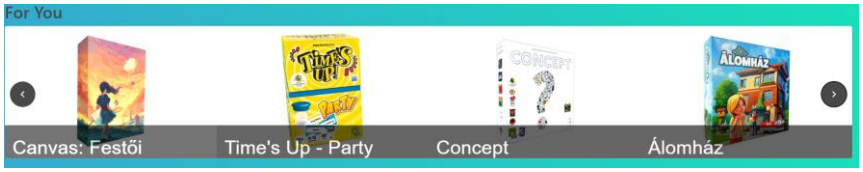
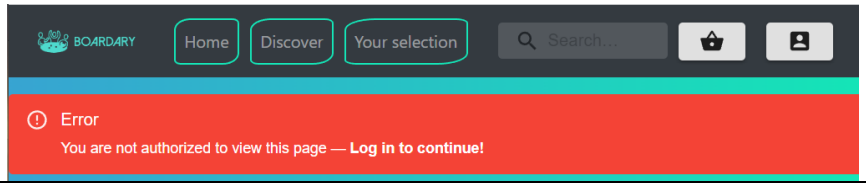
Az integrációs és unit teszteket az IntelliJ fejlesztési környezetemben futtattam le. A funkcionális tesztjeimet a felületen való kattintgatással teszteltem le. Az eredmények alapján, javítottam a programomat.

## 6.3 Teszt eredmények

A teszt eredményeit egy táblázatba gyűjtöttem, ahol az elvárt működést képekkel igazoltam. A 6.2-es táblázatban jól leolvashatóak az elvárások és eredmények.

| Elvárt működés | Eredmény |
|---|---|
| Nem bejelentkezett felhasználó nem látja a your selection fület, sem a profil és kosár ikont |  |
| Nem bejelentkezett felhasználó nem látja a főoldalon a kosárba rakás funkciót |  |

| | |
|---|---|
| A discover fülön lévő játékok a hozzájuk tartozó kategória fülben vannak listázva |  |
| Üres kosár 'empty shopping bag' felirattal jelenik meg |  |
| Rossz jelszó beírása esetén hiba bejelentkezéskor |  |
| Túl rövid felhasználónév beírásakor hiba a regisztrációkor |  |
| Helytelen email beírásakor hiba regisztrációkor |  |
| Túl rövid jelszó beírásakor hiba regisztrációkor |  |
| Létező email regisztrációjakor hiba |  |

| | |
|---|---|
| Nem bejelentkezett felhasználó ne lássa a következő oldalakat: My orders, Profile, Your selection, Checkout |  |
| Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Visit again |  |
| Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: For You |  |
| Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Others with same style liked these |  |
| Bejelentkezett felhasználó ajánlási rendszere megfelelően működik: Try something new |  |
| Admin boardot csak admin joggal |  |

| | |
|---|---|
| rendelkező user nyithatja meg | |
| A leadott rendelés megtalálható a rendeléseim fül alatt, a megfelelő társasjátékokkal, státuszban és ID-val |  |

*6.2 táblázat funkcionális tesztelés futtatása*

A tesztelés során megállapítottam, hogy a programom a biztonsági feltételeknek megfelelt és a szükséges funkciók megfelelően működtek.

# 7. TOVÁBBFEJLESZTÉSI LEHETŐSÉG

## 7.1 Felhasználó felület továbbfejlesztése

A felhasználó felület néhány helyen még hiányosnak mondható. A felhasználó egy letisztult oldalt láthat maga előtt azonban ez néha már túl kevés. A későbbiekben szeretnék a weboldal oldalában egy sávot, ahol akár hirdetések akár ajánlások futnának. A felfedezés oldalon szeretném, hogy a későbbiekben lehessen kategóriák szerint szűrni.

A pénztár szolgáltatásnál a legtöbb funkció szép és letisztult azonban az időpont foglalás melletti sávot szeretném szebbé tenni azzal, hogy mellé teszek plusz információkat, képeket a társasjátékról.

## 7.2 Funkciók

A főbb funkciók működnek azonban találhatóak hiányosságok a programban, amik a felhasználóknak kedveznének, javítanák a felhasználói élményt. Az egyik ilyen funkció a Pagination, azaz, hogy az oldal ne egybe töltsön be, hanem részletekben. Szerintem egy végtelen görgős megoldás lenne a későbbiekben erre a legmegfelelőbb.

Amennyiben a felhasználó nem elégedett a termékkel, illetve azt idő előtt vissza szeretné hozni, az oldalra felkerülhetne egy visszatérítés, visszaküldés funkció. Ezzel a funkcióval biztosíthatnám a felhasználók elégedettségét, nem mellesleg jogszabályi kötelezettségemnek is így tennék eleget.

A fizetési szolgáltatás, fizetős mivoltából, jelenleg nincs integrálva az applikációba. Ennek hiánya egy rendelésképtelen felületet eredményez. Amint az oldalból szeretnék egy éles rendszert beállítani ezt mindenképp bele kell integrálnom.

Amennyiben egy jól működő webshopot szeretnék üzemelni, elengedhetetlen az email kampányok használata. Ezt később szeretném személyre szabott ajánlatokkal kiküldeni, hogy minden felhasználó a számára tökéletes és kedvező ajánlatokat láthassa.

Az admin felület jelenleg nem ad sok információt arról, hogy a webshop, hogy üzemel. Ide könnyen fel tudnék venni egy olyan oldalt, ahol a leggyakrabban megrendelt társasjátékokat, a legnagyobb kattintást szerző játékokat és ehhez hasonlókat láthatnék. Ezzel befolyásolhatnám a felületen megjelenő játékok sorrendjét.

## 7.3 Tartalom

Az oldalon jelenleg egy nagyobb magyar kiadónak a webshopjából lemásolt adatok találhatók. Fontos, hogy több kiadóval együtt tudjak dolgozni, és minden társasjáték elérhető legyen a polcaimon, így a későbbiekben minél több helyről szeretném az adataim beszerezni.

# 8. ÖSSZEFOGLALÁS

A társasjátékok az utóbbi években ugrásszerű növekedésen vannak túl. A pandémia alatt az eladások megugrottak. Ez a tény és a társasjátékok iránti szeretetem indította el bennem a motivációt, hogy egy társasjáték kölcsönző weboldalt hozzak létre.

Az irodalomkutatás ebben a témában egyáltalán nem magától értetődő. Manapság egyre több és több technológiával találkozik az ember. Ezek a technológiák pedig néha jók, néha nem válnak be. Számomra az arany középút volt a megoldás, választottam olyan technológiákat, amiket már ismertem, illetve választottam olyanokat, amik számomra is kihívást fog jelenteni használatuk. Ezzel a kombinációval értem el, hogy a szakdolgozatom alatt rengeteget tanultam.

A weboldal fiatalos designt kapott, élénk színeket és könnyű kezelhetőség jellemzi. Mivel fejlesztéséhez ReactJS-t használtam így könnyen bővíthető lesz a későbbiekben, ezen felül a logikai, alsóbb részek külön találhatók. Ezáltal egy sokkal agilisabb terméket kaptam. A felhasználók gyorsan tudnak regisztrálni, és a regisztrációt követően azonnal már perceken belül rendelni is tudnak. Az interaktív felület végig vezeti őket a szükséges lépéseken ezáltal a rendelés rövid idő alatt lezárható. A felületen lehetőséget biztosítottam a felhasználóknak a rendeléseik nyomon követésére is, hogy bármikor visszajöhessen és azt ellenőrizhessék.

Az ajánló rendszerem arangoDB adatbázison alapul, ahol egy gráf struktúrába gyűjtöm a felhasználókról az adatokat. Látható volt, hogy 10 felhasználóval és 50 társasjátékkal is már több száz él jött létre a csomópontjaim között. Ez a hatalmas adattömeg ad lehetőséget arra, hogy az ajánlásokat pontosítsam és minél pontosabb ajánlásokat tudjak nyújtani a felhasználóknak. A négy féle különböző ajánlási rendszerem együttesen egy remek alapot adnak arra, hogy a weboldalból egy komoly ajánlási rendszert is létre tudjak hozni. A felhasználók a számukra közeli társasjátékokat láthatják, ezen kívül, a felhasználók láthatnak a komfort zónájukból kieső társasjátékokat is. Egy ilyen rendszer használata nagyban megnövelheti a rendelések számát egy weboldalon.

További előnye a gyűjtött adataimnak, hogy később email kampányokat is tudok majd indítani, teljesen testreszabott tartalommal, ami a mai világban egy kihagyhatatlan marketingfogás.

Az alsóbb részek Java és Spring fejlesztése során olyan dolgokat tapasztalhattam és tanulhattam, amikre a kezdetekben nem gondoltam. Felnyitotta a szemem abban, hogy bármennyire is egyszerűnek tűnik egy program, a tervezésére nem szabad az időt spórolni, ugyanis későbbi fejlesztés során, ez hatalmas mennyiségű munkától tud megszabadítani.

A későbbiekben folytatni szeretném a weboldalam fejlesztését. Vannak fejleszteni való területek, és amint úgy érzem, az ötlettel piacra lehet lépni ezt meg fogom tenni. A piac, eddigi tapasztalataim alapján nyitott lenne egy ilyen lehetőségre.

# 9. SUMMARY

Board games have been going up in sales in the last few years. During the pandemic most of us realized the importance, and the enjoyment of playing together with our friends. This fact and my love towards board games gave me the motivation to create a board game renting web application.

While writing the literature research I quickly realized that not everything is black and white in this field. There are thousands of different technologies, each with its own benefit. For me the golden mean was the way to go. I realized that the importance of learning new things is as important as doing something I was already familiar with. With this combination I reached my goal to learn as much as I can.

My website has a youthful, modern and clean design with lively colors. The main goal was to make it as usable as possible and I believe I managed to reach that goal. For the front-end part I used ReactJS. This will result in an application that can be easily extended with features in the future. Completely separated there's my Java and Spring back-end that can also be a functional application and can be reused for multiple applications in the future. With this combination I reached a much more agile product. Users can register in just minutes and right after that, they can also rent a board game. With the interactive user interface the user can click through the checkout process very easily and fast. The interface provides access to other features too for instance the *my orders* page where one can check the status of their orders.

My recommendation system is based on arangoDB, a modern and handy solution for noSQL data storage. It provides document storage as well as graph storage. In my case the data required for the recommendation system was stored in a graph storage. With only 10 users and 50 board games the available data was huge, and this made me realize the potential of this system. In the future this recommendation system can be used for more advanced recommendations, for instance an emailing service where every user can get personalized, targeted emails. For me, other than the increase in rentings, it was also important that the users can get favorable recommendations, and everyone gets the opportunity to try out new board games.

While developing the back-end service I realized the importance of planning and design. Even with prepared planning I had an insane amount of design questions that I was not aware of during the planning phase. These questions and the time I spent answering them were a great eye-opener for me, to never look down on planning ever again, as it is an investment that can have a huge return in the future.

In the future I have plans to continue the work on the website. There are multiple fields that can be improved like quality-of-life improvements. Whenever I feel like the application is ready, I'm planning to release it to the public as what I saw so far, the market shows a huge demand to play more and more board games.

# 10. IRODALOMJEGYZÉK


[1] Angie: The positive benefits of gaming with kids (http://growingupgamers.blogspot.com/2012/02/benefits-of-gaming-with-kids.html), utoljára megtekintve: 2021.12.10.

[2] Emma Bedford: Global board games market value from 2017 to 2023 (https://www.statista.com/statistics/829285/global-board-games-market-value/), utoljára megtekintve: 2021.12.10.

[3] McKinsey & Company: Ordering in: The rapid evolution of food delivery (https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery), utoljára megtekintve: 2021.12.10.

[4] Játszóház project hivatalos weboldala (https://www.jatszohazprojekt.hu/), utoljára megtekintve: 2021.04.18.

[5] Alyssa Walker: SOAP Web Services Tutorial: What is SOAP Protocol? (https://www.guru99.com/soap-simple-object-access-protocol.html), utoljára megtekintve: 2021.12.10.

[6] What is REST API? (https://www.mulesoft.com/resources/api/what-is-rest-api-design), utoljára megtekintve: 2021.04.18.

[7] Kafka hivatalos weboldala (https://kafka.apache.org/), utoljára megtekintve: 2021.12.10.

[8] Gradle hivatalos oldala: Gradle vs Maven comparison (https://gradle.org/maven-vs-gradle), utoljára megtekintve: 2021.10.28.

[9] Ismeretlen szerző: Docker: Bevezetés a containerek világába (https://ithub.hu/blog/post/Docker_bevezetes_a_containerek_vilagaba), utoljára megtekintve: 2021.10.28.

[11] Ida Ozarowska: Kubernetes – mi ez és hogyan kezdjük el használni? (https://flyonthecloud.com/hu/blog/kubernetes-tutorial), utoljára megtekintve: 2021.10.28.

[12] Ammar Khaku: Spring Inversion of Control vs Guice Dependency Injection (https://medium.com/software-ascending/spring-inversion-of-control-vs-guice-dependency-injection-935a408ece89), utoljára megtekintve 2021.10.28.

[13] Farkas Gábor: Amit tudnod kell fejlesztőként, IV. rész: Verziókezelés (https://ithub.hu/blog/post/Amit_tudnod_kell_fejlesztokent_IV_resz_Verziokezeles), utoljára megtekintve: 2021.10.30.

[14] Hillary Nyakundi: What is an IDE in Programming? An IDE Definition for Developers (https://www.freecodecamp.org/news/what-is-an-ide-in-programming-an-ide-definition-for-developers/), utoljára megtekintve: 2021.12.10.

[15] Top Computer Languages (https://statisticstimes.com/tech/top-computer-languages.php), utoljára megtekintve: 2021.12.10.

[16] TechVidvan: Python Advantages and Disadvantages – Step in the right direction (https://techvidvan.com/tutorials/python-advantages-and-disadvantages/), utoljára megtekintve: 2021.12.10.

[17] TechVidvan: Advantages and Disadvantages of Java (https://techvidvan.com/tutorials/pros-and-cons-of-java/), utoljára megtekintve: 2021.12.10.

[18] Appsbee Team: Kotlin – Advantages, Disadvantages & Facts You Should Know (https://www.appsbee.com/blog/kotlin-advantages-disadvantages-facts-you-should-know/), utoljára megtekintve: 2021.12.10.

[19] deborah: C Sharp – Features, Advantages and Disadvanteges (https://urbannaturale.com/c-sharp-features-advantages-and-disadvantages/), utoljára megtekintve: 2021.12.10.

[20] Alex Melnichuk: Pros and Cons of .NET Framework (https://ncube.com/blog/pros-and-cons-of-net-framework), utoljára megtekintve: 2021.12.10.

[21] Voctor Rak: Pros and Cons of Ruby on Rails (https://sloboda-studio.com/blog/pros-and-cons-of-ruby-on-rails/), utoljára megtekintve: 2021.12.10.

[22] Utkarsh Sidana: What are the Advantages and Disadvantages of Angular? (https://www.edureka.co/blog/advantages-and-disadvantages-of-angular/), utoljára megtekintve: 2021.12.10.

[23] Pros and Cons of ReactJS (https://www.javatpoint.com/pros-and-cons-of-react), utoljára megtekintve: 2021.12.10.

[24] The Good and the Bad of Vue.js Framework Programming ( https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/), utoljára megtekintve: 2021.12.10.

[25] Tejas Kaneriya: Advantages & Disadvantages of Node.js: Why to Use Node.js? (https://www.simform.com/blog/nodejs-advantages-disadvantages/), utoljára megtekintve: 2021.12.10.

[26] DataFlair Team: Django Advantages and Disadvantages – Why You Should Choose Django? (https://data-flair.training/blogs/django-advantages-and-disadvantages/), utoljára megtekintve: 2021.12.10.

[27] Jahid Akhtar: Microservices Introduction (Monolithic vs. Microservice Architecture (https://dzone.com/articles/microservices-1-introduction-monolithic-vs-microse), utoljára megtekintve: 2021.12.10.

[28] Wikipédia: ACID (https://hu.wikipedia.org/wiki/ACID), utoljára megtekintve: 2021.12.10.

[29] Vika Klochkova: SQL vs. NoSQL vs. NewSQL: Comparative Advantages and Disadvantages (https://dzone.com/articles/sql-nosql-newsql-comparative-advantages-and-disadvantages), utoljára megtekintve: 2021.12.10.

[30] Relational vs. NoSQL data (https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data), utoljára megtekintve: 2021.12.10.

[31] MongoDB hivatalos oldala: NoSQL vs Relational Databases (https://www.mongodb.com/scale/nosql-vs-relational-databases), utoljára megtekintve: 2021.12.10.

[32] NoSQL-adatbázis – Mi az a NoSQL? ( https://azure.microsoft.com/hu-hu/overview/nosql-database/), utoljára megtekintve: 2021.12.10.

[33] Prior Version(s) of Amazon EC2 Service Level Agreement (https://aws.amazon.com/ec2/sla/historical/), utoljára megtekintve: 2021.09.10.

[34] Renuka Rana: Cloud Uptime Guarantee: How Many Nines Do You Need? (https://www.acecloudhosting.com/blog/cloud-uptime-guarantee-how-many-nines/), utoljára megtekintve: 2021.12.10.

[35] AZ EURÓPAI PARLAMENT ÉS TANÁCS (EU) 2016/679 RENDELETE (https://eur-lex.europa.eu/legal-content/HU/TXT/HTML/?uri=CELEX:32016R0679), utoljára megtekintve: 2021.12.10.