Eötvös Loránd Tudományegyetem

Informatikai Kar

Catan társasjáték

számítógépes megvalósítása

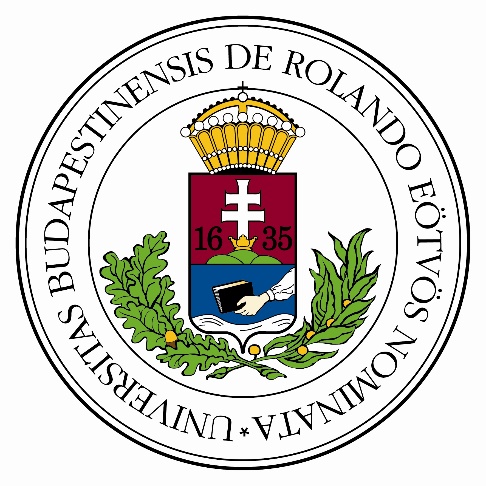
Szakdolgozat

*Készítette: Konzulens:*

Fodor Kristóf Dr. Vörös Péter

RFXQHM, Adjunktus,

Programtervező informatikus BSc – B szakirány Információs Rendszerek Tanszék



Budapest

2023

Tartalomjegyzék

[1. Bevezetés 4](#_Toc136207898)

[2. Felhasználói dokumentáció 5](#_Toc136207899)

[2.1. Játék ismertetője és szabályzata [3] 5](#_Toc136207900)

[2.1.1. Mezőtípusok 5](#_Toc136207901)

[2.1.2. A játék kezdete 6](#_Toc136207902)

[2.1.3. Egy játékos köre 6](#_Toc136207903)

[2.1.4. Építkezés 6](#_Toc136207904)

[2.1.5. Bónuszkártyák 7](#_Toc136207905)

[2.1.6. Rabló mozgatása 8](#_Toc136207906)

[2.1.7. Passzolás 8](#_Toc136207907)

[2.1.8. Játék vége 8](#_Toc136207908)

[2.2. A játék futtatásához szükséges gépigény, programok 8](#_Toc136207909)

[2.3. A játékfelület áttekintése 8](#_Toc136207910)

[2.4. A játék kezelőfelülete 9](#_Toc136207911)

[2.4.1. Regisztráció és bejelentkezés 9](#_Toc136207912)

[2.4.2. Várótermek listája 9](#_Toc136207913)

[2.4.3. Statisztika 11](#_Toc136207914)

[2.4.4. Játék kezdete 11](#_Toc136207915)

[2.4.5. Egy játékos köre 12](#_Toc136207916)

[2.4.6. Bónuszkártyák 14](#_Toc136207917)

[2.4.7. Rabló 14](#_Toc136207918)

[2.4.8. Tengeri kereskedelem 15](#_Toc136207919)

[2.4.9. Kereskedés a játékosok között 15](#_Toc136207920)

[2.4.10. Játék vége 16](#_Toc136207921)

[3. Fejlesztői dokumentáció 17](#_Toc136207922)

[3.1. Felhasznált technológiák a fejlesztéshez 17](#_Toc136207923)

[3.1.1. IntelliJ IDEA 17](#_Toc136207924)

[3.1.2. Kotlin 17](#_Toc136207925)

[3.1.3. Ktor 17](#_Toc136207926)

[3.1.4. Kotest 17](#_Toc136207927)

[3.1.5. Gson 17](#_Toc136207928)

[3.1.6. Hibernate 18](#_Toc136207929)

[3.1.7. TornadoFX 18](#_Toc136207930)

[3.1.8. Gradle 18](#_Toc136207931)

[3.2. Áttekintés, programok felépítése rövid leírása 18](#_Toc136207932)

[3.2.1. Kliens 18](#_Toc136207933)

[3.2.2. Szerver 18](#_Toc136207934)

[3.2.3. SDK 18](#_Toc136207935)

[3.3. SDK 18](#_Toc136207936)

[3.3.1. Map osztály 19](#_Toc136207937)

[3.3.2. Tile osztály 21](#_Toc136207938)

[3.3.3. Player osztály 21](#_Toc136207939)

[3.3.4. Az SDK osztályainak diagramjai 22](#_Toc136207940)

[3.4. Kliens 23](#_Toc136207941)

[3.4.1. Játékon kívüli kommunikáció 23](#_Toc136207942)

[3.4.2. Játékon belüli kommunikáció 24](#_Toc136207943)

[3.5. Szerver 26](#_Toc136207944)

[3.5.1. Kliens és szerver közti kommunikáció 27](#_Toc136207945)

[3.5.2. A játék logikája 29](#_Toc136207946)

[3.5.3. Leghosszabb kereskedelmi út problémája 31](#_Toc136207947)

[3.6. Tesztelés 31](#_Toc136207948)

[3.6.1. Játékon kívüli tesztek 32](#_Toc136207949)

[3.6.2. Játékon belüli tesztek 35](#_Toc136207950)

[3.7. További fejlesztési lehetőségek 42](#_Toc136207951)

[4. Befejezés 44](#_Toc136207952)

[5. Irodalomjegyzék 45](#_Toc136207953)

# Bevezetés

A mai világban nagyon sokan szeretnek társasjátékozni a gyerekek és a felnőttek körében is. A 2019-ben kitörő COVID-járvány miatt a szociális kapcsolatok megszűntek, mindenki távolságot tartott, bezárkózott otthonába. Ezért sajnos a baráti társaságok, nem egy háztartásban élő rokonok közötti társasjátékos alkalmak száma a pandémia idején gyakorlatilag nullára csökkent. Ebben az időszakban terjedtek el a különböző online felületek, ahol mindenki biztonságosan a saját otthonából játszhatott különböző társasjátékokkal. Ha a játék során valamilyen platformon videóhívásban vannak a résztvevők, akkor az elég jól meg tudja közelíteni a személyes találkozókat.

A szakdolgozatom témájának a Catan társasjátékot választottam. Ez a játék volt az, amely rabul ejtett, bevezetett a modern társasjátékok világába. A Catan társasjátékot Klaus Teuber tervezte, és 1995-ben adta ki először Németországban a Kosmos cég [1]. A világ számos nyelvére lefordították, ezáltal nagyon sok ember által kedvelt játékká vált. Ez bizonyítja az is, hogy több díjat is nyert a játék. Ezek közül a legkiemelkedőbbet rögtön a kiadás évében nyerte el, a társasjáték világ egyik legrangosabb díját, a Spiel des Jahres díjat [2].

A Catan egy olyan stratégiai játék, amely során egy hatszöglapkákból álló lakatlan szigetet próbálnak a játékosok belakni településekkel, városokkal, utakkal oly módon, hogy lehetőleg minél kedvezőbben férjenek hozzá az építkezéshez szükséges nyersanyagokhoz, ezáltal minél több győzelmi pontot szerezzenek. A szakdolgozatban készült program az 5. kiadás hivatalos szabálykönyve [3] alapján készült.

A következőkben először a fejlesztői majd a felhasználói dokumentációt szeretném bemutatni.

# Felhasználói dokumentáció

## Játék ismertetője és szabályzata [3]

A Catan társas játék egy körökre osztott stratégiai játék. A játékosok lakatlan szigetet próbálnak a játékosok belakni településekkel, városokkal, utakkal oly módon, hogy lehetőleg minél kedvezőbben férjenek hozzá az építkezéshez szükséges nyersanyagokhoz, ezáltal minél több győzelmi pontot szerezzenek. A céljuk eléréséhez bónuszkártyákat vásárolhatnak. A játékot az nyeri, aki elsőként összegyűjt 10 pontot.

A játéktábla egy szigetet ábrázol, amin hatszög alakú mezők találhatóak. Minden hexagonnak két fontos tulajdonsága van: a nyersanyag típusa és a száma. A játékban erdő, dombvidék, legelő, szántóföld, hegység és sivatag típusú lapkák szerepelnek. A mezők számai 2 és 12 közötti – kivétel a 7 – egész számok. A 2 és a 12 számok csak egy, a többi szám két mezőn szerepel. Egyetlen mező szerepel a játékban, amelynek nincsen száma, mégpedig a sivatag.

A játékosok a körük elején dobnak két darab 6 oldalú dobókockával. A dobott kockákon lévő számok összegével megegyező számú mezők csúcsaiban található minden település, illetve város termel a mezőknek megfelelő nyersanyagot, amelyet a települések és városok tulajdonosai azonnal megkapnak. A települések esetén egy, a városok esetében pedig kettő nyersanyag termelődik.

A hetes szám egy különleges szám. Ha a kör elején dobott számok összege 7, akkor az aktuális játékos köteles a rablót elmozgatni egy másik mezőre. A mozgatás után a lefedett mező csúcsain található települések és városok tulajdonosai közül az aktuális játékos köteles egyet kiválasztani, akitől véletlenszerűen húz egy nyersanyagkártyát. Ha a választott mezőn nincsen az ellenfelek közül senkinek települése vagy városa, ebben az esetben az aktuális játékos nem húz senkitől nyersanyagkártyát.

A játékosok a körükben a hatszöglapkák csúcsaira tudnak településeket vagy városokat építeni, illetve a hexagonok éleire utakat tenni. Ezen felül bónuszlapokat vásárolhatnak.

### Mezőtípusok

Hat különböző mezőtípus található a játékban.

* Sivatag (Desert): A játék elején itt található a rabló, ezen a mezőn nem termelődik semmilyen típusú nyersanyag.
* Erdő (Forest): Itt fa (Lumber) típusú nyersanyag termelődik.
* Legelő (Pasture): Itt gyapjú (Wool) típusú nyersanyag termelődik.
* Hegy (Mountains): Itt érc (Ore) típusú nyersanyag termelődik.
* Dombvidék (Hills): Itt építőkövek (Bricks) típusú nyersanyag termelődik.
* Szántóföld (Fields): Itt gabona (Grain) típusú nyersanyag termelődik.

A képen tervezés látható

Automatikusan generált leírás alacsony megbízhatósággal

.. ábra Mezőtípusok: sivatag, erdő, legelő, hegy, dombvidék, szántóföld

### A játék kezdete

Miután a táblán a játékmezők megjelentek a játékosoknak le kell tenniük két települést, és ahhoz kapcsolódóan egy-egy utat ingyen a játékossorrendnek megfelelően két körben. Ezután minden játékos megkapja azokat a nyersanyagokat, amiket a településeik érintenek, majd elindul a tényleges játék.

### Egy játékos köre

A játékos a köre elején minden esetben dob a dobókockákkal, a termelt nyersanyagok kiosztásra kerülnek. A körében korlátlanul építkezhet, vásárolhat bónuszkártyákat amíg van elegendő nyersanyaga. Kijátszhat egyetlen bónuszkártyát. Ha nem tud, vagy nem szeretne több akciót végrehajtani, akkor passzol, és a következő játékossal folytatódik a játék.

### Építkezés

A játékosok a saját körükben három különböző típusú terméket építhetnek.

Út

* Ár: 1 fa, 1 építőkő
* Utat csak olyan élre építhetnek a játékosok, amelynek a csúcsában van már egy korábban lehelyezett útjuk, településük vagy városuk.
* Leghosszabb kereskedelmi út: Ha egy játékos elsőként épít 5 egybefüggő utat, elnyeri a leghosszabb kereskedelmi út jutalomkártyát, amely 2 pontot ér. Ha egy másik játékos nála hosszabb útvonalat épít, elveszi tőle díjat. Abban az esetben, ha egy játékos társa megszakítja a leghosszabb utat, abban az esetben a díj birtokosa elveszítheti a kitüntetést.

Település (1 pont)

* Ár: 1 fa, 1 építőkő, 1 gabona, 1 gyapjú
* A települést csak olyan csúcsra lehet lerakni, amelynek a szomszédos csúcsain nincsen sem település, sem város, és a játékosnak fut legalább egy útja a csúcsba. Egy időben legfeljebb 5 darab települése lehet egy játékosnak a pályán.

Város (2 pont)

* Ár: 2 gabona, 3 érc
* Csak meglévő település fejleszthető tovább várossá. Ilyenkor a település lekerül a játéktábláról, és annak helyére kerül a város. Egyszerre legfeljebb 5 darab városa lehet egy játékosnak lent a pályán.

### Bónuszkártyák

A játékosok dönthetnek úgy is, hogy bónuszkártyát vásárolnak, amelynek ára 1 gabona, 1 érc és 1 gyapjú. A lefordított bónuszkártya pakliból egy kártyát húz a játékos, amelyet titokban tart ellenfelei előtt a kijátszás pillanatáig. Az aktuális körben húzott bónuszkártya nem játszható ki. Minden körben legfeljebb egy bónusz kártya játszható ki, kivétel a pontkártyák.

Öt különböző típusú bónusz kártya van:

* Lovagkártya (Knight): Amikor a játékos kijátssza ezt a kártyát áthelyezi a rablót, csak úgy mintha 7-est dobott volna, a rabló áthelyezési szabályokat betartva. Ha egy játékot elsőként játszik ki három lovagkártyát, akkor elnyeri a Legnagyobb lovagi hatalom jutalomkártyát, amely 2 pontot ér. Ha egy ellenfele nála több lovagkártyát játszik ki, akkor azonnal át kell adni neki a díjat.
* Bónuszkártyák:
  + Találmány (Year of Plenty): Amikor a játékos kijátssza ezt a kártyát, kiválaszt két tetszőleges nyersanyagot, amit magához vesz a bankból.
  + Útépítés (Road building): Amikor a játékos kijátssza ezt a kártyát, letehet 2 utat az út lerakási szabályoknak megfelelően.
  + Monopólium (Monopoly): Amikor a játékos kijátssza ezt a kártyát, kiválaszt egy nyersanyag típust, és az ellenfelei kötelesek az összes a megjelölt nyersanyagtípussal megegyező a kezükben tartott lapot átadni a játékosnak.
* Pontkártyák: A pont kártyákat csak akkor lehet kijátszani, ha ezzel a játékos megnyerné a játékot. Pontkártyából akárhány kijátszható a játékos utolsó körében.

### Rabló mozgatása

Ha a játékos 7-est dobott, akkor köteles elmozgatni a rablót a jelenlegi helyéről egy másik mezőre. A rablót a mező számára kell helyezni, ezzel letakarva a számát. Ezzel jelzi, hogy ha ezt a számot dobják a játékosok, ez a mező nem termel nyersanyagot. Ha a mozgatott rabló mezőjén van település vagy város, akkor az aktív játékos véletlenszerű elvesz egy nyersanyagot a település/város tulajdonosától. Ha több játékosnak van települése vagy városa a rabolt mező körül, akkor az aktív játékos választhat, hogy kitől szeretne rabolni. Ha nincsen ellenfélnek települése vagy városa a mező körül, az aktuális játékos nem jut nyersanyaghoz.

### Passzolás

Ha egy játékos nem tud, vagy nem akar több akciót végrehajtani, akkor a Pass gomb megnyomásával elpasszolhatja a körét, és a soron következő játékos folytatja a játékot.

### Játék vége

A játék azonnal véget ér, ha az aktuális játékos eléri a 10 pontot.

## A játék futtatásához szükséges gépigény, programok

A programok futtatásához Java 17 szükséges, a szervernek szükséges még egy PostgreSQL adatbázis.

## A játékfelület áttekintése

A program több játékos számára biztosít lehetőséget a Catan társasjáték játszárára. A program felhasználói felületén a feliratok angol nyelvűek. Ezen felhasználói dokumentációban az elnevezések magyar és angol nyelven is feltűntetésre kerülnek.

A program kezdőoldalán tudnak a játékosok regisztrálni vagy bejelentkezni. A bejelentkezett felhasználók számára az alábbi lehetőségek válnak elérhetővé:

* várakozó szoba készítése,
* rendelkezésre álló várakozó szobák kilistázása,
* elérhető várakozó szobához való csatlakozás,
* kilépes a jelenlegi várakozó szobából.

Amint egy várakozó szoba adminisztrátora elindítja a játékot, az szoba eltűnik a rendelkezésre álló szobák listájából, és a csatlakozott felhasználók számára elindul a játék.

## A játék kezelőfelülete

### Regisztráció és bejelentkezés

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás

.. ábra Regisztráció és bejelentkezés

A Register gomb megnyomásával a játékosok új felhasználói fiókot tudnak létrehozni. Sikeres regisztráció után azonnal átnavigál a program a bejelentkező felületre, és a fiókjuk bekerül az adatbázisba. Az adatbázis tárolja, hogy hány játékot játszott, és ezek közül mennyi játékot nyert meg. A regisztráció esetén fontos megjegyezni, hogy kis- és nagybetűérzékeny mind a felhasználónév, mind pedig a jelszó.

A Login gombra kattintva a felhasználók a már létező fiókjukba tudnak bejelentkezni. Mindkét felületen a felhasználónév és a jelszó megadása kötelező. Ha a felhasználó olyan adatpárt ad meg, amely nem létezik az adatbázisban, megfrissül a felület, és újra próbálkozhat a felhasználó.

### Várótermek listája

Ezen a felületen lehet létrehozni várakozó szobákat, azokhoz csatlakozni, azokból kilépni, illetve megtekinthető a saját vagy más játékos statisztikája.

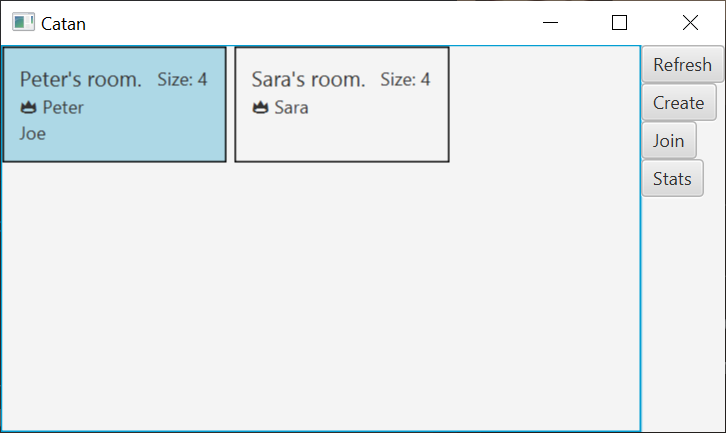
A Refresh gombbal megfrissíthető az elérhető szobák felülete. Minden várakozó szobáról látszik, hogy ki az szoba adminisztrátora (korona jelenik meg a neve előtt), illetve az is, hogy mely játékosok csatlakoztak már a szobához. Amennyiben az egyik szobához már 4 játékos csatlakozott, abban az esetben új játékost már nem enged meg a program belépni, hiszen a Catan alapjátékot 3-4 játékosra tervezték.

A képen képernyőkép, szöveg, számítógép, szoftver látható

Automatikusan generált leírás

.. ábra Nincs elérhető várakozó szoba

Az 2.4. ábrán az első szoba adminisztrátora Peter, és Joe már csatlakozott a játékhoz. Sara is készített egy játékot, de az ő szobájába még senki sem lépett be.



.. ábra Elérhető várakozó szobák

Ha egy szobára rákattintunk, azaz kijelöljük, akkor azt a felület kék háttérrel kiemeli. Ha rányomunk a Join gombra, akkor pedig megpróbálunk belépni a szobába.

Ha egy játékos csatlakozott egy szobához, látja, hogy rajta kívül kik várnak a szobába, illetve, hogy ki az adminisztrátora a szobának. Amennyiben nem szeretne ebben a játékban részt venni, a Leave gombra kattintva el tudja hagyni a szobát.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

.. ábra Joe csatlakozott Peter szobájához.

Az szoba készítője (adminisztrátora) a Start gomb megnyomásával tudja elindítani a játékot.

### Statisztika

A statisztika nézeten lekérhető a regisztrált játékosok statisztikája, amely a Stats gomb megnyomásával érhető el. Egy beviteli mezőben meg kell adni annak a felhasználónak a nevét, akinek a statisztikájára kíváncsiak vagyunk. Az adatbázis minden játékosról tárolja, hogy eddig hány játékot játszott, illetve, hogy ezek közül mennyit nyert meg. Ha olyan felhasználó nevet adunk meg, ami nem szerepel az adatbázisban, akkor a „Get Stats” gomb megnyomája után „Username not found!” felirat jelenik meg, azaz „A felhasználónév nem található!” üzenetet kapjuk. Ha olyan felhasználónevet adtunk meg, amelyről tárolunk adatokat az adatbázisban, akkor a „Get Stats” gomb megjeleníti a mentett információkat.

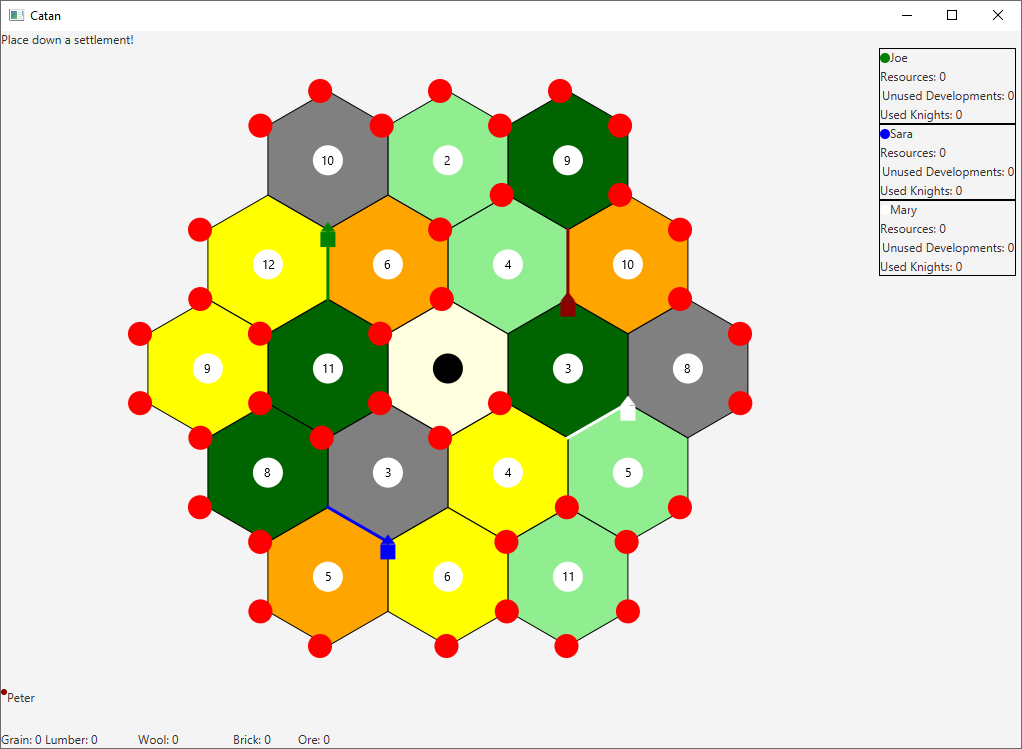
A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

2.6. ábra Statisztika ablak létező és nem létező felhasználó esetén

### Játék kezdete

Miután a szoba tulajdonosa elindítja a játékot, minden felhasználó ablakában megjelenik a játéktábla. A tábla mezőinek elrendezése a játékszabályban ajánlott módon történik. A kezdőjátékos a szoba adminisztrátora, utána csatlakozási sorrendben következnek a játékosok.



.. ábra Peter lehelyezi a második települését (settlement).

Az ablak bal felső sarkában látható mindig az aktuális játékos számára, hogy mit kell tennie a körében. Jobb oldalon láthatóak a többi játékosról publikus információk: összesen mennyi nyersanyagkártyája (Resources), fel nem használt bónuszkártyák (Unused developments) és kijátszott lovagkártyája (Knights) van. Itt lehet látni azt is, ha valamelyik másik játékos már elnyerte a Lovagi hatalom, vagy a Leghosszabb kereskedelmi út díjakat. A saját információit minden játékos a képernyő alján látja.

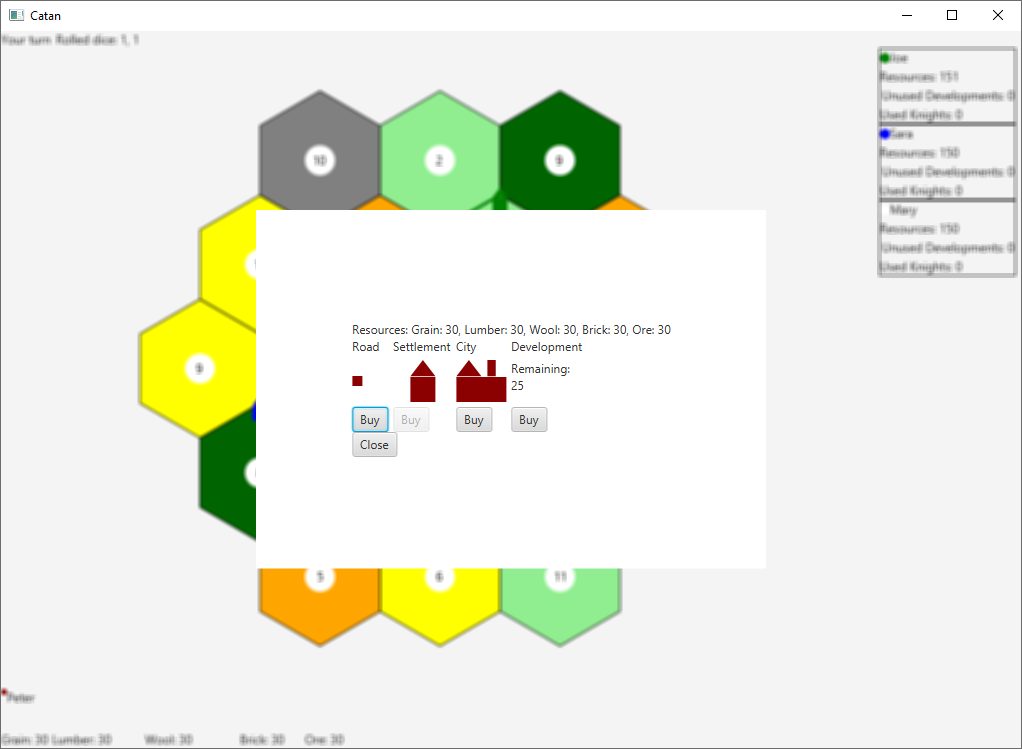
A játék elején minden játékos a játékossorrendnek megfelelően lehelyezi a két települését (settlement) és a hozzájuk kapcsolódó két utat (road). A fenti képen látható módon piros körök jelzik azokat a helyeket, ahova az aktuális játékos teheti a települését.

### Egy játékos köre

Miután minden játékos elhelyezte mindkét települését és útját, elkezdődik a játék. A kör elején automatikusan dob a játék a dobókockával, melynek eredménye a bal felső sarokban látható minden játékos számára. A nyersanyagok a dobás függvényében azonnal kiosztásra kerülnek.

A képernyő alján megjelenő gombokkal tudja az aktív játékos végrehajtani a lépéseit. A Buy gomb megnyomásával vásárolhat utat, települést, bónuszkártyát vagy fejlesztheti fel a települését várossá. A vásárló ablak felső sorában látható, hogy mennyi elérhető nyersanyaga van a játékosnak. Ha valamilyen terméket nem tud megvásárolni a játokos, mert nem tudja lehelyezni, vagy nincs elegendő nyersanyaga rá, akkor a vásárlás gomb nem aktív. Ha út, település vagy város vásárlását választotta a játékos, de mégsem szeretné megtenni, akkor a képernyő nem kiemelt részeire kattintva visszavonhatja a vásárlást.

Ha meggondolta magát a játékos, mégsem szeretne vásárolni, akkor a Close gombbal becsukhatja az ablakot.



.. ábra Vásárlás ablak

Ha a játékos úgy határoz, hogy a települését várossá bővíti, akkor a vásárlás ablakban a város alatti Buy gomb megnyomása után a pályán választhatja ki a fejlesztendő települést. A választható településeket szürke kör jelzi.

A képen képernyőkép, diagram, pixel látható

Automatikusan generált leírás

.. ábra Település fejlesztése várossá.

### Bónuszkártyák

A megvásárolt bónuszkártyákat a Developments gombbal érhetjük el. Minden típusnál jelzi a program, hogy hány jelenleg használható kártyája (Usable) van a játékosnak, hány használható fel később (Has to wait)[[1]](#footnote-1), illetve hányat használt (Used) eddig a játékos.

A képen szöveg, képernyőkép, diagram, szoftver látható

Automatikusan generált leírás

.. ábra Bónusz kártya ablak

A fenti ábrán látható, hogy az aktuális játékosnak van 5 kijátszható lovagkártyája, egyet húzott ebben a fordulóban, és egyet már felhasznált korábbi körökben. Van egy kijátszható Year of Plenty kártyája is, illetve van két pontkártyája, amely a játék végéig rejtve maradnak a többi játékos elől. Amint egy bónuszkártyát kijátszik, minden Use gombok inaktívvá válnak ebben az ablakban.

### Rabló

Ha egy játékos a köre elején hetest dob, a rablót köteles elmozdítani. Ehhez a kívánt mezőre kell kattintani. Ha olyan mezőre lépteti a rablót, amely körül több játékosnak is van települése vagy városa, egy felugró ablakban kiválaszthatja az aktív játékos, hogy kitől kíván véletlenszerűen egy nyersanyagkártyát ellopni. Ha csak egy játékos található a mezőn, akkor attól a játékostól automatikusan lop egy nyersanyagot.

A képen szöveg, képernyőkép, diagram, szoftver látható

Automatikusan generált leírás

.. ábra Rablás célpontjának kiválasztása.

### Tengeri kereskedelem

Ha egy játékosnak nincsen elegendő nyersanyaga, akkor a bankkal illetve a játékostársaival is kereskedhet. A Maritime trade gomb megnyomásával kiválaszthatja, hogy melyik nyersanyagból cserél be négyet egy általa választott másik típusú nyersanyagkártyára. A bal oldali legördülő menüből ki kell választani, miből ad be a játékos négyet a bankba, majd a másodikban meg kell adni mit kér cserébe. A tranzakció a Four to One gomb megnyomásával megy végbe.

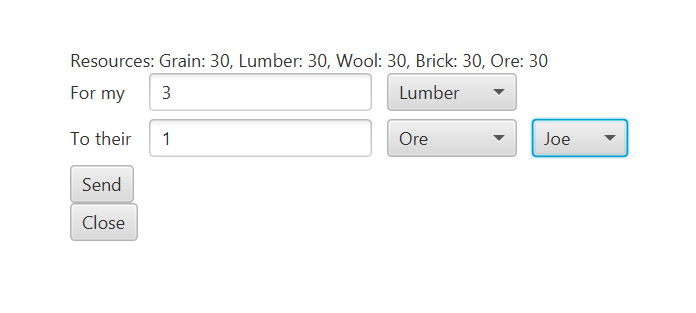
A képen képernyőkép, szöveg, diagram, szoftver látható

Automatikusan generált leírás

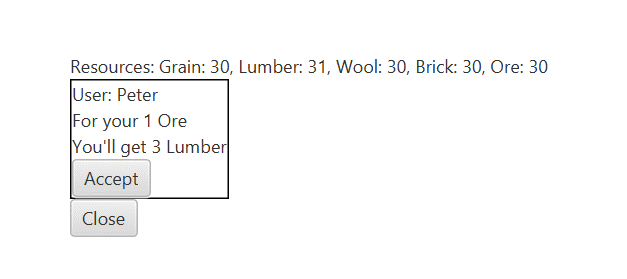
.. ábra Kereskedés a bankkal.

### Kereskedés a játékosok között

Az aktuális játékos kezdeményezhet cserét egy tőle különböző játékossal a „Player Trade” gomb megnyomásával. Felajánl egy bizonyos nyersanyagból valahány darabot (For my), és attól különböző nyersanyag valahány darabjáért (To their) Az ajánlat személyre szóló. Egy körön belül korlátlan számú ajánlat tehető. A csereajánlattal megkeresett játékos a saját körén kívül elfogadhatja az ajánlatokat. A felhasználók számára Player Trade gomb jelenik meg, ha kereskedelmi ajánlatot kapnak. Az „Accept” gombra kattintva elfogadhatja az ajánlatot. Ekkor a tranzakció azonnal végbe megy. Ha nem tetszik neki a felajánlott nyersanyag, akkor a Close gombbal elutasítható a csere.



2.13. ábra Aktuális játékos felajánl Joe-nak 3 fát egy ércért.



.. ábra Joe eldöntheti, hogy elfogadja-e Peter ajánlatát.

### Játék vége

A játék azonnal véget ér, ha az aktuális játékos eléri a 10 pontot. Egy felugró ablak jelzi az összes résztvevő számára, hogy a játékot ki nyerte, és visszatérhetnek a váróterembe.

A képen képernyőkép, szöveg, diagram, Grafikai szoftver látható

Automatikusan generált leírás

.. ábra Játék vége

# Fejlesztői dokumentáció

Ez a fejezet a szakdolgozat programjait fogja részletezni fejlesztői szemszögből, hogy megértesse a működését.

## Felhasznált technológiák a fejlesztéshez

### IntelliJ IDEA

Az IntelliJ IDEA [4] a JetBrains által készített IDE[[2]](#footnote-2) a Java és Kotlin nyelven programozók számára. Specifikusan JVM[[3]](#footnote-3) alapú nyelvekhez lett fejlesztve. Képes használni Gradle-t és Mavent is, illetve rendelkezésre áll egy beépített debugger is.

### Kotlin

A Kotlin [5] egy JVM alapú, statikusan erősen típusos, magas szintű, JetBrains által fejlesztett programozási nyelv. Egyik nagy előnye, hogy egyszerre lehet használni Kotlint és Javat is ugyan abban a kódbázisban, ezért lehetséges az, hogy egy meglévő kódbázis fokozatosan áll át Javaról Kotlinra.

### Ktor

A Ktor [6] az egy aszinkron alapú kommunikációs könyvtár, ami képes http-t készíteni/kiszolgálni, vagy akár Socketeken, WebSocketeken keresztül kommunikálni. A Ktort az elejétől kezdve úgy fejlesztették, hogy Kotlinnal legyen használva. Egyszerűen lehet hozzá készíteni egyedi plugin-okat. Ktor segítségével oldottam meg a kliens és szerver közti socket kommunikációt.

### Kotest

A Kotest [7] egy teszt keretrendszer Kotlinban írt programok tesztelésére készült. Többféle tesztelési stílust támogat, GIVEN WHEN THEN stílusú teszteket lehet vele készíteni, vagy csak sima, egyszerű funkciót futtatni.

### Gson

A Gson [8] egy nyílt forráskódú Java könyvtár, amellyel Java Objecteket lehet Jsonnel reprezentálni, és Json által reprezentált objektumot lehet Java Objectté alakítani. Ezt a könyvtárt használtam arra, hogy a kliens és a szerver tudjon egymással kommunikálni.

### Hibernate

A Hibernate [9] egy ORM[[4]](#footnote-4) eszköz, ami képes arra, hogy Java objectet egy adatbázisban lévő elemhez rendel. Képes adatbázisba menteni, onnan kiolvasni, és már tárolt elemen változtatni.

### TornadoFX

A TornadoFX [10] egy Kotlin wrapper a Java FX-hez, ami megkönnyíti a Kotlinban fejlesztett ablakos alkalmazások fejlesztését.

### Gradle

A Gradle [11] egy automatizáló eszköz, ami nagyban segíti a Java/Kotlin kódbázis fordítását, külső könyvtárak használatát. Lehet benne egyedi feladatokat csinálni, amik megtudják könnyíteni a fejlesztési folyamatot.

## Áttekintés, programok felépítése rövid leírása

A projekt több különböző alkalmazásból áll, mindegyik saját, specifikus feladattal rendelkezik.

### Kliens

Egy grafikus felület, ami megjeleníti a szervertől kapott információkat a képernyőn. A felhasználók ezen keresztül kommunikálnak a szerverrel.

### Szerver

Kiszolgálja a klienst vagy klienseket a tőlük kapott információkkal. A szerverhez egyszerre több kliens is csatlakozhat.

### SDK[[5]](#footnote-5)

A kliens és szerver közös komponenseit tárolja. Megkönnyíti a fejlesztést olyan szempontból, hogy a közös komponenseket csak egyetlen helyen kell megváltoztatni.

## SDK

Az SDK-nak a feladata, hogy a közös komponenseket tárolja. Kotlinban lehetőség van arra, hogy olyan osztályokat használjunk fel, amelyek másik projektekből származnak. Így nincs szükség ugyanazt az osztályt több különböző helyen deklarálni, ezzel csökkentve a kódbázis méretét, és így nem keletkezhet olyan hiba, hogy a kliens és a szerver máshogy implementálja ugyanazokat az osztályokat. Ha egy osztályt módosítani szükséges, vagy új osztályt kell létre hozni, akkor azt csak egy projektben kell megtenni. Például a Dto[[6]](#footnote-6) osztályok, amelyek a kliens és a szerver közötti kommunikációt könnyítik meg.

### Map osztály

A Catan társas játékban a pálya hatszögekből áll, ezért nem lehet egyszerűen ábrázolni. A Map osztály felel azért, hogy a generált pályán a mezők (Tile-ok) megfelelő helyen jelenjenek meg egymáshoz képest. Ezeket a Tile-okat összekapcsolja, a Tile-oknak mind a 6 csúcsára rak egy-egy Vertex osztályt, és ezeket a Vertexeket összeköti Edge osztállyal. Lehetőség van arra, hogy egy előre elkészített pályát generáljon le, ahol a Tile-okhoz előre hozzá van rendelve azok száma és a nyersanyagtípusa.

Map osztályt elő lehet állítani egy MapDto osztályból is, ami tartalmazza tömören a Tile‑okat, és ugyanazt a pályát állítja elő, mint amit a szerver lát.

#### Map generálása

A pálya generálása két lépésből áll:

1. mezők generálása,
2. mezők és csúcsok összekötése.

Mezők generálásához le kell generálni 19 Tile-t a Catan alapjáték pályájának felépítéséhez. A Tile-nak az id-ja két részből áll: az elején egy „T” prefix, majd utána a suffix megadja a Tile generált sorszámát nullától kezdve. Például az első Tile „T0”, második „T1”.

Majd a második lépésben ezeket a Tile-okat összekötjük. Ezt úgy teszi meg a program, hogy végig iterál a tiles listán, és a jelenlegi Tilehoz hozzá köti a tőle jobbra, jobbra lefele és balra lefele lévő Tile-okat. Ezeket a Tile-okat úgy kapjuk meg, hogy a jelenlegi Tile id-ját átalakítjuk Coordinate-té és „getCoordBottomRight / getCoordBottomLeft / getCoordRight” metódusok meghívásával megkapjuk a szükséges Tile-ok relatív koordinátáit. A kapott koordinátát átváltjuk toTile() segítségével Tile-lá, és az attachTileToSide segítségével össze lehet azokat kötni.

#### ***A Map osztály függvényei***

|  |  |
| --- | --- |
| Függvény név és leírása | Működése |
| generateTiles(numberOfRows: Int, isGeneric: Boolean) | Legenerálja az összes Tile-t. Ha az isGeneric true, akkor beállít egy előre készített pályát. Pálya legelső és utolsó sora minden esetben 3 nagyságú. |
| attachAllTiles() | Az összes Tile-t ami ebben a Map-ben található össze kapcsolja az oldalai mentén. |
| addEdges() | A tile-okon található vertexeket összekötjük egymással. |
| toDto(): MapDto | Összeállít egy olyan MapDto-t amiből össze lehet állítani ugyanezt a Map objektumot. |
| attachTileToSide(  attachToThis: Tile,  bottomRight: Tile,  right: Tile,  topRight: Tile,  topLeft: Tile,  left: Tile,  bottomLeft: Tile,  ) | Az attachToThis-hez oda köti a Tile-okat, olyan irányból amilyen irányból megadjuk. Azok a paraméterek, amelyek null-ok, ott egy új Vertex keletkezik, amely csak az attachToThis-hez csatlakozik. |
| idToCordinate(id: Int) | A tile ID-jából összeállít egy koordinátát, ami egy számpárból áll: a pár első eleme a sorszámra, a második eleme a sorban lévő helyére vonatkozik. |
| getCoordBottomRight(cord: Coordinate): Coordinate | Visszaadja a „cord”-hoz képest a jobbra lent lévő Vertex koordinátáját. |
| getCoordBottomLeft(cord: Coordinate) : Coordinate | Visszaadja a „cord”-hoz képest a balra lent lévő Vertex koordinátáit. |
| getCoordRight(cord: Coordinate) : Coordinate | Visszaadja a „cord”-hoz képest a jobbra lévő Vertex koordinátáit. |

### Tile osztály

Ez az osztály reprezentálja a Catan mezőit. A Catan táblájának minden mezője hatszög alakú. Az osztályban van egy vertices lista, ami 6 Vertex elemet tartalmaz. A 3.3.3. ábrán látható, hogy a lista mely indexű elemei melyik csúcsot reprezentálják.

A képen diagram, tervezés látható

Automatikusan generált leírás

3.3.3. ábra A Tile osztály csúcsai grafikusan reprezentálva.

#### ***Függvények***

|  |  |
| --- | --- |
| Függvény név és leírása | Működése |
| connectVertices(edges: List<Edge>) | Összeköti a Vertexeket a fenti ábra alapján, majd a keletkezett éleket a paraméterként megadott listába helyezi el. |

### Player osztály

Ez az osztály felel egy aktív játékban lévő játékos reprezentálására.

#### ***Függvények***

|  |  |
| --- | --- |
| Függvény név és leírása | Működése |
| calculateMyPoints(): Int | Kiszámolja a játékos pontjait, beleszámítva a ki nem játszott pontkártyákat. |
| toDto(hidden: Boolean): PlayerDto | Átalakítja a Player osztályt egy PlayerDto osztállyá, amit a szerver tovább tud küldeni a klienseknek. Ha a hidden paraméter true, akkor olyan dto-t ad vissza, amin a játékos kártyái és nyersanyagai nem láthatóak, csak a darabszámukat tartalmazza. Ha a hidden paraméter false értékű, akkor minden információja elérhető a kártyáknak. |
| refreshFromDto(dto: PlayerDto) | A paraméterül kapott dto alapján felfrissíti a játékos adatait. |

### Az SDK osztályainak diagramjai

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás

.. ábra Az SDK-ban a fontosabb osztályok diagramjai.

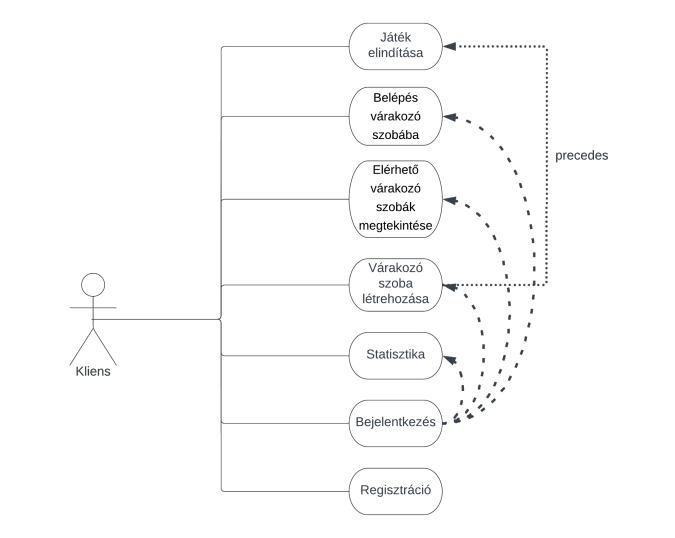
## Kliens

A Kliens egy grafikus felület, amely a szervertől kapott információkat jeleníti meg a felhasználó számára, és lehetőséget biztosít arra, hogy ezen keresztül kommunikáljon a szerverrel.

A Kliensnek a szerver irányába lévő kommunikációt két részre lehet bontani, játékon kívüli és játékon belüli kommunikációra.

### Játékon kívüli kommunikáció

A játékon kívüli kommunikációk legtöbbször olyan formában történnek, hogy a kliens küld egy üzenetet, és erre a szerver válaszol a küldő kliensnek, és nem feltétlen jelez a többi kliensnek.



.. ábra A kliens játékon kívüli tevékenységei.

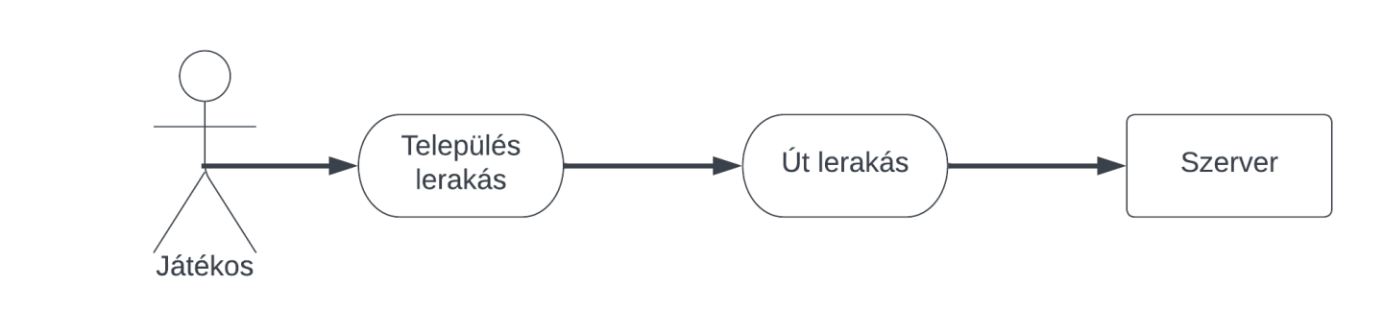
A program futtatása során a kliens az alábbi esetekben kommunikál a szerverrel:

* Regisztráció: A felhasználó készít egy új felhasználói fiókot, amivel be tud a későbbiekben jelentkezni.
* Bejelentkezés: A felhasználó bejelentkezik egy már regisztrált felhasználói fiókkal.
* Statisztika: A felhasználó lekéri egy játékos (akár saját maga) tárolt statisztikai adatait.
* Várakozó szoba létrehozása: Egy várakozó szobát hoz létre a felhasználó, amelyhez a többi játékos csatlakozhat.
* Elérhető várakozó szobák megtekintése: Lekérhető a szervertől a jelenleg elérhető várakózó szobák.
* Belépés várakozó szobába: Egy elérhető várakozó szobába csatlakozhat a felhasználó.
* Játék elindítása: A várakozóban, a szoba adminisztrátora elindíthatja a játékot, és a szobában lévő játékosok átkerülnek a játék felületére.

### Játékon belüli kommunikáció

A játékon belüli kommunikáció során, amint az egy játékos végrehajt egy akciót, arról a többi játékos kliensét is mindig értesíteni kell, és a státuszváltozás függvényében a felületen meg kell jeleníteni az új információkat. A játék két részre osztható: mielőtt minden játékos lerakta a maga kezdő faluit és útjait, illetve miután ténylegesen elkezdődött a játék.

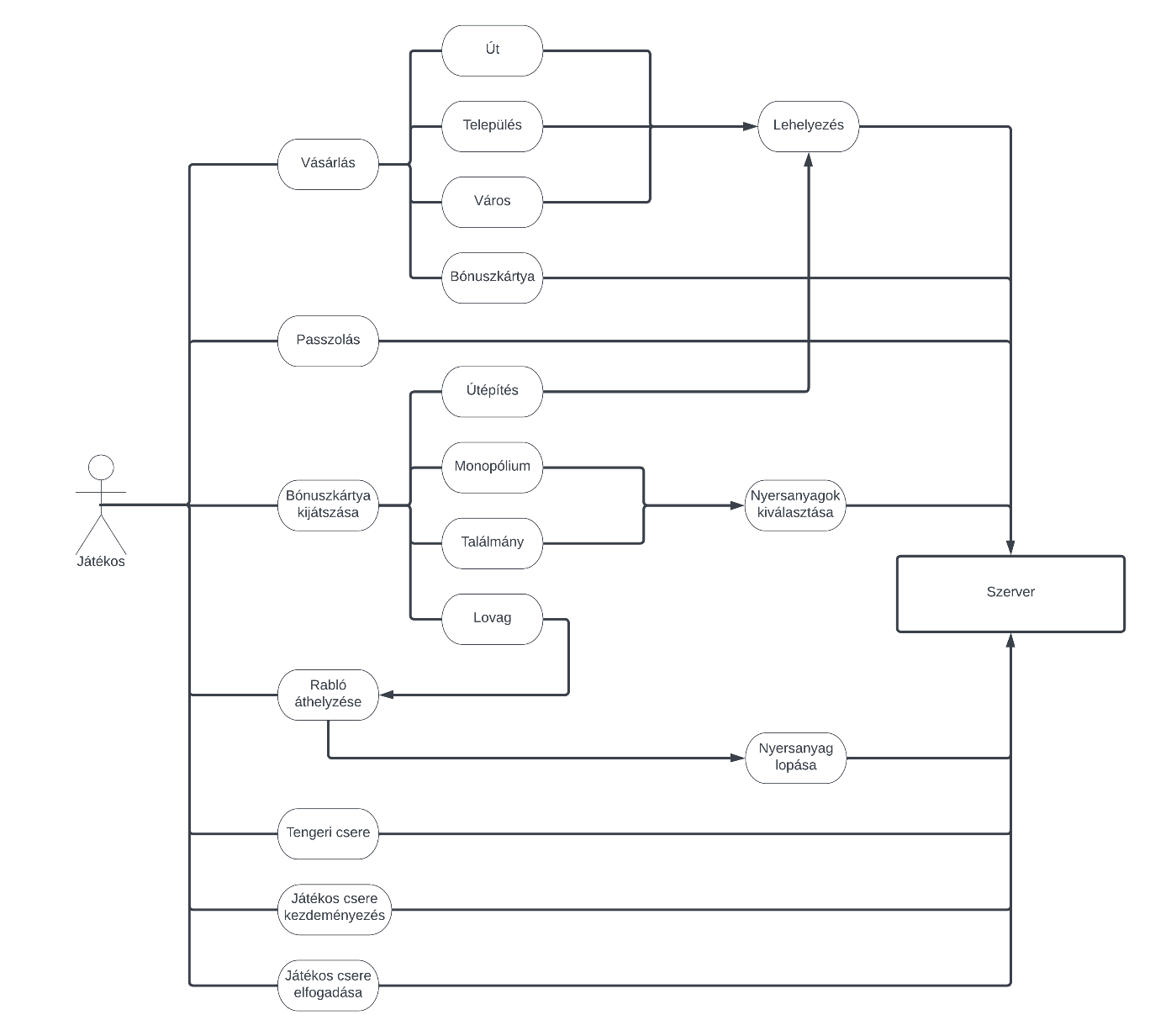
#### Mielőtt mindenki mindent lerakott



.. ábra A játék kezdetének folyamata.

Játék elején a felhasználónak ki kell választania egy csúcsot, ahova leteszi az első települését, és egy élt, ami ehhez a csúcshoz kapcsolódik. Majd ezt a kiválasztott csúcs-él párt küldi tovább a kliens a szervernek. Majd ezt a folyamatot még egyszer megismétli minden játékos. Ha minden játékos letette mindkét települését és útját, azután elindul ténylegesen a játék.

#### Miután a játék ténylegesen elkezdődött



.. ábra A játékos lehetséges döntései

Miután elindult a játék ténylegesen a játékos 7 különböző dolgot tud csinálni:

* Passzolás: Elküldi a szervernek, hogy végzett a játékos a körével.
* Bónuszkártya kijátszása: Előfeltétele mindegyiknek, hogy a játékos rendelkezik az adott kártyával, és azt egy korábbi körben vásárolta.
  + Útépítés: A játékos az útépítés szabályait betartva elhelyez legfeljebb két utat.
  + Monopólium: A játékos kiválaszt egy nyersanyag típust, és a többi játékos köteles átadni minden vele egyező nyersanyagkártyáját.
  + Találmány: A játékos kiválaszt két tetszőleges nyersanyagot, amelyet a banktól megkap.
  + Lovag: a játékos áthelyezi a rablót a rabló áthelyezés szabályait betartva.
* Rabló áthelyezés: a játékos elmozgatja a rablót a jelenlegi mezőjéről, és ha az új mezőn van más játékosnak települése vagy városa, akkor tőle elrabol véletlenszerűen egy nyersanyagot. Amennyiben több játékos is eleget tesz a feltételnek, az aktuális játékos dönt, kitől szeretnek nyersanyagot lopni.
* Vásárlás: Előfeltétele mindegyiknek, hogy a megfelelő nyersanyagokkal rendelkezik a játékos. Az akció után a szükséges nyersanyagok levonásra kerülnek.
  + Út: Ha olyan az él, aminek legalább az egyik csúcspontjának az egyik élét birtokolja a jelenlegi játékos, akkor letesz egy utat az élre.
  + Település: Ha olyan a csúcs, aminek a szomszédos csúcsaiban nincs település vagy város, és annak a csúcsnak legalább az egyik élét birtokolja a játékos, akkor oda letesz egy települést. Legfeljebb 5 települése lehet minden játékosnak egyidőben a pályán.
  + Város: Ha van lent települése a játékosnak, akkor azt felfejleszti várossá. Legfeljebb 5 városa lehet minden játékosnak egyidőben a pályán.
  + Bónuszkártya: Ha van még bónuszkártya a pakliban, amit még nem húztak fel, akkor vásárol egyet.
* Tengeri csere: Játékos kiválaszthatja, hogy melyik nyersanyagból cserél be négyet egy általa választott másik típusú nyersanyagkártyára.
* Játékos csere kezdeményezés: Az aktuális játékos felajánl tetszőleges mennyiségű azonos típusú nyersanyagot egy másik játékos számára, amelyért cserébe megjelöli, hogy milyen nyersanyagból mennyit szeretne kérni.
* Csere elfogadása: Elfogadhatja azt a cserét, amit a jelenlegi játékos kezdeményezett vele. A fenti akciók közül ez az egyetlen, amelyet nem a saját körében hajthat végre a játékos.

## Szerver

A szerver feladata, hogy a klienseket kiszolgálja a tőlük kapott információk alapján. A kliens és szerver közti kommunikáció Socketeken keresztül történik TCP[[7]](#footnote-7) segítségével.

TCP a Catan esetében sokkal jobb, mint az UDP[[8]](#footnote-8), mivel relatív kicsi a forgalom a program futtatása során, és sokkal fontosabb, hogy minden adat jó sorrendben és biztosan megérkezzen, mint hogy a késleltetés az adatcsomagok között alacsony legyen.

### Kliens és szerver közti kommunikáció

A kliens és a szerver egymással Json-ökkel kommunikálnak, amik egy dto-t reprezentálnak. Attól függően, hogy a kliensek milyen információt küldenek az alapján dől el, hogy éppen hogyan lesz az üzenet interpretálva.

Egy példa az üzenetre, amit elvárunk a klienstől:

{"username":"Joe","password":"SecretPassword","type":"LOGIN"}

A SocketService az üzenet „type” attribútuma alapján dönti el, hogy milyen típusú üzenet érkezett a klienstől, és ez alapján továbbítja a megfelelő komponensnek.

#### LoginService

A LoginService-hez akkor kerül a kérés, ha a típus LOGIN. Ebben az esetben megnézi, hogy van-e ezzel a felhasználónév-jelszó kombinációval felhasználó az adatbázisban. Ha nincs, akkor hibát dob. Ha van, akkor visszaad a kliensnek egy sessionId-t és azt, hogy sikeresen bejelentkezett. A sessionId alapján állapítható meg, hogy egy adott felhasználó be van-e jelentkezve, és hogy melyik Socket kapcsolható hozzá.

#### RegisterService

A RegisterService-hez akkor kerül a kérés, ha a típus REGISTER. Ebben az esetben, leellenőrzi, hogy a felhasználónév, amit kapott létezik-e az adatbázisban, és hogy a jelszó megfelel-e formai követelményeknek. Ha nem felel meg, hibád dob. Ha megfelel, akkor pedig elmenti a felhasználót az adatbázisba, és a kliensnek jelzi, hogy a regisztráció sikeres volt.

#### LobbyService

A LobbyService-hez akkor kerül a kérés, ha a típus LOBBY. Ebben az esetben, ezeknek a üzeneteknek van commandType attribútumuk, amelyek jelzik, hogy a felhasználó éppen mit szeretne csinálni.

***CREATE típus***

A felhasználó létre szeretne hozni egy szobát. Létrehozunk a szerveren egy szobát, majd a szoba információit visszaküldjük a felhasználónak. A keletkezett szoba bekerül egy közös listába.

***JOIN típus***

A felhasználó beszeretne lépni egy szobába. Megnézzük, hogy létezik-e az a szoba azonosítója, amibe csatlakozni kíván. Ha nem létezik, hibát dobunk, ha létezik, akkor hozzáadjuk a szobához, és elküldjük az összes szobában lévő embernek, a frissült szoba információit.

REFRESH típus

A felhasználó frissíteni szeretné a szobákat, hogy lássa milyen szobák érhetőek el. Elküldjük neki az elérhető szobák információit.

LEAVE típus

A felhasználó ki szeretne lépni abból a szobából, amibe korábban csatlakozott. Ha bent van egy szobában, akkor kivesszük onnan, és jelezzük ezt a többi várakozó felhasználónak is. Ha nincs benne egy szobában sem, akkor nem történik semmi.

START típus

A felhasználó elszeretné indítani azt a játékot, amiben éppen tartózkodik, ha a szoba nem létezik vagy nem ő a szoba tulajdonosa, akkor nem indul el a játék. Egyébként pedig elindítjuk a játékot a GameService segítségével.

#### GameService

A GameService-nek két fontos feladata van: az első, hogy készítsen egy új Game objektumot a LobbyService-től kapott információk alapján és jelezze azt a várószobában lévő klienseknek, hogy elindult a játék. A Game osztály felel a játék logikájának megvalósításáért.

A második, hogy kezelje a felhasználóktól érkezett GAME típusú üzeneteket, továbbítsa a megfelelő Game objektumnak. Ha nem létezik olyan játék, amiben éppen részt vesz a játékos akkor hibát dobunk. Egyébként pedig a Game objektummal lekezeljük a kérést.

Ha egy kérés nem az aktuális játékostól érkezett, a GameService leellenőrzi, hogy az csere elfogadása akció-e, és ha nem, akkor a kérést eldobja.

#### StatisticsService

A StatisticsService-hez kerül a kérés, ha a kérés típusa STATS. Ebben az esetben, a kérésben található felhasználónevet megkeresi az adatbázisban, ha létezik a felhasználó elküldjük a statisztikáit, ha nem akkor pedig elküldjük, hogy nem találtuk meg a felhasználót.

#### DatabaseService

A DatabaseService felel azért, hogy a felhasználók információit helyesen mentse le, kérje le, és módosítsa az adatbázisban.

#### SessionService

A SessionService felel azért, hogy a Kliensnek a socket-jét össze tudjuk kötni felhasználónévvel és sessionId-val, hogy később, tudjunk jelezni a kliensnek, amikor szeretnénk.

### A játék logikája

A játék logikáját egy Game osztály valósítja meg. Ez felel azért, hogy a játékosok által kapott információk alapján a játékban minden a szabályoknak megfelelően történjen, és a végrehajtott akciókról az összes játékos kliense értesüljön.

Ha a kliens egy bónuszkártyát játszik ki, akkor a kártyának a used tulajdonságát true-ra állítja a szerver, és értesíti az összes klienst erről. Ha a kijátszott kártya lovagkártya, akkor megnézi, hogy a lovagi hatalom kártya tulajdonost vált-e.

Ha egy új település vagy út épül, akkor minden esetben meg kell nézni, hogy a leghosszabb kereskedelmi út díj máshoz kerül-e.

#### A Game osztály publikus függvényei

| Függvény név és leírása | Működése |
| --- | --- |
| placeAtBeginning(dto: PlaceBeginningDto) | A dto-ban szereplő Edge és Vertex id-ját leellenőrzi, hogy szabad-e, és a szabályoknak megfelelően lerakható-e a település és út. Ha szabályos, lehelyezi, és értesíti a játékosokat erről. |
| spendResources(dto: BuyDto) | A dto-ban szereplő buyType alapján utat, települést, várost vagy bónuszkártyát vásárol az aktuális játékos. |
| nextPlayer() | Az aktuális játékos befejezi a körét, a következő játékos lesz az új aktuális játékos. Dob a dobókockával, termel vagy jelzi az aktuális játékosnak, hogy hetest dobott, a bónuszkártyák állapotát frissíti, majd ezekről értesíti a játékosokat. |
| maritimeTrade(  tradeDto: MaritimeTradeDto  ) | A dto alapján leellenőrzi, hogy van-e elegendő nyersanyaga a játékosnak a kiválasztott típusból, és ha van, végrehajtja a cserét. |
| handleTradeOffer(dto: PlayerTradeDto) | A dto alapján leellenőrzi, hogy van-e elegendő nyersanyaga a játékosnak a kiválasztott típusból. Leelenőrzi, hogy a kiválasztott játékos különbözik-e, az aktuális játékostól, és ha igen, akkor elküldi a játékosnak az ajánlatot. |
| acceptOffer(  dto: AcceptTrade,  username: String  ) | Megnézzük, hogy a dto-ban létezik-e a csereazonosító a cserelistában. Ha létezik, ellenőrizzük, a cserélő játékos megegyezik-e a username-mel, illetve, hogy a cserélő játékosnak van-e elegendő nyersanyaga. Ha igen, végrehajtjuk a cserét. |
| useMonopoly(monopolyDto: MonopolyDto) | Megnézi, hogy a jelenlegi játékosnak van-e használható Monopólium kártyája, és ha van, végrehajtja a hatását. |
| useYearOfPlenty(  yearOfPlentyDto: YearOfPlentyDto  ) | Megnézi, hogy a jelenlegi játékosnak van-e használható Találmány kártyája, és ha van, végrehajtja a hatását. |
| useTwoRoad(twoRoadDto: TwoRoadDto) | Megnézi, hogy a jelenlegi játékosnak van-e használható Útépítés kártyája, és ha van, végrehajtja a hatását. Ellenőrzi azt, hogy a második él függ-e az első éltől. |
| steal(stealDto: StealDto) | Ha a dto-ban szereplő isKnight változó igaz, akkor ellenőrzi, hogy van-e használható Lovagkártyája az aktuális játékosnak. Ha van, akkor végrehajtja azt. Ha az isKnight hamis, akkor végrehajtja azt, mintha rablót mozgatnánk. |
| calculatePoints() | Kiszámolja minden játékos pontszámát, és ha az aktuális játékos eléri a nyeréshez szükséges értéket, akkor a játék állapotát nyerésre állítja, és értesíti a játékosokat. |
| afterGameStatistics() | Minden játékos statisztikáját változtatja az alábbiak szerint: a lejátszott meccsek számát eggyel növeli, a győztesek nyert meccseinek a számát is növeli. |

### Leghosszabb kereskedelmi út problémája

A leghosszabb kereskedelmi út problémáját egy módosított Backtrack algoritmussal [12] oldottam meg. A legrövidebb útszakasz helyett a leghosszabb nem kör útvonalat keresem, betartva a Catan definícióját a nem megszakított útszakaszra.

A megoldás kulcsfontosságú része, hogy egy csúcsból egy adott játékosnak megkapjuk a leghosszabb út hosszát. Minden csúcsból megkeressük a leghosszabb út hosszát, majd az így kapott út hosszakból megkeressük a maximumot, és megállapítjuk, hogy történt-e változás a díj birtoklásában.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

3.5.2. ábra: Leghosszabb út meghatározásának struktogramja egy adott csúcsból.

## Tesztelés

A programok várt működését tesztekkel lehet biztosítani, ezzel garantálva, hogy fejlesztés közben az eddigi funkciók megfelelően működnek.

A tesztek formáját Given When Then alakban fogom leírni. A Given a teszt előtti állapotot írja le, a Then azt, hogy mit csinál a felhasználó vagy egy program részlet, és a When, hogy mi az elvárt működés.

### Játékon kívüli tesztek

Az összes tesztnél feltételezzük, hogy a szerver és a kliens sikeresen létrehozták a kapcsolatot. A szervernek van egy elérhető adatbázisa, amiben a szükséges tábla helyesen létre lett hozva.

|  |  |
| --- | --- |
| Tesztnév | Sikeres regisztráció |
| Given | A felhasználó a regisztráló felületen van. |
| When | A felhasználó beír egy olyan felhasználó nevet, ami még nem létezik az adatbázisban, és beír egy érvényes jelszót.  Ezután a felhasználó rányom a Register gombra. |
| Then | A szerver megkapja a kérést, leellenőrzi, hogy nincs-e ilyen felhasználónév regisztrálva, lementi az adatokat, majd jelzi a kliensnek, hogy sikerült a regisztráció. |

|  |  |
| --- | --- |
| Tesztnév | Sikeres bejelentkezés |
| Given | A felhasználó a bejelentkező felületen van. |
| When | A felhasználó beír egy olyan felhasználónév és jelszó kombinációt, ami már be van regisztrálva és rákattint a Login gombra. |
| Then | A szerver megkapja a kérést, leellenőrzi, hogy létezik-e ilyen felhasználó, és ha létezik megnézi-e, hogy a jelszavak megegyeznek-e. Ha megegyeznek, jelzi a kliensnek, hogy sikeres a bejelentkezés, és hozzárendel egy sessionId-t amit elküld a felhasználónak. |

|  |  |
| --- | --- |
| Tesztnév | Elérhető szobák frissítése |
| Given | A felhasználó a LobbySelection felületen van, és a felhasználó sikeresen bejelentkezett. |
| When | A felhasználó frissíti az elérhető szobákat. |
| Then | A szerver megkapja kérést, majd leellenőrzi, hogy érvényes-e a sessionId, majd, ha érvényes, akkor összecsomagolja az el nem indult szobákat, és elküldi a kliensnek.  A kliens a kapott információkat megjeleníti felületen |

|  |  |
| --- | --- |
| Tesztnév | Szoba létrehozása |
| Given | A felhasználó a LobbySelection felületen van, és a felhasználó sikeresen bejelentkezett. |
| When | A felhasználó rányom a Create gombra. |
| Then | A szerver leellenőrzi, hogy jó-e a sessionId, majd létrehozza a várakozó szobát, és elküldi a kliensnek a várakozó szoba adatait, ahol a kliens a szoba adminisztrátora. |

|  |  |
| --- | --- |
| Tesztnév | Szoba csatlakozás |
| Given | A felhasználó a LobbySelection felületen van, és a felhasználó sikeresen bejelentkezett, lekérte az elérhető szobákat, és van egy elérhető szoba, amiben van üres hely. |
| When | A felhasználó kiválasztja az elérhető szobát, majd rányom a Join gombra. |
| Then | A szerver leellenőrzi, hogy jó-e a sessionId, megnézi, hogy a kért szoba létezik-e, megnézi, hogy van-e üres hely a szobában, és utána hozzáadja a felhasználót a szobához, majd a szobában lévő felhasználóknak jelzi, hogy csatlakozott egy új felhasználó. Az összes felhasználó felületén megjelenik az új játékos. |

|  |  |
| --- | --- |
| Tesztnév | Az adminisztrátor elindítja a játékot |
| Given | A felhasználó a szoba adminisztrátora, és elegendő játékos van a szobában. |
| When | A felhasználó rányom a start gombra. |
| Then | A szerver leellenőrzi, hogy a felhasználó-e az adminisztrátor, és elindít egy új játékot. A szobát bezárja, és a felhasználóknak elküldi, a pálya információit, kezdő játékost, hogy melyik felhasználó milyen színű. Kliens oldalon megjelenik az a pálya, amit a szerver generált, és a játékosoknak a színei. |

|  |  |
| --- | --- |
| Tesztnév | Nem adminisztrátor lép ki a szobából a játék elindítása előtt |
| Given | A felhasználó a Lobby felületen van, sikeresen becsatlakozott egy szobába, ahol nem ő az adminisztrátor. |
| When | A felhasználó rányom a Leave gomb-ra. |
| Then | A szerver leellenőrzi, hogy jó-e a sessionId, megnézi, hogy a kért szoba létezik-e megnézi, hogy a felhasználó ténylegesen volt-e a szobában. Ha volt kiveszi a szobából, és jelzi ezt a bent maradt felhasználóknak, hogy valaki kilépett. A jelzett felhasználóknak a felületén megjelenik az új játékos lista. |

|  |  |
| --- | --- |
| Tesztnév | Lekéri egy létező játékos statisztikáját |
| Given | A felhasználó a Statistics felületen van. |
| When | A felhasználó beírt egy létező játékos nevét, és rányomott a Get Stats gombra. |
| Then | A szerver leellenőrzi, hogy létezik-e a játékos név, és visszaküldi a játékos statisztikáit a kliensnek. A felhasználó felületén megjelennek a játékos statisztikája. |

|  |  |
| --- | --- |
| Tesztnév | Lekéri egy nem létező játékos statisztikáját |
| Given | A felhasználó a Statistics felületen van. |
| When | A felhasználó beírt egy nem létező játékos nevét, és rányomott a Get Stats gombra. |
| Then | A szerver leellenőrzi, hogy létezik-e a játékos név, és visszaküldi a kliensnek, hogy nem létezik ilyen játékos. A felhasználó felületén megjelenik, hogy nincs ilyen játékos. |

### Játékon belüli tesztek

Az itteni teszt leírásoknál feltételezzük, hogy sikeresen elindult a játék, három játékos van Peti, Sanyi és Jani. A játékos sorrend Peti, aztán Sanyi, aztán Jani.

|  |  |
| --- | --- |
| Tesztnév | Elindult a játék |
| Given | Peti kijelzőjén jelzi, hogy az ő köre van. |
| When | Peti kiválaszt egy csúcsot és a csúcs melletti élt. A Kliens elküldi a szervernek a kiválasztott csúcs és él id-ját. |
| Then | A szervernél a játék térképen a kiválasztott csúcs és él Petihez tartozik.  A szerver elküldi az összes kliensnek, hogy melyik élt és csúcsot választotta Peti, és hogy Sanyi a következő játékos.  Mindegyik kliensen megjelenik Peti új útja és települése.  Sanyi kijelzőén megjelenik, hogy ő a jelenlegi játékos. |

|  |  |
| --- | --- |
| Tesztnév | Peti és Sanyi már lerakta a kezdő településeket és utakat |
| Given | Peti és Sanyi már lerakott 2-2 települést és utat, Jani egyet. |
| When | Jani kiválaszt egy csúcsot és egy a csúcs melletti élt. A kliens elküldi a csúcs és él id-ját a szervernek. |
| Then | A szervernél a játék térképen a kiválasztott csúcs és él Janihoz tartozik.  A szerver elküldi az összes kliensnek, hogy melyik élt és csúcsot választotta, Peti a következő játékos, és hogy a játék elkezdődött.  Mindegyik kliensen megjelenik Jani új útja és települése.  Peti kijelzőén megjelenik a Buy, Developments, Player Trade, Maritime Trade, Pass gombok.  Többi játékos kijelzőjén a Player Trade inaktív gomb jelenik meg. |

A következő teszteseteknél feltételezzük, hogy már mindegyik játékos letette a kezdeti fázisban a 2-2 utat és települést, a jelenlegi játékos Peti. A játékosoknak feltételezzük, hogy nincs nyersanyagjuk. Minden játékos szabályosan le tud helyezni utat, várost és települést.

|  |  |
| --- | --- |
| Tesztnév | Peti megnyitja a Buy menüt |
| Given | A felületen van egy Buy gomb. |
| When | Peti rányom a Buy gombra. |
| Then | Megjelenik egy új menü, ahol Petinek a vásárlási opció felvannak sorolva. Az összes Buy gomb deaktiválva van. |
| Tesztnév | Peti utat vásárol |
| Given | A Buy menu felület megjelent. A Road alatti Buy gomb aktív. Petinek több, mint egy fája és építőköve van. |
| When | Peti rányom a Road alatti Buy gombra. A pályán pirossal ki vannak emelve azok az élek, amikre utat vásárolhat. Peti rákattint egyre. A kliens elküldi, hogy Peti melyik utat szeretné megvásárolni. |
| Then | A szervernél a pályáján az él Petihez tartozik.  Peti fája és építő köve eggyel csökkent.  A szerver értesítette a Klienseket az új útról, és a nyersanyag változtatásokról.  A klienseknél megjelenik az út, és a nyersanyag csökkenés. |

|  |  |
| --- | --- |
| Tesztnév | Peti települést vásárol |
| Given | A Buy menu felület megjelent. A Settlement alatti Buy gomb aktív. Petinek több, mint 1 fája, építőköve, gyapjúja és gabonája van. |
| When | Peti rányom a Settlement alatti Buy gombra. A pályán megjelennek pirossal azok az csúcsok amelyikre tud építeni települést. A kliens elküldi, hogy Peti melyik csúcsra szeretne települést venni. |
| Then | A játék pályáján lévő csúcson megjelent Petinek egy települése.  Petinek a fája, építő köve, gyapjúja és gabonája csökkent eggyel.  Mindegyik kliens értesült az új településről és nyersanyag csökkenésről. |

|  |  |
| --- | --- |
| Tesztnév | Peti várost vásárol |
| Given | A Buy menu felület megjelent. A City alatti Buy gomb aktív. Petinek több, mint 2 gabonája és 3 érce van. |
| When | Peti rányom a City alatti Buy gombra. A pályán megjelennek szürkével azok az csúcsok amelyikre tud építeni várost. A kliens elküldi, hogy Peti melyik csúcsra szeretne várost venni. |
| Then | A pályán lévő csúcson megjelent Petinek egy városa a település helyén.  Petinek csökkent a gabonája kettővel és érce hárommal.  Mindegyik kliens értesült az új városról és nyersanyag csökkenésről. |

|  |  |
| --- | --- |
| Tesztnév | Peti bónuszkártyát vásárol – Lovag |
| Given | A Buy menu felület megjelent. A Development alatti Buy gomb aktív. Petinek több, mint egy gabonája, gyapjúja és érce van.  A Development poolban a legfelső kártya az egy Lovag. |
| When | Peti rákattint a Development alatti Buy gombra. A kliens elküldi, hogy Peti bónuszkártyát vesz. |
| Then | Petinek adott egy Lovag kártyát, aminek a hasToWait tulajdonsága true.  Minden kliens értesült arról, hogy Peti kapott egy új bónuszkártyát. |

|  |  |
| --- | --- |
| Tesztnév | Peti megnézi a bónuszkártyáit |
| Given | Petinek van egy lovag kártyája, aminek várnai kell még. |
| When | Peti rányom a Developments gombra. |
| Then | A felületen megjelenik a Developments menü, ahol látszik az összes lehetséges bónuszkártya.  A lovagoknál látszik, hogy Petinek van egy olyan lovagja, aminek várnia kell. |

|  |  |
| --- | --- |
| Tesztnév | A bónuszkártyák, amiknek várnia kellet, a következő körben már nem kell |
| Given | Petinek van minden típusú bónuszkártyája, és mindegyiknek várnia kell. |
| When | Peti, aztán Sanyi, aztán Jani is passzolja a körét. |
| Then | Peti rányom a Developments gombra és látja, hogy mindegyik bónuszkártyáját kijátszhatja. |

|  |  |
| --- | --- |
| Tesztnév | Peti kijátssza a lovagi kártyát |
| Given | Petinek van egy lovag kártyája, amit azonnal kijátszhat. A rabló a sivatagban tartózkodik. |
| When | Peti kijátssza a lovagkártyát. Rákattint egy olyan mezőre, amin Sanyinak van egy települése. |
| Then | A pályáján a sivatagon nincs rabló, a mező amire kattintott Peti arra rákerült a Rabló.  Sanyinak csökkent a nyersanyagjai eggyel.  Petinek növekedett a nyersanyagjai eggyel.  Petinek az aktív lovagjainak száma növekedett eggyel.  Peti kártyái között eggyel kevesebb kijátszható lovagi kártyája van.  Mindegyik kliens értesült a rabló mozgásról, nyersanyag változásról, és hogy Petinek van egy új aktív lovagja. |

|  |  |
| --- | --- |
| Tesztnév | Peti kijátssza a Találmány kártyát |
| Given | Petinek van egy találmány fejlesztése, amit kijátszhat. |
| When | Peti kijátssza a találmány fejlesztést. A felületen megjelenik a találmány felület, ezen a felületen Peti kiválaszt 2 tetszőleges nyersanyagot. |
| Then | Peti megkapta a választott nyersanyagjait.  Peti kártyái között eggyel kevesebb olyan találmány van, amit ki tud játszani.  Mindegyik kliens értesült arról, hogy Petinek több nyersanyaga lett, és hogy eggyel kevesebb bónuszkártyával rendelkezik. |

|  |  |
| --- | --- |
| Tesztnév | Peti kijátssza az Útépítés kártyát |
| Given | Petinek van egy Útépítés fejlesztése, amit kijátszhat. |
| When | Peti kijátssza az Útépítés fejlesztést. A felületen megjelennek a vásárolható utak. Kiválaszt közülük 2 utat. |
| Then | Szervernél a kiválasztott 2 út Petié lett.  Petinek eggyel csökkent a kijátszható Útépítés bónuszkártyái.  Mindegyik kliens értesült arról, hogy került a pályára 2 új út, ami Petié, és hogy Petinek eggyel kevesebb bónuszkártyája van. |

|  |  |
| --- | --- |
| Tesztnév | Peti kijátssza a Monopólium kártyát |
| Given | Petinek van egy Monopólium fejlesztése, amit kijátszhat. |
| When | Peti kijátssza a Monopólium kártyát. A felületen megjelenik a Monopólium menü. Peti kiválaszt egy tetszőleges nyersanyagot. |
| Then | Sanyinak a választott nyersanyagból 0-ja van.  Janinak a választott nyersanyagból 0-ja van.  Petinek a választott nyersanyagból annyival van több, amennyivel Janinak és Sanyinak csökkent.  Petinek eggyel csökkent a kijátszható Monopólium bónuszkártyái.  Mindegyik kliens értesült arról, hogy elvesztették a nyersanyagjait, és hogy Petinek növekedett, és hogy eggyel kevesebb bónuszkártyája van. |

|  |  |
| --- | --- |
| Tesztnév | Peti tengeri kereskedik |
| Given | Petinek van 4 gabonája. Peti az aktív játékos. |
| When | Peti rákattint a Maritime Trade gombra, és kiválasztja, hogy 4 gabonát cserél 1 fára. |
| Then | Petinek a gabona mennyisége 0.  Petinek a fa mennyisége 1.  Mindegyik kliens értesült arról, hogy Petinek nyersanyag változása volt. |

|  |  |
| --- | --- |
| Tesztnév | Peti cserét kezdeményez Sanyival |
| Given | Petinek van 1 gabonája és Sanyinak van 1 fája. Peti az aktív játékos. |
| When | Peti rákattint a Player Trade gombra, és kiválasztja, hogy Sanyival cserélne egy gabonát egy fára. |
| Then | A szervernél a currentTradeOffers-be bekerült az ajánlat.  Sanyi értesül arról, hogy Peti akar vele cserélni, egy gabonát egy fáért. |

|  |  |
| --- | --- |
| Tesztnév | Sanyi elfogadja Peti ajánlatát |
| Given | Petinek van egy gabonája és Sanyinak van egy fája. Peti az aktív játékos. Peti küldött egy cserét egy gabonáért kér egy fát Sanyitól. |
| When | Sanyi megnyitja a Player Trade ablakot, és elfogadja a cserét. |
| Then | A szervernél a currentTradeOffers üres.  Sanyinak csökken eggyel a fája.  Sanyinak növekedik eggyel a gabonája.  Petinek csökken eggyel a gabonája.  Petinek növekedik eggyel a fája.  Mindegyik kliens értesült arról, hogy nyersanyag változások voltak. |

|  |  |
| --- | --- |
| Tesztnév | Peti passzol – nem hetest dob |
| Given | Peti köre van. Van egy aktív csere ajánlata Sanyival. A következő dobás összege 8. Petinek az egyik települése olyan mezőn van, ami fát ad, és 8-as szám tartozik hozzá. A többiek települése és városa nincs olyan mezőn, amihez 8-as tartozik. |
| When | Peti rányom a Pass gombra. |
| Then | A szervernél a jelenlegi játékos Sanyi.  A szervernél a currentTradeOffers üres.  A szerver dobott dobókockával értesíti a dobás eredményével a klienseket.  Petinek a fa mennyiség növekedett eggyel.  Többi játékos nyersanyagjai változatlan. |

|  |  |
| --- | --- |
| Tesztnév | Peti passzol – hetest dob |
| Given | Peti köre van. Van egy aktív csere ajánlata Sanyival. A következő dobás összege 7. |
| When | Peti rányom a Pass gombra. |
| Then | A szervernél a jelenlegi játékos Sanyi.  A szervernél a currentTradeOffers üres.  A szerver dobott dobókockával értesíti a dobás eredményével a klienseket.  A játékosok nyersanyagai változatlan.  A szerver jelzett Sanyi kliensének, hogy mozgassa a Rablót. |

|  |  |
| --- | --- |
| Tesztnév | Sanyi játékos rablót mozgat, csak 1 ember van a mezőn |
| Given | Sanyi köre van. |
| When | Sanyi kliense kapott a szervertől olyan információt, hogy mozgassa a rablót. Áthelyezi egy másik mezőre, ahol csak Janinak van települése, Janinak van egy fája, Sanyinak 0 fája van. |
| Then | Janinak a fa mennyisége 0.  Sanyinak a fa mennyisége 1.  Mindegyik kliens értesítve lett arról, hogy a Janinak és Sanyinak változott a nyersanyaga, és a rabló át lett mozgatva. |

|  |  |
| --- | --- |
| Tesztnév | Sanyi rablót mozgat, több ember van a mezőn |
| Given | Sanyi köre van. |
| When | Sanyi kliense kapott a szervertől olyan információt, hogy mozgassa a rablót. Áthelyezi egy másik mezőre, ahol Janinak és Petinek van települése, Janinak van egy fája, Sanyinak 0 fája van. Sanyi Janit választja ki, hogy tőle lopjon. |
| Then | Janinak a fa mennyisége 0.  Sanyinak a fa mennyisége 1.  Mindegyik kliens értesítve lett arról, hogy a Janinak és Sanyinak változott a nyersanyaga, és a rabló át lett mozgatva. |

|  |  |
| --- | --- |
| Tesztnév | Sanyi rablót mozgat, csak ő van a mezőn |
| Given | Sanyi köre van. |
| When | Sanyi kliense kapott a szervertől olyan információt, hogy mozgassa a rablót. Áthelyezi egy másik mezőre, ahol csak a saját települése található meg. |
| Then | A játékosok nyersanyag mennyisége változatlan.  Mindegyik kliens értesítve lett arról, hogy a rabló át lett mozgatva. |

|  |  |
| --- | --- |
| Tesztnév | Sanyi rablót mozgat, nincs a mezőn település vagy város. |
| Given | Sanyi köre van. |
| When | Sanyi kliense kapott a szervertől olyan információt, hogy mozgassa a rablót. Áthelyezi egy másik mezőre, ahol senkinek nincs települése vagy városa. |
| Then | A játékosok nyersanyag mennyisége változatlan.  Mindegyik kliens értesítve lett arról, hogy a rabló át lett mozgatva. |
| Tesztnév | Peti megszerzi a Lovagi hatalmat |
| Given | Peti köre van, kijátszott az előző körökben 2 lovagi kártyát. Petinek van egy használható lovagi kártyája. |
| When | Peti kijátssza a lovagi kártyáját, rablót elmozgatja egy üres mezőre. |
| Then | A játékosok nyersanyag mennyisége változatlan.  Peti megkapta a Lovagi hatalom díjat.  Mindegyik játékos értesítve lett arról, hogy Peti megkapta a díjat, és hogy a rabló arrébb mozgott. |

|  |  |
| --- | --- |
| Tesztnév | Peti megszerzi Leghosszabb kereskedelmi utat |
| Given | Peti köre van. Előző körökben épített egy 4 hosszú utat. Ezt az utat lehet folytatni úgy egy ötödik úttal, hogy Peti megkapja a leghosszabb kereskedelmi út díjat. Petinek van egy fája és építőköve. |
| When | Peti megépíti úgy az utat, hogy megkapja a Leghosszabb kereskedelmi utat. |
| Then | Petinek 0 fája és építő köve van.  Peti megkapta a Leghosszabb kereskedelmi út tulajdonát.  Mindegyik játékos értesítve lett arról, hogy Peti megkapta a leghosszabb kereskedelmi utat, rakott le egy utat, és csökkent a nyersanyaga. |

|  |  |
| --- | --- |
| Tesztnév | Peti megszerzi az utolsó pontot a nyereséghez |
| Given | A jelenlegi játékban az elérendő pontok száma 10.  Petinek van 5 települése és 4 városa, 3 érce és 2 gabonája. Peti köre van. |
| When | Peti az egyik települését felfejleszti várossá. |
| Then | Petinek 0 érce és gabonája van.  Peti elérte a 10 pontot.  Mindegyik játékos értesítve lett arról, hogy Peti megnyerte a játékot. |

## További fejlesztési lehetőségek

Jelenleg nincsen implementálva a kikötői csere lehetősége, illetve az, hogy hetes dobás esetén azok a játékosok, akik hétnél több nyersanyagkártyával rendelkeznek, kötelesek eldobni a nyersanyagkártyáik felét.

A játék felhasználói felülete jelenleg kezdetleges. A játék elemeit lehetne grafikákkal helyettesíteni, ezzel is színesebbé tenni a játékélményt.

A játék felületét felhasználóbarátabbá tenné, ha lenne egy visszajelző log, amelyben nyomon lehetne követni a játék történéseit.

A programot sokféleképpen lehet tovább fejleszteni egyéb funkciókkal. Lehetne készíteni egy gyors játék indítás módot, ahol a játékosok vagy képesség szint alapján, vagy teljesen véletlenszerűen kerülnének egy játékba.

A szobák adminisztrátorai tudják az adott játékot testre szabni, a bónuszkártyapakliban lévő kártyák mennyiségét változtatni, utak, települések és városok mennyiségét korlátozni, beállítani, hogy a leghosszabb kereskedelmi útnak mi legyen a minimális hossza. Lehetősége legyen a szoba adminisztrátorának egyedi pályát készítésére, így ne csak a szerver által beállított pályán játszhassanak a játékosok. A játékosok tudják kiválasztani a saját színüket.

A játékosok közti kommunikációt is meglehetne könnyíteni egy chat rendszerrel, ahol a játékosok tudnának egymással beszélgetni a játék közben.

Több statisztikai adatot lehetne tárolni a játékosokról. A játszott meccsek előzményeit le lehetne menteni és visszanézni, hogy az elmúlt időben ki mennyit nyert, kik ellen játszott mennyi körig vagy ideig tartott a játék. Játékosok általános statisztikáit ki lehetne egészíteni azzal, hogy mennyi pontot érnek el átlagosan egy meccsben, átlagos leghosszabb kereskedelmi útjuk a játék végén, mennyi bónuszkártyát vettek a játékaik során, melyik kártyából mennyit kaptak, és azokból hányat játszottak ki.

A Catan játéknak számos kiegészítője jelent meg, ezeket is le lehetne bővíteni a programot.

# Befejezés

A dolgozatom témáját a társas játékok iránti érdeklődés miatt választottam. A Kotlin programozás nyelvvel a szakmai gyakorlat közben ismerkedtem meg igazán, ahol segíthettem a munkahelyemnek, egy Java projektet átírni Kotlinba. Akkor szerettem meg ezt a nyelvet, és tudtam, hogy ebben szeretném írni a szakdolgozatomat.

A legnagyobb kihívást az okozta, hogyan kommunikáljon a kliens és a szerver egymással, hogy lehetne kód szinten jól kezelni, hogy különböző játékosok különböző információkat kapjanak, az üzenetek hogyan legyen reprezentálva. Korábban nem csináltam még komoly Socket alapú kommunikációt, ahol a szerver és a kliens dinamikusan kapcsolatot tart egymással.

A második legnagyobb fejtörést pedig a játék állapot tér reprezentálása okozta. Hogyan lehet úgy készíteni egy hatszögekből álló táblát, hogy a mezők rálássanak a csúcsaira és az egymás mellett lévő mezőknek a csúcsai megegyezzenek.

A program készítése közben rengeteg tapasztalatot szereztem a Socket alapú kommunikációról, a folyamatok tervezéséről, a problémamegoldásról. A fejlesztés közben többször újra terveztem a szerver és a kliens közötti kommunikációt. Meg kellett tanulnom, hogy mikor kell újrakezdeni, és másik megoldási irányt választani.

# Irodalomjegyzék

1. Catan társasjáték ismertetője [https://www.catan.com/catan](https://www.catan.com/catan%20)   
   (Utolsó megtekintés: 2023.04.15.)
2. Spiel des Jahres díj <https://www.spiel-des-jahres.de/en/spiele/die-siedler-von-catan/> (Utolsó megtekintés: 2023.04.15.)
3. Catan játékszabály 5. kiadás <https://www.catan.com/sites/default/files/2021-06/catan_base_rules_2020_200707.pdf> (Utolsó megtekintés: 2023.04.15.)
4. IntelliJ Idea <https://www.jetbrains.com/idea/> (Utolsó megtekintés: 2023.05.12.)
5. Kotlin dokumentáció <https://kotlinlang.org/docs/home.html> (Utolsó megtekintés: 2023.05.12.)
6. Ktor dokumentáció <https://ktor.io/docs/welcome.html> (Utolsó megtekintés: 2023.05.12.)
7. Kotest dokumentáció <https://kotest.io/docs/framework/framework.html> (Utolsó megtekintés: 2023.05.14.)
8. Gson github <https://github.com/google/gson> (Utolsó megtekintés: 2023.05.14.)
9. Hibernate <https://hibernate.org/orm/> (Utolsó megtekintés: 2023.05.14.)
10. TornadoFX dokumentáció <https://tornadofx.io> (Utolsó megtekintés: 2023.05.16.)
11. Gradle <https://gradle.org> (Utolsó megtekintés: 2023.05.16.)
12. Backtrack <https://en.wikipedia.org/wiki/Backtracking> (Utolsó megtekintés: 2023.05.16.)

1. Egy kör alatt legfeljebb egy fejlesztéskártya használható fel. Az aktuális körben vásárolt fejlesztéskártya nem játszható ki (kivétel a pontkártya, ha ezzel nyer a játékot.) [↑](#footnote-ref-1)
2. Integrated Development Environment [↑](#footnote-ref-2)
3. Java Virtual Machine [↑](#footnote-ref-3)
4. Object Relation Mapping [↑](#footnote-ref-4)
5. Software Development Kit [↑](#footnote-ref-5)
6. Data Transfer Object [↑](#footnote-ref-6)
7. Transmission Control Protocol [↑](#footnote-ref-7)
8. User Datagram Protocol [↑](#footnote-ref-8)