Eötvös Loránd Tudományegyetem

Informatikai Kar

Média- és Oktatásinformatika Tanszék

Azul társasjáték megvalósítása webes technológiákkal

Témavezető

Dr. Horváth Győző

egyetemi docens

Szerző

Hosek Henrietta

Programtervező informatikus BSc.

Budapest, 2022

SZAKDOLGOZAT / DIPLOMAMUNKA

EREDETISÉG NYILATKOZAT

Alulírott **Hosek Henrietta** Neptun-kód: **V09MTN** ezennel kijelentem és aláírásommal megerősítem, hogy az Eötvös Loránd Tudományegyetem Informatikai Karának, **Média- és Oktatásinformatika** Tanszékén írt, **Azul társasjáték megvalósítása webes technológiákkal** című szakdolgozatom/diplomamunkám saját, önálló szellemi termékem; az abban hivatkozott szakirodalom felhasználása a szerzői jogok általános szabályainak megfelelően történt.

Tudomásul veszem, hogy szakdolgozat/diplomamunka esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Budapest, 2022.05.29.

.....
hallgató aláírása

Nyilatkozat nagy terjedelmű dolgozat elhelyezéséről

A dolgozatom alapjául szolgáló program meghaladja a 200 MB méretet ezért a teljes szakdolgozati programot az inf-es levelezőrendszerből elérhető Onedrive-ra töltöttem fel.

Vállalom, hogy a záróvizsga beosztásról kapott értesítést követően legfeljebb 1 napon belül a teljes szakdolgozati programot a bizottság elnökének átadom.

Budapest,

.....
hallgató aláírása

EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:
Név: Hosek Henrietta
Neptun kód: V09MTN

Képzési adatok:
Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)
Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: *Horváth Győző*
munkahelyének neve, tanszéke: ***ELTE-IK, Média- és Oktatásinfromatika Tanszék***
munkahelyének címe: ***1117, Budapest, Pázmány Péter sétány I/C.***
beosztás és iskolai végzettsége: ***egyetemi adjunktus***

A szakdolgozat címe: Azul társasjáték megvalósítása webes technológiákkal

A szakdolgozat témája:
(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

A dolgozatom témája egy webes alkalmazás, ami egy online multiplayer játék az Azul című társasjáték alapján.
Az Azul 2-4 játékos által játszható, absztrakt, átlagosan 30-45 perces játék. A játékosok feladata csempéket gyűjteni, majd azokat beépíteni a saját falukba úgy, hogy ezért minél több pontot kapjanak.
A játék kezdetén a felhasználó készíthet szobát, vagy csatlakozhat egy már meglévőhöz egy kód segítségével.
Miután elindították a játékot, megjelenik a játékosok táblája, ami egy falból és egy csempegyűjtő részből áll, valamint játékosszám által meghatározott számú manufaktúra (karika), ahol a csempékből lehet választani.
A kör elején véletlenszerűen feltöltődnek a manufaktúrák (körök) csempékkel.
A kezdőjátékos választhat először. A csempe elvevés során a választott manufaktúrából a választott színből az összes csempét el kell venni.
Mielőtt egy csempét véglegesen be tudnánk építeni a falunkba, a csempegyűjtő részen kell adott számú darabot összegyűjteni.
Ahhoz, hogy a falunk (ami egy 5x5-ös mátrix) sorába építhessünk be egy csempét, annyi csempét kell összegyűjteni a csempegyűjtő helyen, ahányadik sorba szeretnénk beépíteni. (Tehát a második sorba való beillesztéshez kettő csempe kell.)
A játékosok felváltva választanak a manufaktúrákról (karikákról) és helyezik rögtön el a szerzett csempéket a csempegyűjtő helyen.
Minden kör végén a csempegyűjtőről beépítésre kerülnek a csempék és pontokat is adnak attól függően, hogy mennyi korábban beépített csempe veszi körül.
A játék minimum 5 körből áll, akkor van vége, amikor valamelyik játékos kitöltött egy sort csempékkel. A sorok és oszlopok kitöltésével lehet bónuszpontokat gyűjteni a játék végén.

Budapest, 2021. 12. 15.

Tartalom

1. Bevezetés

1.1 Motivációm

Minidig is nagyon szerettem társasjátékokkal foglalkozni, már régóta ez a legfőbb szabadidős tevékenységem. A barátaim gyakran partnerek ebben, és rendszeresen ülünk össze játszani. Amikor a korlátozások miatt mindenki hazaköltözött a fővárosból, és csak online tudtuk tartani egymással a kapcsolatot, online társasjátékokkal játszottunk egymással. Viszont a számomra legkedvesebb játék, az Azul nem volt elérhető az általunk használt egyik felületen sem, így azzal nem tudtunk játszani, és gyakran hiányoltuk a vele való játékot. Ez adta az ötletét a témámnak, szerettem volna egy saját Azult készíteni, amivel tudunk játszani a barátaimmal.

Az Azul[1] a modern társasjátékozás egyik hatalmas népszerűséggel rendelkező műve. Egy meditatív, absztrakt játék, ahol minden játékosnak egy saját falat kell építenie csempékből, ezzel minél több pontot szerezve, és így megnyerve a játékot. 2018-ban nyerte el a Spiel des Jahres díját, ami a legnagyobb elismerés, amit egy társasjáték kaphat.

1. ábra: Azul társasjáték eredeti verziója (saját példányom)

[1] https://boardgamegeek.com/boardgame/230802/azul

1.2 Az ötlet megszületése után

Nagyon örültem, hogy rátaláltam erre az ötletre, de én korábban csak Asztali alkalmazások megvalósításával foglalkoztam, viszont ezt a témát csak online multiplayer játékként tudtam elképzelni. Ezért a karon felvettem egy webprogramozással kapcsolatos tantárgyat, ahol megismerkedhettem kicsit a JavaScripttel, ami megalapozta a frontend fejlesztési tudásomat. A szakdolgozatom írásával párhuzamosan végeztem a Kliensoldali webprogramozás tantárgyat, ami szintén sok tudást adott, amit fel tudtam használni.

De maga a megvalósítás nem csak ebből áll. Számomra igen érdekes, hogy csupa új technológiával ismerkedhettem meg, és ezek között felhasználtam egy olyat is, ami még annyira új, hogy kísérleti fázisban van. Persze ennek hátránya, hogy fejlesztés közben belefutottam néhány elsőre nem megmagyarázható esetbe, de erről már csak a dokumentációmban lehet olvasni, ugyanis a programot tekintve született megoldás, így az tökéletesen működik.

2. Felhasználói dokumentáció

2.1 Rendszerkövetelmények

A program futtatásához elengedhetetlen egy számítógép, mely frl van szerelve billentyűzettel, egérrel és kijelzője legalább1920x1080 felbontással rendelkezik, illetve fontos a stabil internetkapcsolat. Többféle számítógépen is tesztelve lett, a minimális rendszerkövetelmény:

- Processzor: Intel(R) Core(TM) i3-7020U CPU @2.30GHz
- Memória: 4,0 GB
- GPU:Intel(R) HD Graphics 620
- Méret: 408 MB
- Támogatott böngészők:
  - Google Chrome (verzió 49.0+)
  - Microsoft Edge (verzió 14.0+)

A játék futtatása telefonokon nem támogatott a kijelző mérete miatt.

A játék üzemeltetéséhez szükség van továbbá:

- Windows 10 64 bit-es operációs rendszerre
- Terminálra
- Node.js telepítésére

2.2 Futtatás

2.2.1 Futtatáshoz szükséges előkészületek:

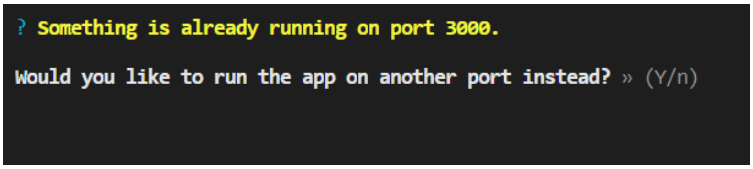
Ha teljesülnek a rendszerkövetelmények, a megfelelő mappába lépést követően az ***npm install*** paranccsal lehet installálni a szükséges könyvtárakat, első használat esetében, majd ***npm start*** paranccsal lehet elindítani a programot.

```
PS C:\Users\USER\Desktop\szakdoga> cd .\azul-game-app\
PS C:\Users\USER\Desktop\szakdoga\azul-game-app> npm start
```

2. ábra: Terminálba beírt parancsok

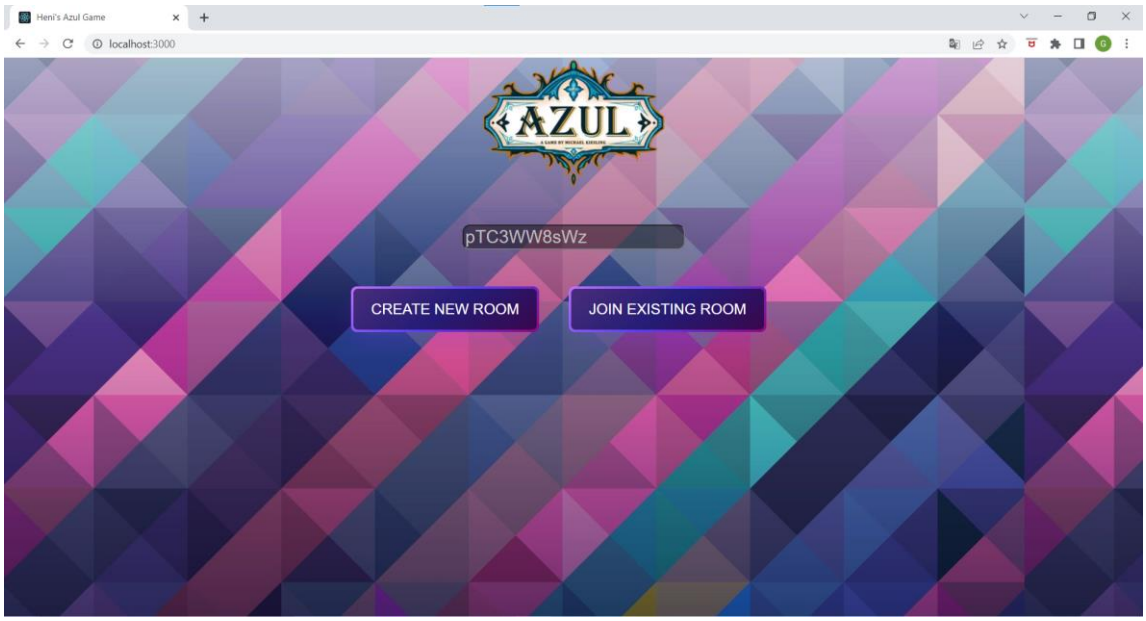
2.2.2 Futtatás

Sikeres indítást követően a játék a böngészőben érhető el a http://localhost:3000/ oldalon. Abban az esetben, ha ez foglalt lenne, a terminálban jelezve van, hogy egy másik portra fog a játék kerülni.



3. ábra: Terminál jelzése, ha más már fut a 3000-es porton

A böngészőben a megfelelő url megnyitása után a kezdőoldal fogadja a felhasználót.

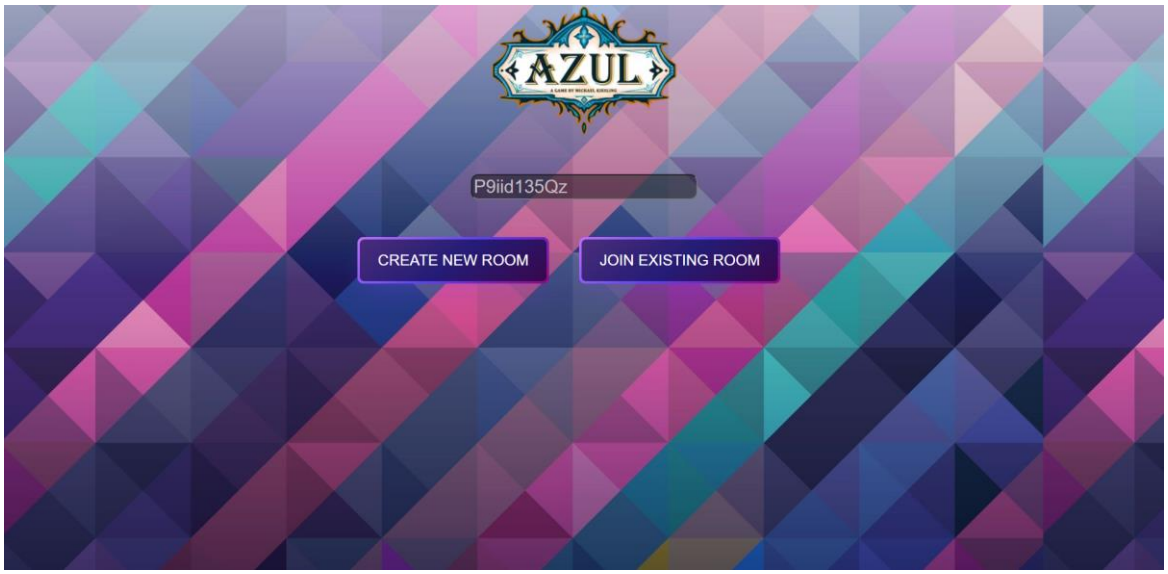


4. ábra: A futtatás után a kezdőlap fogadja a felhasználót

2.3 Alkalmazás működése

2.3.1 Kezdőoldal

A kezdőoldalon látható a játék logója [1], egy input mező, mely automatikusan ki van töltve egy véletlenszerűen generált kóddal. Ez módosítható, a felhasználó akár saját kódot is beírhat. Ebbe a mezőbe kell hogy kerüljön a szoba kódja melyet létre szeretne hozni, vagy amibe csatlakozni szeretne.

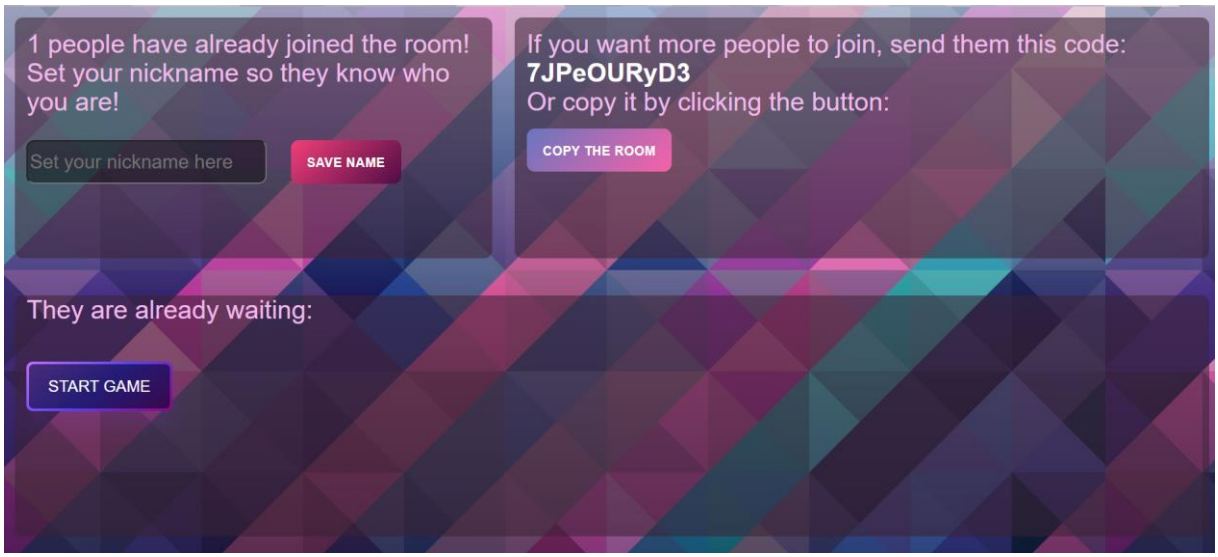


5. ábra: Kezdőoldal kinézete

Megjelenik továbbá két gomb is, a „CREATE NEW ROOM” feliratú gombbal szobát lehet létre hozni, míg a „JOIN EXISTING ROOM” feliratú gombbal lehet csatlakozni egy már meglévő szobához. Fontos, hogy a gombok megnyomása elött megbizonyosodjunk arról, hogy az input mezőben helyes szobakódot adtunk meg, illetve hogy nem hagytuk azt üresen. Ezekben az esetekben felugró ablak jelzi, hogy nem helyesen töltöttük ki az input részt.

A helyes kitöltést követő gombnyomás után megjelenik a várószoba.

2.3.2 Várószoba

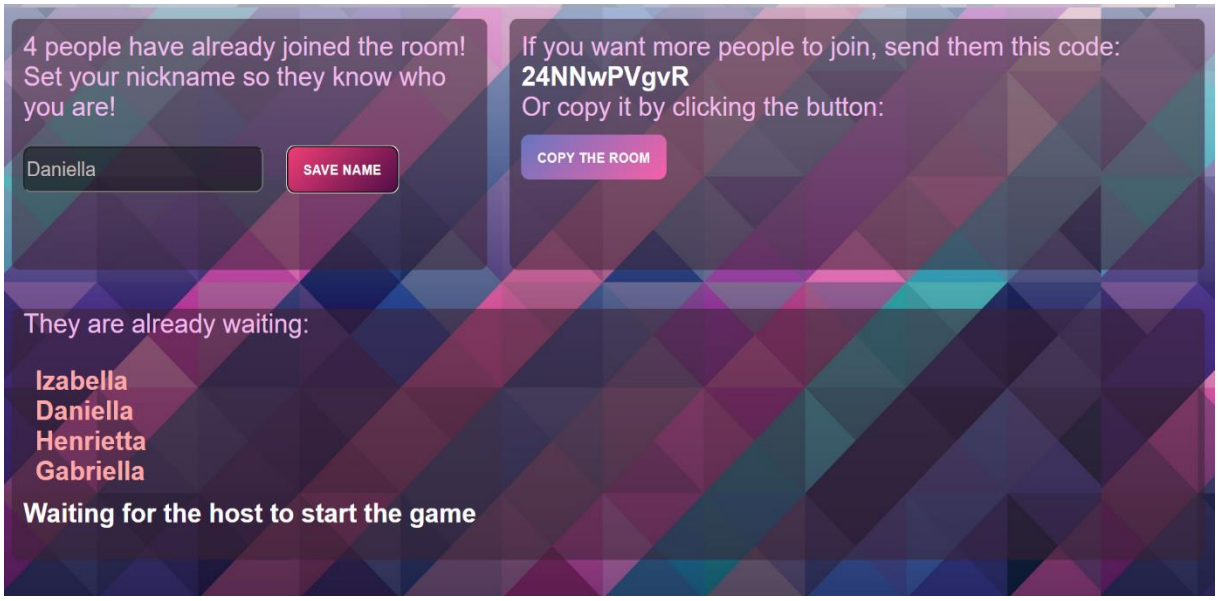


6. ábra: A várószoba közvetlenül a létrehozás után

A várószoba 3 részre bontható, a jobb felső sarokban kaphatnak a szobában lévő felhasználók információt arról, hányan vannak jelenleg a szobában, illetve becenevüket is itt tudják beállítani. Kezdetben senkinek sincs beállítva a beceneve, ezt az input mezőbe beírva majd a „save name” gombre kattintva tehetik meg.

A bal felső sarokban található a szoba kódja, és annak kimásolására alkalmas „copy the room” gomb. Bárki kimásolhatja és elküldheti másoknak, hogy csatlakozni tudjanak.

Az alsó részben a becsatlakozott felhasználók beállított nevei olvashatóak. Ez a rész kezdetben üres, de amint valaki beállítja a nevét, megjeleníti azt, illetve ha valaki módosítja, ez a rész is frissül. Ebben a részben jelenik meg a host számára a játék elindítására alkalmas „start game” gomb, mint ahogy az a 4-es ábrán is látszódik. A nem host felhasználóknál nem jelenik meg ez a gomb, helyette egy felirat olvasható, hogy a hostra kell várni a játék kezdéséhez, mint ahogy az az 5-ös ábrán is látható.

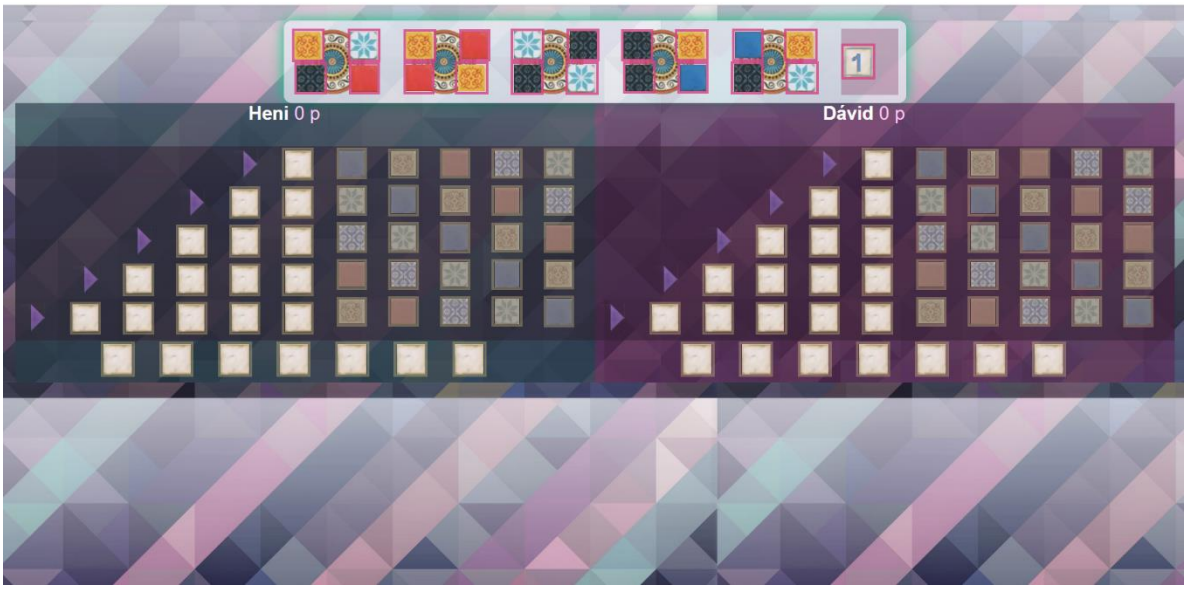


7. ábra: A várószoba négy játékos esetén egy nem host szemszögéből

Fontos, hogy játékot indítani csak akkor lehet, ha 2, 3 vagy 4 játékos van a szobában, ugyanis a játék ezekkel a létszámokkal játszható csak.

2.3.3 Játék oldala

A sikeres játék elindítást követően minden felhasználó számára megjelenik maga a játék oldala, melyet a 8. ábrán láthatunk.

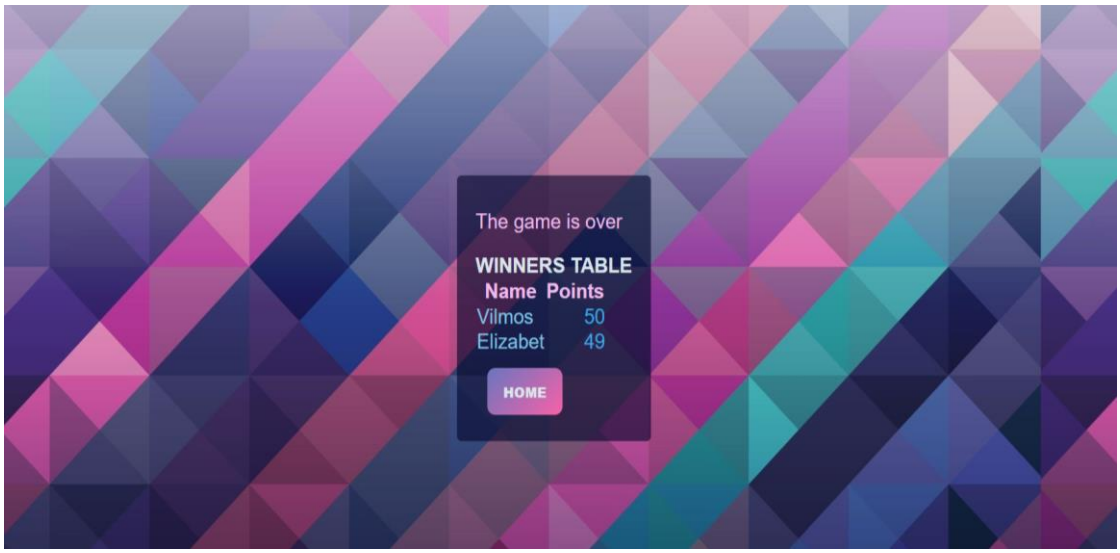


8. ábra A játék oldala a játék indítását követően két játékossal

Két fő részre lehet bontani, felül helyezkedik el a piac és alatta a játékosok táblái. Zöld neon fénnyel jelzi az aktív játékos számára hogy melyik részre kell kattintania.

Minden felhasználónak van egy saját játék táblája, melyek különböző háttér színekkel vannak megkülönböztetve, illetve mindegyiken szerepel a játékos által beállított becenév is. Ha valaki nem állított volna be korábban becenevet akkor nála egy generált név látható.

2.3.4 Játék vége



9. ábra: Játék vége nézet két játékos esetén

2.4 A játék működése

A program teljes mértékben az eredeti Azul szabályait[2] követi.

A 2-4 fő számára játszható Azul lényege, hogy minden játékos egy falat építsen meg csempékből. A játék nyertese az, aki a csempék építgetésével a legtöbb pontot szerzi. A játék megértéséhez fontos megismerni a játékelemeket.

[2] https://fejlesztojatekvilag.hu/uploaded/hu-azul-rules-2017-12-04.pdf

2.4.1 A játék elemei

Csempe: 5 féle különböző színű csempe van a játékban, ezeket kell gyűjteni

10. ábra: Csempe fajták kinézete

Kezdőjátékos jelölő: Csempéhez hasonló kinézetű jelölő, mely megszerzésével a játékos kezdőjátékossá vállhat a következő körre.

11. ábra: Kezdőjátékos jelölő

Manufaktúra: 4 véletlenszerűen kiválasztott csempe tárolására alkalmas korong.

12. ábra: Kép egy manufaktúráról

Maradék gyűjtőhely: A manufaktúráról ki nem választott csempék gyűjtőhelye.

13. ábra: A maradékgyűjtőhely a kör kezdetén

14. ábra: A maradékgyűjtőhely egy játék közbeni állapota

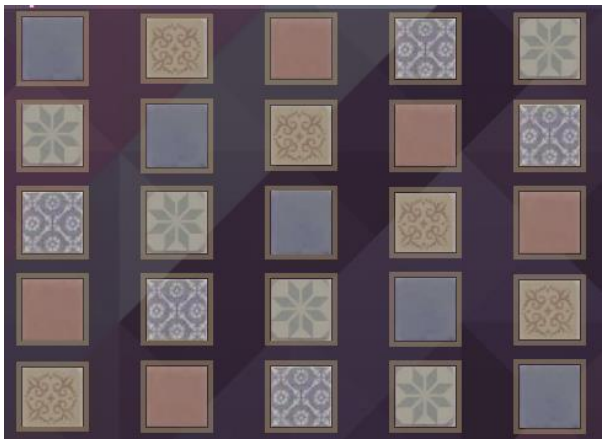
Piac: Játékosszámtól függő számú manufaktúrából és a maradék gyűjtőhelyből áll, innen lehet beszerezni a csempéket.

15. ábra: A játék kezdetén a piac egy két fős játék esetén

Gyűjtőtábla: A csempék összegyűjtésére szolgál, a piacról elvett darabok a kézbők ide kerülnek.

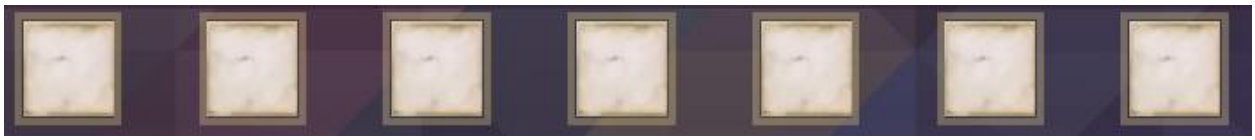
16. ábra: Kezdeti állapotban az egyik játékos gyűjtőtáblája

Fal: 5x5-ös tábla, ide kerülnek a gyűjtőtáblából megfelelő mennyiségű darab összegyűjtése után a csempék. A csempehelyek halvány háttere jelzi, hogy hova melyik színű csempe fog kerülni.



17. ábra: Kezdeti állapotban a fal

Mínusz tábla (vagy mínuszpont tábla): 7 hellyel rendelkező tábla, melyben a negatív pontot érő csempék gyűlnek.



18. ábra: A kör kezdetén a mínusz tábla

Játékos információs sáv: A játékos nevét, pontszámát, kezében tartott (piacról elvett, de gyűjtőtáblába még le nem tett) csempéket tartalmazza.

19. ábra: Egy játékos információs táblája játék közben

Játékos tábla: A játékos információs sávja, a gyűjtőtábla, a fal és a mínusztábla összesége.

20. ábra: Egy játékos tábla a játék során

2.4.2 Játék előkészületei

Minden játékos kap egy játékostáblát, mely kezdetben üres, tehát se a kezében nincs csempe, se a gyűjtőtáblán, se a mínusz táblán, se a falon. Az információs sávban a neve mellett kezdeti pontszámként a 0 pont található.

A játékosok számától függően felkerülnek a piacra a manufaktúrák:

- 2 játékos esetén 5 db
- 3 játékos esetén 7db
- 4 játékos esetén 9 db

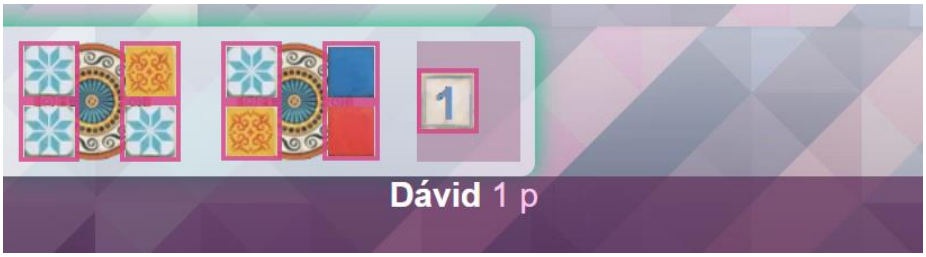
Az előkészületekbe tartozik a kezdőjátékos kiválasztása is, aki az elején minden esetben a host, aki a játékot elkezdte.

2.4.3 Játék menete

A játék körökre van bontva. Minden körben a játékosok egymás után csempéket vesznek el a piacról majd azt lehelyezik a gyűjtőtáblájukra. Az aktív játékost minden esetben segíti a neonzöld világítás, mely megmutatja, hogy a játéktér melyik részére kell kattintania.

21. ábra: A táblán zöld fénnyel van jelezve, hova kell most az aktív játékosnak kattintania

Amikor a játékos a piacról vesz el egy csempét, akkor a kiválasztott manufaktúrából vagy a maradékgyűjtőről az összes ugyanolyan színű csempét a kezébe veszi. Ha a maradék gyűjtő részről vesz fel csempét, és ő az első aki onnan választ, akkor a kezdőjátékosjelölőt is megkapja, ami lehelyeződik a mínusz táblájába a játékosnak.



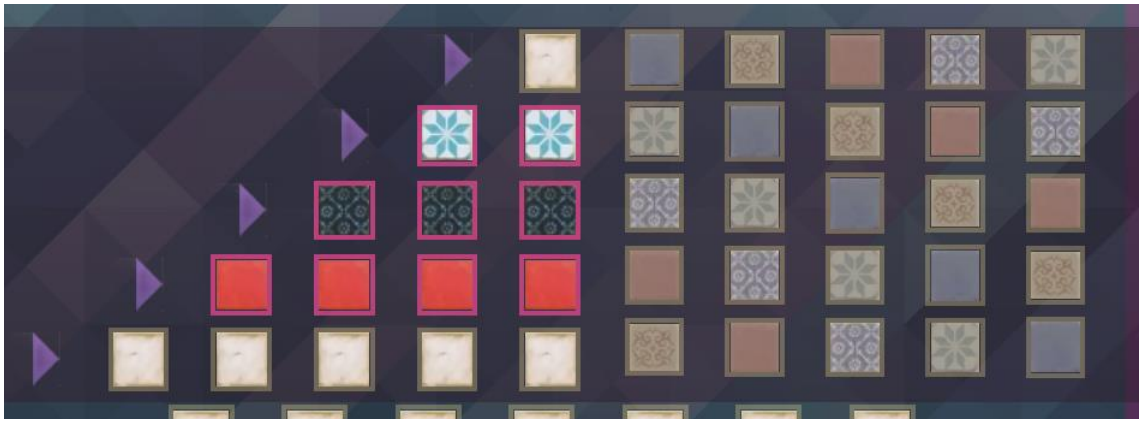
22. ábra: A piac részlete, mielőtt a játékos elvenne egy csempét

23. ábra: A piac részlete, miután a játékos elvett három világoskék csempét az utolsó előtti piacról

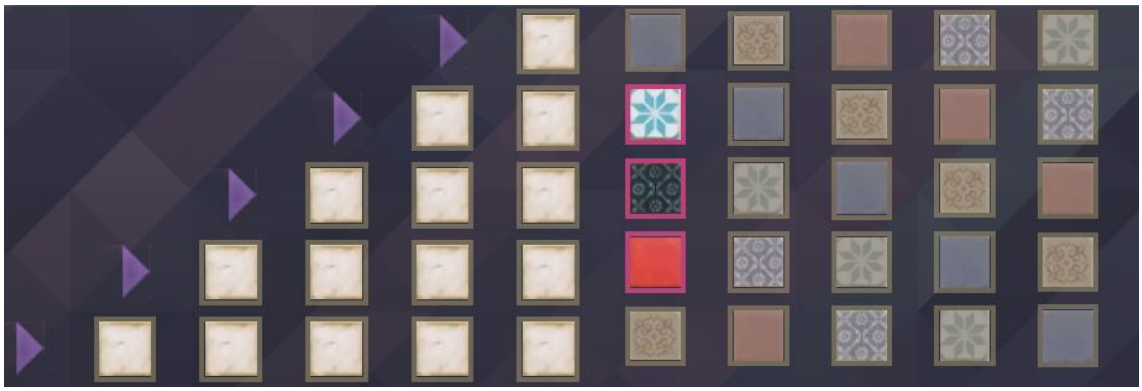
Ezután a gyűjtő táblában kell a kiválasztott csempét/csempéket elhelyeznie a következő szabályok szerint:

- ha egy sorban már elkezdett gyűjteni egy másik szint, akkor oda nem helyezheti
- ha egy sorban már megépítette a falba a választott színű csempét akkor oda se helyezheti le
- ha nem tudja, vagy nem szeretné lehelyezni a gyűjtő táblába, akkor le helyezheti a mínusz táblába is
- ha több csempe van a kezében mint amennyi elfér a gyűjtő tábla kiválasztott sorában, akkor a maradék csempék a mínusz táblába kerülnek.

A csempék elvételét és lerakását felváltva csinálják a játékosok egészen addig amíg el nem fogy az összes csempe a piacról. A kör végén a gyűjtőtáblában teljesen kitöltött sorok csempéje beépül a fal adott sorába. Az új csempék bekerülése lepontozódik, majd a kezdőjátékos jelölő visszakerül a maradék gyűjtőhelyre. Újabb csempék kerülnek a manufaktúrába, és egy új kör kezdődik, melynek kezdőjátékosa az a játékos, akinél a kezdőjátékos jelölő volt.



24. ábra: A teli sorok kör vége előtt



25. ábra: A teli sorok kör vége után beépültek a falba.

2.4.3 Pontozás

Amikor egy kör végén átkerülnek a gyűjtőtáblából a falba a csempék, külön-külön pontot érnek. Az éppen beillesztett csempedarab környezetét vizsgálva számolja ki, hogy mennyi pont jár érte.

A pontszámításhoz több tényezőt is figyelembe kell venni:

- ha a csempe környezetében (vele oldalszomszédosan) nem található másik csempe, 1 pontot ér.

-ha alatta vagy felette található csempe, annyi pontot kap a játékos az oszlopért, amennyi csempéből áll az összefüggő rész.

-ha mellette található csempe, annyi pontot kap a játékos a sorért, amennyi csempéből áll az összefüggő rész.

- ezeken kívül plusz pontok is szerezhetőek:

    - teljesen kitöltött sorért a falban 2 pont jár
    - teljesen kitöltött oszlopért 10 pont jár
    - ha egy színből összegyűlik az összes (tehát 5), az 15 pontot ér

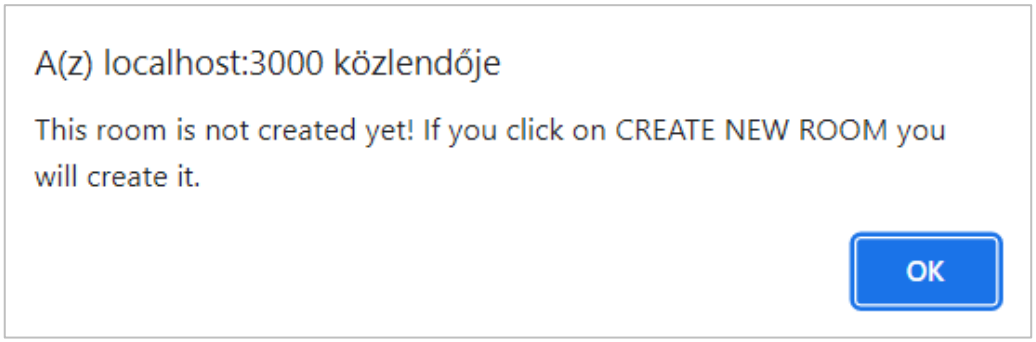
2.5 Hibalehetőségek

Léphetnek fel hibák a szobákba való helytelen bekapcsolódás során, ha a felhasználó nem ad meg vagy nem helyes szobakódot ad meg, a név

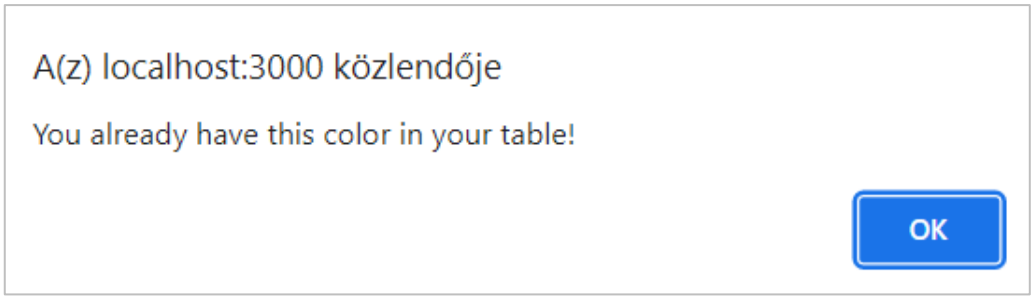
kitöltéséné, ha az input mező üres. A játéktéren is vannak hibalehetőségek, ha a játékos nem a saját táblájára kattint, vagy olyan helyre szeretne csempét lehelyezni, ahova nem lehetséges.

2.5.1 Hibajelzések

A hibák felbukkanó ablakokkal vannak jelezve:



26. ábra: Ha a felhasználó egy nem létező szobába szeretne csatlakozni

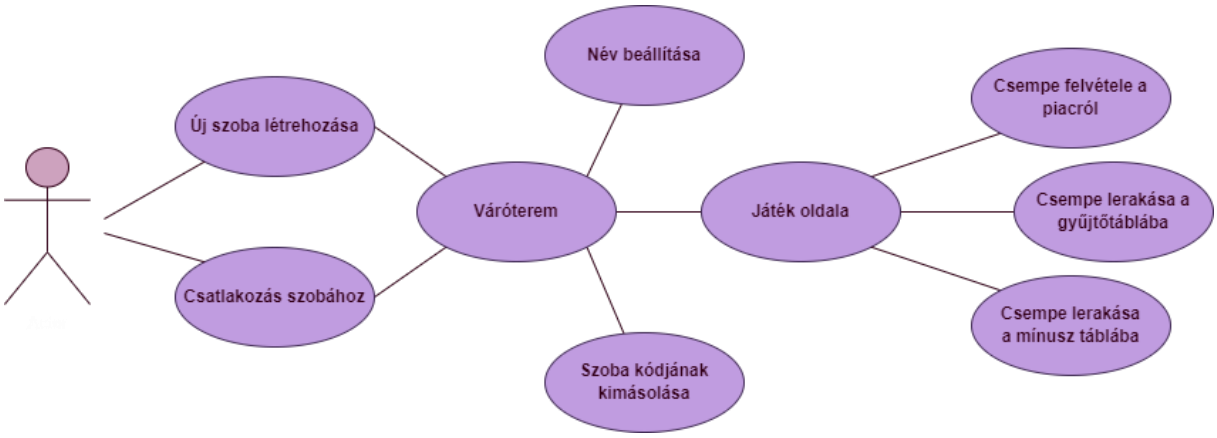


27. ábra: Hibajelzés, ha a játékos olyan sorba tenne le csempét, melyben már falba van építve az a szín.

3 Fejlesztői dokumentáció

3.1 Felhasználói esetek

A következő példán látható a felhasználói esetek diagramja.



28. ábra Használati eset diagram

A felhasználónak kezdetben lehetősége van új szobát létrehozni, illetve már meglévőhöz csatlakozni. Ezután megjelenik a váróterem, ahol be tudja állitani a becenevét illetve ki tudja másolni a szoba kódját. A játék elindulása után a felhasználok elött a játék oldala jelenik meg. Itt lehetőségük van a piactól csempét felvenni, és csempét lerakni a gyűjtőtáblára vagy a mínusz táblára.

3.2 A játék funkciói

3.2.1 Szerepkörök

Egészen a játék kezdetéig két szerepkört különböztetünk meg, van egy host és van(nak) az egyszerű userek. Host az lesz, aki a szobát létrehozza, akik később csatlakoznak a szobához, ők a sima userek. A host az egyedüli, aki a játékot elindíthatja, csak nála jelenik meg a „start game” gomb. Ezen kívül a játék betöltésekor a host lesz automatikusan a kezdőjátékos az első körben. A játék további részében nincsenek megkülönböztetve egymástól a szerepek, mindenki egyenrangú játékos.

3.3 Technológiai áttekintés

Ebben a részben egy áttekintést szeretnék adni az általam használt technológiákról és eszközökről.

3.3.1 React[3]

A React egy nyílt forrás kódú (open-source) JavaScript könyvtár, amit a Facebook fejlesztett. Különlegessége, hogy komponens alapú fejlesztést biztosít, ami azt jelenti, hogy változáskor csak azok a kódrészletek frissülnek, melyek tartalmazzák az új értékekkel frissült változókat, melyeket itt állapothorgoknak nevezünk. Ez a kódot sokkal hatékonyabbá és kiszámíthatóbbá teszi, és egy játék esetében különösen hasznos, hogy csak megadott részletet frissít, és egy apró változtatás miatt nem tölti újra az egész oldalt.

A komponens alapú fejlesztés másik nagy előnye, hogy az elkülöníthető oldalelemek külön komponensekben vannak, így a kód is sokkal átláthatóbb, például az Applikáción belül van a szobakezelés, a várakozó szoba és a játéktér, amik egymástól jól elkülöníthető komponensek.

A komponensek egymással szülő-gyerek viszonyban is tudnak lenni, ilyenkor eseményeken keresztül tudnak kommunikálni egymással. Tehát például a gyerek-komponens akár a szülő-komponens állapothorogját is tudja frissíteni, ezzel frissítve magát a szülő-komponenst (pl utolsó ember lehelyezi a piac is frissül)

3.3.2 Valtio[4]

A valtio az általunk megadott js objektumot egy proxyivá alakítja. Különbség a korábban említett állapot horog és a proxy között az, hogy a proxyt függvényen kívül is lehet definiálni és használni. Tradicionális JavaScript objektumokhoz hasonlóan lehet módosítani, tehát szimpla egyenlőségjellel lehet érteket adni neki.

A useSnapshot-tal lehet elkapni a proxy változásait, ami egy readonly (csak olvasni lehet, módosítani nem) objektummal tér vissza, így a megjelenítésben érdemes csak használni.

[3] https://hu.reactjs.org/
[4] https://github.com/pmndrs/valtio

3.3.3 Yjs[5]

Az Yjs jelenleg a leggyorsabb CRDT (konfliktusmentes replikált adattípus), amivel kollaboratívan alkalmazásokat lehet létrehozni. Benne megosztott adattípusokat lehet használni, amiket konkurensen lehet manipulálni. Ezek az adattípusok a map és az array, ami itt YMap és YArray.

Minden ilyen típust egy YDoc típusú osztályból származtatunk, ami egy gyűjteménye ezeknek az automatikusan szinkronizálódó megosztott objektumoknak. Ezt a dokumentumot éri el minden kliens egy lokális vagy egy már létező szerveren keresztűl. A változtatások automatikusan szinkronizálódnak minden kliensnél.

3.3.4 valtio-yjs[6]

A valtio.yjs összekapcsolja a valtio proxyt az yjs megosztott dokumentumával, ezzel ötvözi a két technológia legjobb tulajdonságait. Ezt a kettőt a*bindProxyAndYMap* vagy a *bindProxyAndYArrray* függvényekkel lehet összekapcsolni, attól függően hogy map-et vagy array-t szeretnénk társítani a proxyhoz.

Ez a technológia pillanatnyilag alfa verzióban van, amit jelenleg is tesztelnek, ezért akár bugokat is lehet benne találni.

3.3.5 Y-websocet[7]

Az y-websocet a sima websocetprovhoz hasonlóan ez is egy egyszerű kliens-szerver kapcsolatot hoz létre, ahol mindegyik kliens kap egy dokumentumot, amit megváltoztathatnak és ezek a frissítések megjelennek a többi kliensnél is. Akár tartalmazhatnak információkat egy kurzor hollétéről (például ahogy a google dokumentumnál látszik, hogy ki hol van éppen)

Y-websocetet létre lehet hozni lokálisan is, de a fejlesztők hozzáférést adnak egy ingyenes szerverhez, ahol megkötések nélkül bárki bármennyi felhasználót és bármennyi szobát létrehozhat, ennek a szervernek ez az elérési útja: „wss://demos.yjs.dev”.

[5] https://docs.yjs.dev/
[6] https://github.com/dai-shi/valtio-yjs
[7] https://github.com/yjs/y-websocket

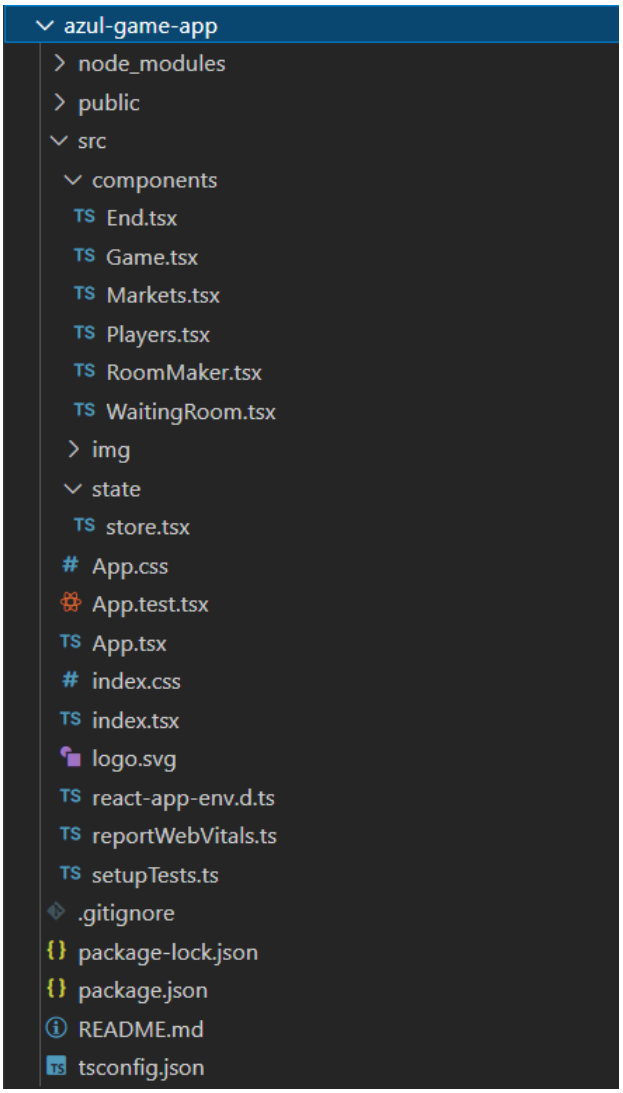
Létrehozásnál meg kell adni a szerver elérési útját, a szobanevet (amit felcsatlakozás előtt be lehet állítani), és egy ydoc típusú dokumentumot.

3.3.6 Visual Studio Code

Az Visual Studio Code (rövidítve: VS Code) egy alkalmazás, mely fejlesztőknek nyújt segítséget a kód írásban, biztosít egy környezetet, IDE-t (integrált fejlesztői környezetet). a Microsoft készítette. A programom megírását tejes egészében Visual Studio Code használatával írtam.

3.4 Kódszervezés

Így épül fel a könyvtárstuktúra:



29. ábra: Könyvtárstuktúra

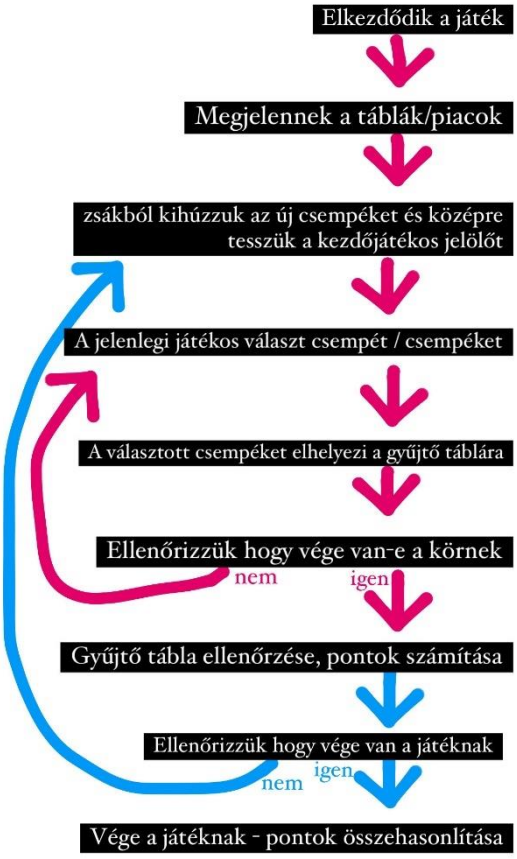
A components mappában találhatóak a komponensek, a state mappában az állapotteret tartalmazó state.tsx, és az összes kép az img mappában van.

3.4 Állapottér

Az állapotteret és a hozzá tartozó műveleteket a store.tsx nevű file tartalmazza.

3.4.1 A játék folyamata, állapotai

A játék folyamatát már az elején lerajzoltam, miközben terveztem, hogy milyen állapotok legyenek majd.



30. ábra: Játék menetének folyamatábrája

Végül a játék állapotainak definiálására számkódot használok melyek jelentése a következő táblázatban látható:

-1	üres hely, ahol nincs csempe
0	játéktér vagy annak elemeinek betöltése, generálása
1	csempe kiválasztása a piacról
2	csempe lehelyezése a gyűjtőtáblába
3	a kör végével, játék végével kapcsolatok ellenőrzések, kör vége esetén a kör lezárása, pontozás
5	a játéknak vége

3.4.2 Típusok és Interfész

A hatékony adattárolás érdekében bevezettem két új típust, először ezeket mutatom be.

A csempe fajtájának, színének definiálására számkódot használok melynek jelentése a következő táblázatban látható:

0	üres hely, ahol nincs csempe
1	sötétkék csempe
2	sárga csempe
3	piros csempe
4	fekete csempe
5	világoskék csempe

user: A szobákba való belépéshez és szobák kialakításához használt típus. A felhasználókat, akik a szobában vannak, user típusként tárolja a program. A következő adattagjai vannak:

- **id**: szám, a felhasználó egyedi azonosítója
- **nickname**: szöveg, a felhasználó maga beállíthatja arra amire szeretné, miután létrehozott vagy belépett egy szobába

player: A játék során használt típus. A játék résztvevőit és adatait, melyek a játék szempontjából fontosak, player típusként vannak tárolva. Adattagjai:

- **id:** szám, a player egyedi azonosítója
- **name:** szöveg, a felhasználó korábban beállított beceneve, vagy ha nem állított be, akkor egy generált név
- **points:** szám, a játékos pontszámát tartalmazza
- **choosenTile:** további adattagokkal rendelkezik, a kiválasztott csempét tartalmazza:
    - "type": szám, a csempe színének kódja
    - "number": szám, a csempék darabszámát tárolja
    - "market": szám, azt a manufaktúrát jelöli, amelyikről a játékos elvette a csempéket
- **collectorTable:** számokból álló diagonális mátrix, a gyűjtőtábla adatait tárolja
- **tileTable:** számokból álló 5x5-ös mátrix, a falat tárolja
- **minusPointsTable:** 7 elemű számokból álló tömb, a mínusz tábla adatait tárolja

IStore: Ezzel interfésszel adtam meg az objektumom szerkezetét, ami a következő:

- **game,** amiben a következő adatok vannak:
    - **templateTileTable:** 5x5-ös mátrik, melyben a falba beépítési szabály szerepel
    - **tileBag**: számokból (csempék típusai) álló tömb, a játék húzózsákját reprezentálja
    - **numberOfPlayers:** szám, a játékosok száma
    - **currentPlayerID:** szám, az aktív játékos indexének a száma.
    - **currentStateID:** szám, a játék jelenlegi állapotának kódját tárolja.

- **players:** player típusokból álló tömb, mely a játékosokat tartalmazza.
- **numberOfmarkets:** száma, a játékosszámhoz tartozó piac elemeinek számát tárolja.
- **markets:** tömb, mely számokból álló tömböket tartalmaz, a piac elemeit tárolja, a 0. indexben a maradék gyűjtőhelye van, és a többi indexen a manufaktúrák.

- **synced:** logikai érték, mely megmutatja, hogy a szobába sikeresen belépett-e a felhasználó
- **clients:** user típusú tömb, ez tárolja a felhasználókat, akik a

2.4.3 Műveletek

A következő részben az állapottér üzeneteit szeretném bemutatni:

SetMarkets:

A játékosszám alapján létrehozza a megfelelő mennyiségű piac elemet: Két játékos esetén 5 manufaktúrát és a maradékgyűjtőhelyet, három játékos esetén 7 manufaktúrát és a maradékgyűjtőhelyet, négy játékos esetén 9 manufaktúrát és a maradékgyűjtőhelyet, és feltölti üres elemekkel. Majd meghívja a GenerateMarkets függvényt.

GenerateMarkets:

A maradékgyűjtőhelybe beilleszti a kezdőjátékosjelölőt és a húzózsákból (Tilebag) a manufaktúrák feltölti a húzózsák elemeinek áthelyezésével.

GenerateTilebag:

A húzózsákot feltölti, mind az 5 fajta csempéből 20 kerül belem majd ezeket összekeveri, hogy a sorrendjük véletlenszerű legyen.

GetMarkets:

A manufaktúra és annak elemének id-ja alapján visszatér az ott lévő csempe fajtájával.

SetPlayers:

Létrehozza a játékosokat, és azokat beilleszti a játékosok tömbjébe.

NewGame:

Játék kezdése, beállítja az alap adatokat (mint például a játék állapotának változóját 0-ra), meghívja a SetPlayers, GenerateTilebag, és SetMarkets függvényeket, majd a kiválasztja a kezdőjátékost (aki a host) és a játék állapotának kódját 1-re állítja.

nextPlayer:

A játék tovább lép a következő játékosra.

freePlacesInCollectorTable:

Visszatér azzal hogy a megadott játékosnak a gyűjtőtáblájának megadott sorában hány szabad hely található.

CanPutInTheRow:

Visszatér azzal, hogy a játékos a gyűjtőtábla kiválasztott sorába lehelyezheti-e a kezében tartott csempét. Itt megvizsgálja, hogy a gyűjtőtáblának kiválasztott sorában van-e már más típusú csempe, illetve hogy a falban ugyan ebben a sorban van-e már beépítve ez a típusú csempe.

minusPointsTableLength:

Visszatér azzal hogy a mínuszpont táblában, hogy mennyi szabad hely található, úgy hogy megvizsgálja hány nullával kitöltött cella van.

DeleteTilesFromMarket:

A kiválasztott piac elem és a csempe indexe alapján a piac eleméből törli az összes kiválasztott csempével megegyező színű csempét. Ha a maradékgyűjtő helyen lett kiválasztva a csempe, és még a kezdőjátékos jelölő itt van, akkor a kezdő játékos jelölő átkerül a jelenlegi játékos mínuszpont táblájába.

PutFromMarket:

A kiválasztott piac elem és csempe indexe segítségével a piac elemből áthelyeződnek a kiválasztott csempék a játékos kezébe, azaz a táblájának információ sávjába.

PutToTable:

A játékos kezében lévő csempe vagy csempék lehelyeződnek a játékos által kiválasztott gyűjtőtáblabeli sorba vagy mínuszpont táblába, amennyiben a játék szabályai szerint le helyezhetőek oda.

PutFromCollectorToTileTable:

Az adott játékos nak a gyűjtő táblájának adott sorát vizsgálva helyezi át a falba a csempéket. Ez akkor lehetséges, ha a gyűjtő tábla beli sor teljesen fel van töltve ugyanolyan színű csempékkel. Meghívja a calculatePoints függvényt.

PutAllFromCollectorToTileTable:

Az összes játékos gyűjtő táblájának összes sorára meghívja a PutFromCollectorToTileTable függvényt.

calculatePoints:

A falba újonnan bekerült csempe hatására kapott pontokat számolja ki és adja hozzá a játékos összpontszámához, illetve a mínusz táblában lévő pontokat vonja le a játékos pontszámából. Ennek meghatározására meghívja a calculateMinusPoints függvényt. Megvizsgálja hogy az új csempe mellett vannak-e másik csempék. Ezt egy while ciklussal teszi, mely vizsgálja mind a négy oldalán lévő szomszédait, és azok szomszédait, ..stb.

calculateMinusPoints:

Megvizsgálja, hogy pluszpont adható-e, tehát az új csempe bekerülésével egy oszlop jött-e létre, vagy egy sor, vagy összegyűlt az adott színből az összes csempe.

Az adott játékos mínusz pontjait számolja ki a mínusz tábla alapján. A mínusz tálba szabályát betartva, tehát az első két elem - 1 pontot, a következő három elem - 2 pontot, és az utolsó két elem - 3 pontot ér.

NewRound:

Új kör kezdéséhez készíti fel a játékteret, ha a húzózsák kifogyott, akkor újra generálja, ha még van a húzózsákban elem, akkor azokat kiosztja a GenerateMarkets függvény meghívásával.

CheckRoundEnds:

Ellenőrzi, hogy vége van-e a körnek, tehát a piacon van-e még elvenni való csempe.

CheckEndGame:

Ellenőrzi, hogy vége van-e a játéknak, tehát van-e olyan játékos, akinek a falában valamelyik sora teljesen megtelt volna csempékkel.

RoundEnds:

A kör végén meghívja a PutAllFromCollectorToTileTable függvényt és CheckEndGame függvény szerint ha vége van a játéknak akkor meghívja GameEnds függvényt, ha nem akkor pedig az új kört kezdő NewRound függvényt.

GameEnds:

Sorba rendezi a játékosokat a pontszámok alapján.

initStore:

A state változóba bele másolja az a kitöltött állapot teret.

addClient:

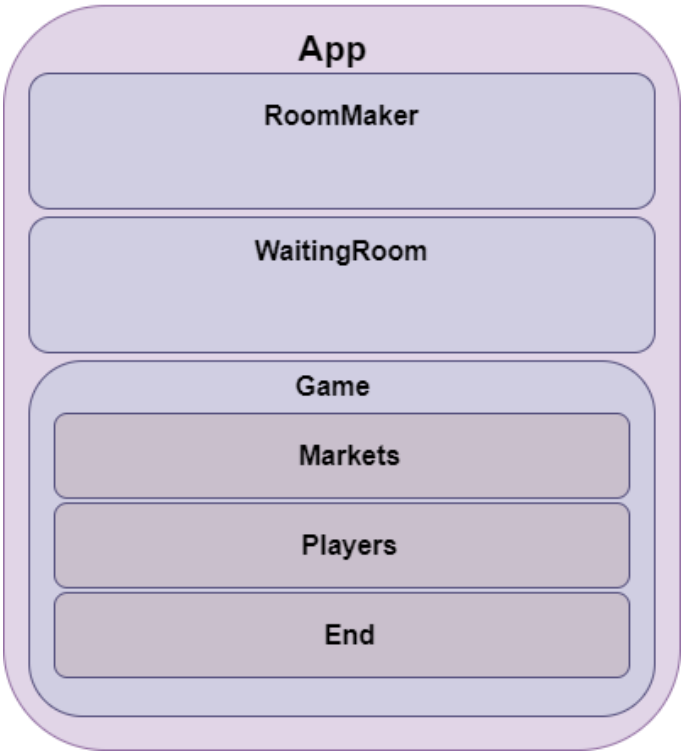
Hozzáadja a klienseket tartalmazó tömbhöz a megadott klienst a megadott id alapján.

setSynced:

A synced változó értékét igazra állítja be.

3.5 Komponensek

A React keretrendszer támogatja a komponens alapú webfejlesztést, és ezt kihasználva komponensekbe szerveztem az appot. Az alábbi képen látható ennek felépítése:



31. ábra: Komponensek felépítésének szemléltetése

Ahogy az a 29. ábrán is látszik, a legfőbb komponens az az App, és ebben van minden más komponens is. Az appot három komponensre lehet osztani a RoomMaker, a WaitingRoom és a Game. A Game-en belül további három komponens van.

A következőekben ezeket a komponenseket szeretném bemutatni.

3.5.1 RoomMaker

Ez a komponens biztosítja, hogy a felhasználó szobákat tudjon létrehozni, illetve szobákba tudjon csatlakozni. Az alkalmazás kezdőképernyőjén a felhasználó már rögtön ennek a komponensnek a megjelenítésével találkozik.

3.5.2 WaitingRoom

A szobába való csatlakozás vagy szoba készítés után ez a következő komponens. Ebben a részben a várószoba van.

3.5.3. Game

Ez a játéktér komponense. Ezen belül három komponens található: Market, Players és End.

3.5.4 Market

A Game komponensen belüli komponens. Ebben a piac elemei kerülnek megjelenítésre. Hozzáadódnak a funkciók, mint hogy az aktív játékos tudjon levenni csempéket a piacról. Fontos változója a marketdivs, ugyanis ebbe kerülnek bele a piac elemei.

3.5.5 Players

A Game komponensen belüli komponens. Ez felel a Játékosok tábláinak megjelenítéséért és megfelelő működéséért. Az állapottér alapján legenerálja minden egyes játékosnak a játékos tábláját és ellátja a szükséges funkciókkal, például, hogy az a játékos a körében rá tudjon kattintani a gyűjtőtáblára, amikor a kezében lévő csempéket beletenné.

3.5.6 End

A Game komponensen belüli komponens. A játék végével jelenik meg. Legfontosabb része a táblázat, melyben a játékosokat jeleníti meg pontszámaikkal együtt, pontszám alapján csökkenő sorrendbe rendezve.

Szerepel itt egy gomb is, mellyel az oldal újra töltésével a kezdőlapra kerül a felhasználó.

3.6 Játék működése

Minden komponensben egy snap változó van, ami useSnapshottal a state-re van feliratkozva, így érzékeli bármilyen változás van is a benne.

A komponensek használhatják a state függvényeit, és ezzel idézhetnek elő változást az állapottérben. Így működik például a piacról a csempe elvétele. A Market komponens a stateben definiált PutFromMarket-et használja a módosításra. A helyzet hasonló a Players komponens és a

felhasználó csempe lehelyezése esetében is. A komponens meghívja a PutToCollectorTable-t és így változtatja meg az állapotteret.

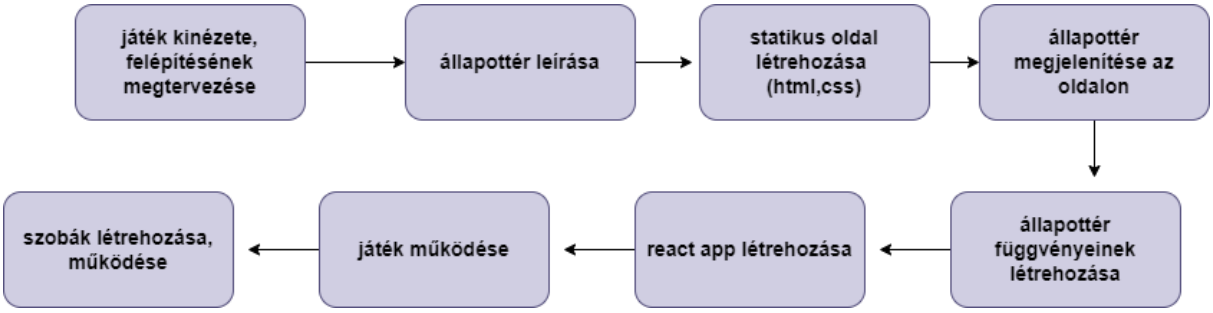
Ezzel szemben, ha valamilyen változást a komponens idézne elő, amiben egy olyan függvényt használna, ami nem az övé, hanem például a szülőkomponensé, az már összetettebb.

Ezt egy példán keresztül szeretném bemutatni:

A játék elindításához a WaitingRoomban kell a host felhasználónak megnyomnia a „start game” gombot. Ennek hatására egy sikeres ellenőrzést követően elindul a játék minden felhasználó oldalán. A működés háttere az, hogy a gomb megnyomásakor kiváltódig egy NewGameEvent és egy setGameIsStartedEvent. Mindkettőt az ő szülője, az App fogja lekezelni. Az eventhandlerjeiben meghívódnak a saját függvényei, tehát a handleSetGameISStartedEvent-ben meghívja a setGameIsStarted settert, mellyel beállítja a GameIsStarted változó értékét igazra. Ennek hatására vált át a nézet a játéktáblára minden felhasználónál.

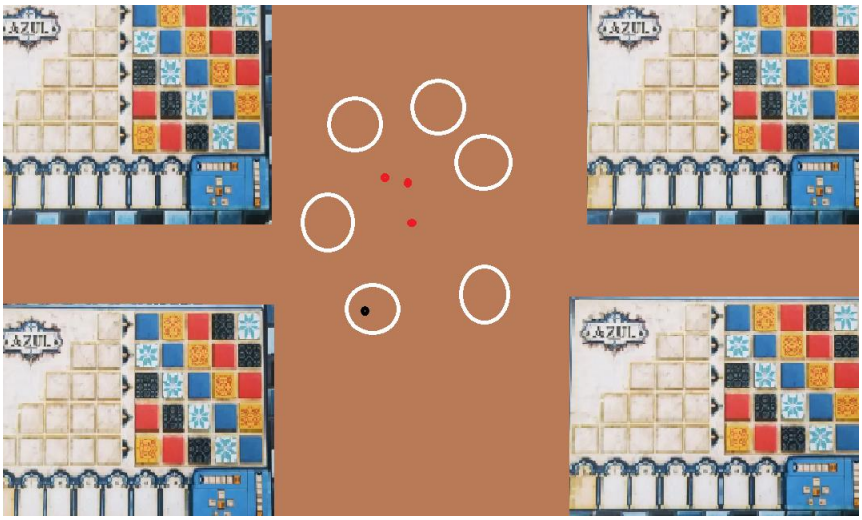
3.6 Fejlesztés folyamata

Az alábbi ábrán jól látható, milyen folyamatok voltak a fejlesztés során:



32. ábra: Fejlesztés folyamatai

Kezdetben a kinézetének terve a 33. ábrán látható. Nem kevés változtatáson ment keresztül, míg megszületett a végeredmény.



33. ábra: A játék kinézetének terve

3.8 Fejlesztés során felmerült problémák

Szeretnék említést tenni néhány fontosabb problémáról, melyek számomra igazán érdekesek voltak.

3.8.1 Szinkronizálási problémák

Ahogy a valtio-yjs-sel kapcsolatban már említésre került, ez még egy alfa verzióval rendelkező könyvtár, ami azt is jelenti hogy annyira nem elterjedt, és azt is, hogy esetleg tartalmazhat hibákat. A programmal kapcsolatban is fennált a probléma, hogy nem megfelelően működött. Számunkra elsőre érthetetlen okokból nem mindig sikerült szinkronizálni a szobát. Gyakorlatban az a problémám volt, hogy amikor a felhasználó becsatlakozott egy már létező szobába, akkor esetenként ez ahhoz vezetett, hogy kitörölte a már szobámban tartózkodók tömbjét, tehát lényegében felülírta néha az állapottér már beállított részeit. Kezdetben nem jöttünk rá, hogy ez miért történik, de nem is mindig fordult elő, csak az esetek felében.

A valtio-yjs hivatalos oldalán olvasható volt, hogy mivel teszt fázisban van, ezért előfordulhatnak bugok ezzel kapcsolatban. Ennek ellenére nem állt össze a kép, ugyanis a példa amelyet a hivatalos oldalukon a megadtak, soha nem produkálta ezt a hibát, mégis a működésének elve ugyanaz, ami a programomban szerepel.

A probléma megoldása az volt, hogy a valtio felülírta az Ydocot. Ez azért fordult elő, mert kezdetben a state nem volt üres, és ezért írta felül az új felhasználó az Ydoc-ban a dolgokat. A kódot átalakítva, úgy hogy kezdetben a state üres legyen és csak utólag legyen inicializálva, teljes mértékben megoldotta ezt a problémát.

3.8.2 Objektumok szinkronizálása

Kezdetben célom volt az objektumorientáltság megtartása és az állapotér adatait objektumokban szerettem volna tárolni. Viszont a gyakorlatban ez nem volt olyan egyszerűen megvalósítható, ugyanis szinkronizációs problémák léptek fel az alobjektumokkal kapcsolatban.

Ezt a megoldási tervet végül el kellett vetni.

3.9 Tesztelés

Igyekeztem minél részletesebben letesztelni a programot és annak minden funkcióját. Ebben a fejezetben részekre lebontva bemutatom, milyen teszteket végeztem el sikeres eredménnyel.

3.9.1 Szoba létrehozásával, bekapcsolódásával kapcsolatos tesztelés

sorszám	leírás	eredmény
1	oldal megjelenésekor az input mezőben egyedi azonosító jelenik meg	az oldal frissítésekor minden esetben új és másik azonosító kerül az input mezőbe
2	az input mezőben szereplő kód módosítása	az input mezőre kattintva lehet módosítani, akár teljesen új kódot beleírni
3	create new room gombra kattintintás kitöltött input mezővel	Létrehoz egy szobát és a várószobába (Waiting Room) kerül a felhasználó.
4	create new room gombra kattinttatás üres input mezővel	Felugró üzenetben olvasható, hogy üres inputmezővel nem lehet létrehozni új szobát, és nem hoz létre új szobát.
5	create new room gombra	Felugró üzenetben olvasható,

	kattintás úgy, hogy az input mezőben egy már létrehozott szoba kódja van	hogy hiba történt, nem lehet létrehozni a szobát ami már létre van hozva.
6	join room gombra kattintás kitöltött input mezővel, amiben egy már létrehozott szoba kódja szerepel	Sikeresen csatlakozik a szobához, a felhasználó átkértül a várószobába
7	join room gombra kattintás kitöltött input mezővel, amiben egy még nem létrehozott szoba kódja szerepel	Felugró üzenetben olvasható, hogy hiba történt, nem lehet csatlakozni nem létező szobához.
8	join room gombra kattintás üres input mezővel	Felugró üzenetben olvasható, hogy üres inputmezővel nem csatlakozni új szobába.

3.9.2 Várószobával kapcsolatos tesztelés

sorszám	leírás	eredmény
9	A várószobába belép a felhasználó, akkor helyesek az információk	Helyesen mutatja, hány ember van a szobában illetve a szoba kódját
10	A felhasználó a várószobában van, és eközben csatlakoznak új emberek abba a szobába	Az emberek száma helyesen változik.
11	a „set your nickname here” input mező kitöltése után rákattint a „save name” gombra	Sikeresen beállítóik a beceneve, és ez megjelenik a képernyőn az jelen lévő játékos nevei között.
12	a „set your nickname here” input mezőt üresen hagyva rákattint a „save name” gombra	Felugró üzenetben olvasható, hogy nem lehet üres az inputmező.
13	Egy szobában lévő felhasználó beállítja a becenevét	Minden más vele azonos szóbában lévő felhasználónál megjelenik a beceneve a jelenlévő felhasználók között.
14	Egy felhasználó, aki már állított be becenevet magának, kitörli a	Felugró üzenetben olvasható, hogy nem lehet üres az

	„set your nickname here” input mezőt és rákattint a „save name” gombra	inputmező.
15	Egy felhasználó, aki már állított be becenevet magának, átírja a „set your nickname here” input mező tartalmát és rákattint a „save name” gombra	A beceneve sikeresen módosul és új beceneve minden más vele egy szobában tartózkodó felhasználónál megváltozik
16	A felhasználónak megjelenik a szoba kódja amiben benne van, és a „copy room” gombra rákattint.	Vágólapra másolódik a szoba kódja.
17	A felhasználó aki létrehozta a szobát host rangot kap. A játékot csak host indíthatja el.	A felhasználónál megjelent a „start game” gomb.
18	A felhasználó aki már létrehozott szobához csatlakozott, nem indíthat játékot.	A felhasználóknál a „start game” gomb helyett egy felirat jelenik meg, melyben az áll, hogy a hostra kell várni, hogy elindítsa a játékot.
19	A felhasználó, aki a host megnyomja a „start game” gombot.	A felhasználó nézete átvált a játéktérre.
20	A host elindítja a játékot, a nem host felhasználónak nem kell semmit se csinálnia.	A nem host felhasználóknál is átvált a nézet a játéktérre.
21	Egy felhasználó van csak a szobában, és a játék indítását kezdeményezné.	A gombra való kattintáskor egy felugró üzenetben olvasható, hogy nincs elég játékos a játék kezdéséhez. A játék nem indul el.
22	Négynél több felhasználó van a szobában, amikor játékot indítana el a host.	A gombra való kattintáskor egy felugró üzenetben olvasható, hogy a játékosszám nem megfelelő. A játék nem indul el.

3.9.3 Játék tesztelése – Játéktér generálása

sorszám	leírás	eredmény
23	Elindul egy játék két fővel	A játéktéren két játékostábla és öt manufaktúra jelenik meg.
24	Elindul egy játék három fővel	A játéktéren három játékostábla és hét manufaktúra jelenik meg.
25	Elindul egy játék négy fővel	A játéktéren négy játékostábla és kilenc manufaktúra jelenik meg.
26	Egy felhasználó beállította a nevét, mielőtt a játék elindult	Ennek a felhasználónak a tábláján megjelenik a neve, a többiek esetében a generált játékosnév jelenik meg.
27	Minden játékos beállította a becenevét a játék elindulása előtt.	Mindegyik játékos játékostábláján megjelenik a beállított név.
28	Senki sem állított be becenevet a játék kezdete előtt.	Mindenkinek a játékostábláján a generált név szerepel, ami egyedi.
29	A host a kezdőjátékos	Minden esetben, akárhány fővel is teszteltem, a host a kezdőjátékos
30	A játék kezdetében üresek a táblák, a pontszáma mindenkinek 0.	Minden játékos tálbája üres.

3.9.4 Játék tesztelése – Piac (market) működése

sorszám	leírás	eredmény
31	A játék elején véletlenszerűen kerülnek a piacra a csempék.	Minden játék indításakor más a csempék felhozatala.
32	A kör kezdetén a maradékgyűjtőhelyben csak a kezdőjátékos jelölő van.	A maradékgyűjtőhelyben nincsenek a kezdőjátékos jelölőn kívüli csempék
33	A játékos nézetében, aki jelenleg választ a piactér zöld körvonallal világít.	A zöld körvonal csak a jelenlegi játékosnál látszik, a többi játékosnál nem.
34	A játékos nem a saját körében kattint egy csempére.	Semmi nem történik, ilyenkor a csempék nem kiválaszthatóak

35	A játékos a saját körében, kattint egy csempére.	Semmi nem történik, ilyenkor a csempék nem kiválaszthatóak
36	Bármelyik manufaktúráról kiválaszt egy olyan csempét aminek színéből csak egy szerepek azon a manufaktúrán.	Az adott manufaktúra másik három csempéje a maradékgyűjtőhelyre kerül, a kiválasztott színből egy a játékos kezébe (táblája felső részébe) kerül.
37	Bármelyik manufaktúráról kiválaszt egy olyan csempét aminek színéből két darab is van azon a manufaktúrán.	Az adott manufaktúra további két csempéje a maradékgyűjtőhelyre kerül, a kiválasztott színből mind a két darab a játékos kezébe (táblája felső részébe) kerül.
38	Bármelyik manufaktúráról kiválaszt egy olyan csempét aminek színéből három darab is van azon a manufaktúrán.	Az adott manufaktúra maradék egy csempéje a maradékgyűjtőhelyre kerül, a kiválasztott színből mind a három darab a játékos kezébe (táblája felső részébe) kerül.
39	Egy manufaktúrán négy ugyanolyan színű csempe van, és a felhasználó ezt választja ki.	A négy csempe megjelenik a felhasználó táblája felső részében, míg a maradékgyűjtőhelybe nem kerül új elem.
40	A maradékgyűjtőhelyről választ a játékos, mielőtt más tenné.	A kezdőjátékosjelölő a játékos mínusztáblájára kerül.
41	A játékos a maradékgyűjtőhelyen a kezdőjátékos jelölőre kattint, hogy azt válassza ki mint csempét.	Nem veszi fel a kezébe, a kezdőjátékosjelölő nem kattintható.
42	A játékos a maradékjelölőből választ ki csempét.	A csempék a kezébe kerülnek.

3.9.5 Játék tesztelése – Csempe lehelyezése a táblára

sorszám	leírás	eredmény
43	A játék elején a gyűjtőtábla és a fal is üres.	Minden játékosra jellemző a kezdetben üres tábla
44	A játékos a csempe piacról való	Felugró üzenetben látható, hogy a

	elvétele után nem a saját táblájára kattint.	saját táblájára kell kattintania, hogy sikeresen letegye a csempét.
45	A játékos a csempe piacról való elvétele helyett a táblára kattint.	Nem helyeződnek le csempék, ilyenkor nem kattintható a tábla.
46	A játékos nem a saját körében kattint a táblájára.	Nem aktív játékosként nem kattintható a tábla.
47	A játékos nem a saját körében kattint egy másik játékos táblájára.	Felugró üzenetben látható, hogy amire kattintott, az nem a saját táblája.
48	Az aktív játékos a csempe piacról való elvétele után az addig üres mínusztáblába kattint.	A játékos kezében lévő csempék a mínusz táblájába helyeződnek le.
49	Az aktív játékos a csempe piacról való elvétele után a mínusztáblába kattint, ahol már van egy csempe.	A játékos kezében lévő csempék a mínusz táblájába helyeződnek le a már lent lévő csempe melletti szabad helyekre.
50	Az aktív játékos a csempe piacról való elvétele után a mínusztáblába kattint, ahol nincs annyi szabad hely, ahány csempe az ő kezében van.	A játékos kezében lévő csempék kitöltik a mínusz tábla szabad helyeit.
51	Az aktív játékos a csempe piacról való elvétele után a mínusztáblába kattint, ami már teljesen tele van, nincs szabad hely.	A letenni kívánt csempék eltűnnek, a mínusztáblán felül nem adható egyéb mínusz pontot, így ezeket nem kell sehol megtartani.
52	Az aktív játékos a csempe piacról való elvétele után, a kezében egy csempével a gyűjtőtáblájának egy üres sorában lévő elemre/nyílra kattint.	A kézben lévő csempe lehelyeződik a gyűjtőtábla kattintott sorába.
53	Az aktív játékos a csempe piacról való elvétele után, a kezében 2 csempével a gyűjtőtáblájának első üres sorában lévő	A kézben lévő egyik csempe lehelyeződik a gyűjtőtábla első sorába, a második csempe a mínusz táblába helyeződik le.

	elemre/nyílra kattint, a mínusztáblája üres.	
54	Az aktív játékos a csempe piacról való elvétele után, a kezében 2 csempével a gyűjtőtáblájának második üres sorában lévő elemre/nyílra kattint.	A kézben lévő két csempe lehelyeződik a gyűjtőtábla második sorába.
55	Az aktív játékos a csempe piacról való elvétele után, a kezében 5 csempével a gyűjtőtáblájának második üres sorában lévő elemre/nyílra kattint, a mínusz táblája üres.	A kézéből két csempe lehelyeződik a gyűjtőtábla első sorába, a maradék három csempe a mínusz táblába helyeződik le.
56	A játékos kezében fekete csempével a gyűjtőtábla olyan sorába szeretné letenni a csempét, ahol már elkezdett korábban pirosat gyűjteni.	Felugró üzenetben látható, hogy a a csempét nem teheti le, mert már másik színt kezdett el gyűjteni.
57	A játékos kezében sárga csempével a gyűjtőtábla olyan sorába szeretné letenni ami mellett a falban már be van építve a sárga csempe.	Felugró üzenetben látható, hogy a a csempét nem teheti le, mert beépítette azt a színt a falba.
58	A körben az utolsó játékos a mínusz táblába teszi a csempéit.	A mínusz tábla feltöltődik a csempével.

3.9.5 Játék tesztelése – Kör vége és új kör kezdete

sorszám	leírás	eredmény
60	Elfogytak a csempék a piacról	Az utolsó csempe lehelyezését követően pontszámítás történik a teli sorok falba beépítésével.
61	A manufaktúrákból elfogytak a csempék, de a maradékgyűjtőben van.	Még nincs vége a körnek.
61	Az egyik játékosnak megtelt a sora.	Vége a játéknak, a nézet a játék végére vált.

62	Az egyik játékosnak megtelt az egyik oszlopa.	Kap érte plusz 10 pontot, a játéknak ennek hatására nincs vége.
63	Mindegyik játékosnak van teli sora a falában	A játék véget ért.
64	A játékosnak kör végén nincs teli sora a gyűjtő táblában	A játékos nem épít be új elemet, pontot sem kap
65	A játékosnak az össze sora tele van a gyűjtőtáblában.	Mindegyik beépül a falba, megkapja érte a pontokat.
66	Játék vége van.	Helyesen megjelenik a játékosokat tartalmazó táblázat, pontszámuk alapján csökkenő sorrendben sorba rendezve.

4 További fejlesztési lehetőségek

4.4 Animációk

A játék során bármikor, amikor csempék mozognak látványos lenne animációval kiegészíteni. A körök között is, amikor a gyűjtőtáblából a falba épül be a csempe.

4.5 Tanító funkció

A játékszabályok könnyebb megértése érdekében hasznos lenne, ha a játékosok ha szeretnének, végig legyenek vezetve a játékon egy tanító funkció által. Megmutatná hova hogy mi alapján kell kattintani.

4.6 Chat ablak

Egy bezárható chatablak implementálása is hasznos lehet, hogy a játék felületén tudjanak a játékosok írásban kommunikálni.

4.7 Regisztráció és belépési lehetőség

A felhasználóknak legyen lehetőségük regisztrálni, így nem kellene a becenevet minden játék elött beállítani, illetve bevezetésre kerülhetne a profilkép is, mely mindenkinek a táblája jelenhetne meg. Mindenki megtekinthetné az eddig lejátszott játékait és azok statisztikáit.

5 Összefoglaló

Igaz, hogy megéri olyan szakdolgozat témát választani, ami közel áll hozzánk, mert akkor az tényleg szívből fogjuk elkészíteni, nem pedig kényszerből. Én is így éreztem miközben a teljesen nulláról építettem fel ezt a projektet, és valósítottam meg a kedvenc társasjátékomat. Ez idő alatt rengeteget tanultam, ugyanis több számomra új technológiával találkoztam, melyeket meg kellett értenem, és el kellett sajátítanom mielőtt használtam volna őket. A fejlesztés nem volt akadálymentes, de végül minden problémának megoldása született, és sikeresen elkészült a program.

Köszönetnyilvánítás

Fontosnak érzem, hogy ebben a részben megragadjam az alkalmat, hogy kifejezzem hálámat azoknak, akik segítették az utamat, ami ehhez a kész szakdolgozathoz vezetett.

Először is szeretném megköszönni Horváth Győzőnek, akitől rengeteget tanulhattam, és részben neki köszönhetem azt, hogy elkezdtem érdeklődni a webes technológiák iránt, és így egyetemi éveim végére megtalálhattam azt, ami igazán érdekel.

Végtelenül hálás vagyok Anyukámnak és Mamámnak, akiktől rengeteg segítséget kaptam, minden helyzetben támogattak engem, és nélkülük sosem juthattam volna el idáig. Anyukám volt az, akire mindig számíthattam, bármi is történt, és ő fáradhatatlanul, minden helyzetben támaszt nyújtott, amikor szükségem volt rá.

Köszönöm igaz barátaimnak, akik nem csak motiváltak, de hozzájárultak szakmai fejlődésemhez és mellettem voltak, a legnehezebb időszakaimban.

Forrásjegyzet

[1] „Azul játék logója,” [Online]. Available: https://www.netclipart.com/isee/mJRobw_in-the-board-game-azul-from-plan-b/ [Hozzáférés dátuma: 29 05 2022].

[2] „App háttérképa” [Online]. Available: https://kartinkin.net/9627-fon-dlja-oblozhki-v-vk-27-foto.html [Hozzáférés dátuma: 29 05 2022].

"Játékszabály" [Online] Avaiable: https://fejlesztojatekvilag.hu/uploaded/hu-azul-rules-2017-12-04.pdf [Hozzáférés dátuma: 29 05 2022].

Konzultáción mutatott kódrészletek, melyek nem elérhetőek online

6.1 Ábrajegyzék