Paralelna implementacija kartaške igre Blackjack

Sebastian Medjaković
Voditelj projekta
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb

Nika Šljubura
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb

Nikola Marić
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb

Domagoj Capar
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb

Tomislav Kožul
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb

***Abstract*—U sklopu ovog projekta razvijena je aplikacija za igru kartaške igre „Blackjack“. Aplikacija je razvijena koristeći programski jezik Erlang i razvojni alat Rebar3. Erlang nudi podršku za paralelno izvođenje programskog koda, što omogućuje igračima (modul „player“) da istovremeno komuniciraju s djeliteljem karata (modul „dealer“) te da istovremeno zatraže karte, bez mogućnosti da dobiju istu kartu.**

***Index Terms*—Blackjack, Erlang, Paralelizam, Konkurentnost**

I. Uvod

Blackjack [1] je kartaška igra u kojoj igrači ne igraju međusobno, već svaki igrač igra protiv kuće (kockarnice). Kada se igra uživo s fizičkim kartama, svaki igrač ima svoj potez kada smije igrati, a ostatak vremena čeka da drugi igrači odigraju vlastite poteze. Slična implementacija koristi se u većini online kockarnica.

Ako su u igri četiri igrača, svaki od njih će tri četvrtine vremena provedenog igrajući čekati svoj red za igru. U sklopu ovog projekta htjeli smo razviti aplikaciju koja će znatno smanjiti vrijeme čekanja na potez igrača. Samim time, poboljšali bismo vremensku efikasnost igre, a posljedično i igračevo iskustvo.

Kako bismo ostvarili navedeno, razvili smo aplikaciju koja igračima dopušta simultano igranje, bez čekanja na red. Kada djelitelj započne igru, svi igrači dobivaju karte te imaju pravo odigrati poteze u isto vrijeme, čime smo efektivno vrijeme igranja povećali s jedne četvrtine na približno jedan.

II. Pozadina

A. Pravila igre blackjack

Prije donošenja odluke o tehnologiji za implementaciju igre Blackjack, nužno je detaljno razumjeti pravila i tijek igre, moguće poteze igrača, logiku djelitelja i ishode ovisno o vrijednostima karata u ruci, no za početak, osvrnimo se na osnovna pravila igre.

Špil za kartanje igre Blackjack sastoji se od nekoliko standardnih špilova od pedeset i dvije karte, bez karte *'joker'*. Karte od 2 do 10 nose vrijednost jednaku vlastitom broju, tako na primjer karta *'3 srce'*, nosi vrijednost 3. Karte *'dečko'*, *'dama'* i *'kralj'* nose vrijednost deset, dok situacija s kartom *'as'* nije toliko jednostavna. Naime, početna vrijednost karte *'as'* iznosi jedanaest – tada kažemo da je ruka *'soft'*. Ukoliko je zbroj vrijednosti karata u ruci veći od dvadeset jedan i ruka sadrži kartu *'as'*, utoliko iznos njezine vrijednosti više nije jedanaest, već jedan, čime ruka postaje *'hard'*.

Na početku igre, nakon što su igrači postavili svoje uloge, djelitelj dijeli po dvije karte svakom igraču i otkriva vrijednost jedne od svojih karata. Po primitku karata, svaki od igrača donosi jednu od dvije moguće odluke: zatražiti novu kartu od djelitelja (tzv. *'hit'*) ili nastaviti igrati s trenutnim kartama (tzv. *'stand'*). Ne postoji ograničenje koliko karata igrač može zatražiti od djelitelja, sve dok je zbroj vrijednosti karata u igračevoj ruci manji ili jednak broju dvadeset jedan.

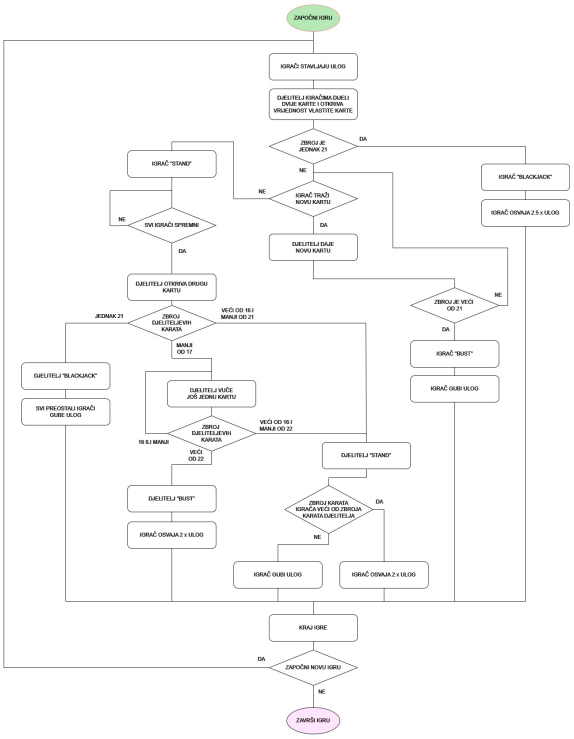
Ako je u igračevoj ruci po dobitku prve dvije karte jednak dvadeset jedan, igrač je ostvario *'blackjack'* (termin po kojem je igra dobila ime), čime igrač automatski osvaja iznos jednak dva i pol puta veći od uloga te je igra za njega gotova. Blackjack se postiže samo s dvije karte: *'as-om'* i kartom vrijednosti deset. S druge strane, ako zbroj karata u igračevoj ruci premaši dvadeset jedan (tzv. *'bust'*), igrač gubi svoj ulog i završava igru.

Nakon što su svi igrači, osim onih koji su ostvarili *'blackjack'* ili *'bust'*, spremni za nastavak igre s trenutnim kartama (*'stand'*), djelitelj igračima otkriva vrijednost svoje druge karte. Djelitelj izvlači nove karte sve dok je zbroj vrijednosti karata u njegovoj ruci manji od sedamnaest. Ako je zbroj karata u djeliteljevoj ruci veći od dvadeset jedan (*'bust'*), svaki od igrača koji su još u igri osvaja iznos dvostruko veći od njihovog uloga. Inače, samo igrači koji su nastavili igrati te čiji je zbroj vrijednosti karata u ruci veći od zbroja vrijednosti karata u djeliteljevoj ruci osvajaju iznos dvostruko veći od uloga, dok svi preostali igrači gube svoj ulog.

Time je završena igra. Nova igra započinje postavljanjem uloga od strane igrača. Pošto djelitelj predstavlja kuću (kockarnicu), on ne stavlja ulog. Djelitelj je na dobitku ako igrači gube.

B. Tijek igre blackjack

U pravilima igre Blackjack pojavljuju se uloge djelitelja i igrača. Obje uloge bit će implementirane u zasebnim modulima. Raspodjelom funkcionalnosti u odvojene module omogućit ćemo bolju čitljivost koda, skalabilnost te olakšati ponovnu upotrebu komponenata igre. Dijagram tijeka igre Blackjack prikazan je na *slici 1.*



Slika 1. Dijagram tijeka igre Blackjack u kojoj je na poslužitelja (djelitelja) spojen najmanje jedan klijent (igrač)

III. Paralelizacija igre Blackjack

A. Funkcionalnost djelitelja

Djeliteljeva glavna zadaća je raspodjela karata igračima. Iako se djeliteljev zadatak čini banalnim, dijeljenje karata i komunikacija s igračima omogućuju upravljanje tijekom igre. Budući da djelitelj ne sudjeluje u igri direktno, već se ponaša u skladu s igračevim zahtjevima i predefiniranim pravilima, u kontekstu aplikacije ulogu djelitelja možemo dodijeliti poslužitelju (*engl.* **server**).

Prije početka same igre, djeliteljev zadatak je promiješati karte, odnosno, ako pričamo o programskoj implementaciji djelitelja, *generirati špil* za igru. Djelitelj započinje igru kada svakom od igrača podijeli dvije karte te ih obavijesti o karti koju je sam izvukao.

Na zahtjev igrača za vrijeme njegova poteza, djelitelj istome dodjeljuje novu kartu iz špila, sve dok je zbroj vrijednosti karata u ruci igrača manji od dvadeset jedan. Kada su svi igrači spremni za nastavak igre (*'stand'*, *'bust'* ili *'blackjack'*), djelitelj igračima otkriva svoju drugu kartu te vuče nove karte sve dok je zbroj vrijednosti karata u njegovoj ruci manji od sedamnaest.

Kada je spomenuta suma vrijednosti karata premašena, djelitelj igrače obavještava o zbroju vrijednosti karata u ruci te nakon podjele osvojenih iznosa započinje novu igru. Također ako broj karata u igraćem špilu padne ispod određene vrijednosti, djelitelj promiješa preostale i već iskorištene karte, čime se igraći špil vraća u početno stanje.

1) Funkcionalnost igrača: Ako djelitelju dodjeljujemo ulogu poslužitelja, u istom kontekstu igrač ima ulogu klijenta (*engl.* **client**). Klijent (igrač) se spaja na poslužitelj (djelitelja) čime započinje njihova komunikacija. Broj igrača koji sudjeluju u jednoj igri nije ograničen.

Po ulasku u igru, igrač kroz korisničko sučelje unosi iznos ukupnih raspoloživih sredstava te na početku svake igre unosi svoj ulog za istu. Nakon što su mu dodijeljene karte, igrač bira želi li novu kartu ili želi nastaviti igrati s kartama koje trenutno ima u ruci. Igraču su na raspolaganju iste opcije po dodjeli svake nove karte, sve dok zbroj vrijednosti karata u njegovoj ruci ne premaši dvadeset jedan, kada o istome obavještava djelitelja i završava igru. S druge strane, ako je pri prvoj dodjeli karata ostvario *'blackjack'*, odnosno dobio karte ukupne vrijednosti dvadeset jedan, odmah osvaja adekvatan iznos i završava igru. Također, igrač za vrijeme igre može promijeniti ulog (tzv. "stake").

Ako je igrač i dalje u igri, ishod se određuje po primitku ukupne vrijednosti djeliteljeve ruke. Ako je iznos ukupne vrijednosti karata u igračevoj ruci veći od onog u djeliteljevoj, igrač pobjeđuje i vrši se isplata. U suprotnom, igrač je izgubio.

Po završetku svake igre, igrač ponovo unosi svoj ulog i započinje nova igra.

2) Paralelizacija igre Blackjack: U tradicionalnoj igri s fizičkim kartama, protekne određeno vrijeme prije samog početka igre dok djelitelj svakom igraču podijeli dvije karte. Budući da je u aplikaciji nasumične karte moguće izvući (generirati) i o istima obavijestiti igrača gotovo trenutno, taj vremenski period možemo zanemariti. Trenutnom dodjelom karata smo ubrzali tijek igre u odnosu na igru s fizičkim kartama. Radi vjerodostojnije usporedbe paralelnog i serijskog izvođenja igre, oba slučaja su promotrena u kontekstu aplikacije (s trenutnim generiranjem i podjelom karata).

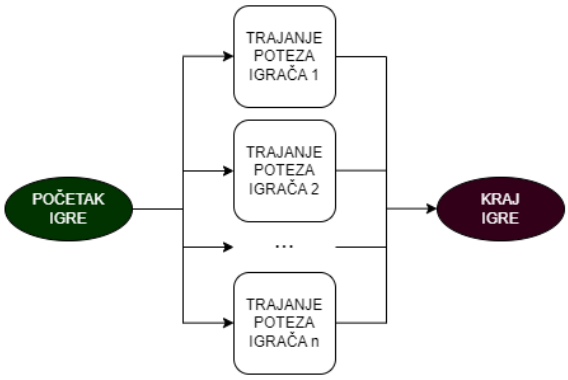
Ako svaki igrač (osim prvog) mora čekati završetak poteza prethodnog igrača kako bi započeo vlastiti, radi se o **serijskom** izvođenju igre, čiji je dijagram tijeka prikazan na *slici 2.*

Slika 2. Dijagram tijeka *serijskog* izvođenja igre gdje je vrijeme trajanja poteza svakog igrača označeno s 'TRAJANJE POTEZA IGRAČA X', gdje 'X' označava redni broj igrača, a 'n' ukupni broj igrača

Kod serijskog izvođenja, ukupno vrijeme trajanja igre jednako je zbroju trajanja poteza svakog pojedinog igrača. Ukupno trajanje igre računa se prema formuli *(1)*, gdje T_{ukupno} označava ukupno trajanje igre, T_x vrijeme trajanja poteza igrača s rednim brojem x, a n ukupan broj igrača.

$$T_{ukupno} = \sum_{x=1}^{n} T_x \tag{1}$$

S druge strane, ako svi igrači započinju vlastiti potez u isto vrijeme, radi se o **paralelnom** izvođenju igre, čiji je dijagram tijeka prikazan na *slici 3*.



Slika 3. Dijagram tijeka *paralelnog* izvođenja igre gdje je vrijeme trajanja poteza svakog igrača označeno s 'TRAJANJE POTEZA IGRAČA X', gdje 'X' označava redni broj igrača, a 'n' ukupni broj igrača

Kod paralelnog izvođenja, ukupno vrijeme trajanja igre jednako je vremenu trajanja najduljeg poteza među svim igračima. Ukupno trajanje igre računa se prema formuli *(2)*, gdje T_{ukupno} označava ukupno trajanje igre, T_x vrijeme trajanja poteza igrača s rednim brojem x, a n ukupan broj igrača.

$$T_{ukupno} = \max\left(T_1, T_2, ..., T_n\right) \tag{2}$$

Kako bismo odredili točan omjer efektivnog vremena provedenog igrajući Blackjack, pretpostavit ćemo da su vremena trajanja poteza svih igrača međusobno jednaka i to vrijeme označit ćemo s T_{potez}, a ukupan broj igrača označit ćemo s n. Pod pretpostavkom da je vrijeme ukupnog trajanja igre **kraće u slučaju paralelnog izvođenja u odnosu na serijsko**, rezultat omjera ukupnog trajanja serijskog (označenog kao $T_{serijsko}$) i paralelnog (označenog kao $T_{paralelno}$) izvođenja označit će višestrukost ubrzanja paralelizacijom igre. Spomenuti omjer prikazan je u funkciji *(3)*.

$$\begin{aligned} \frac{T_{serijsko}}{T_{paralelno}} &= \frac{\sum_{x=1}^{n} T_{potez}}{\max\left(T_{potez}, T_{potez}, ..., T_{potez}\right)} \\ &= \frac{n * T_{potez}}{T_{potez}} \\ &= n \end{aligned} \tag{3}$$

Paralelnim izvođenjem igre Blackjack trajanje igre skratili smo **približno *n* puta**, gdje je n ukupan broj igrača.

IV. Implementacija

1) Problem paralelnog izvođenja: U programskoj implementaciji igre Blackjack, želimo igraći špil simulirati što vjerodostojnije fizičkom špilu u tradicionalnoj igri, stoga nakon izvlačenja karte, istu želimo ukloniti iz špila. Budući da su svi igrači na potezu u isto vrijeme, postoji mogućnost da dva ili više igrača kartu zatraže istovremeno. Djelitelj istu fizičku kartu ne može dodijeliti dva igrača, pa taj scenarij moramo spriječiti. Budući da smo igru odlučili implementirati paralelno, igrače možemo gledati kao **zasebne dretve**. Špil karata je zajednički resurs svim igračima, pa je potrebno onemogućiti višestruki istovremeni pristup istome, što je zahtjev koji će igrati ključnu ulogu pri odabiru programske potpore za razvoj igre.

A. Korištena tehnologija

Glavni uvjet pri odabiru programskog jezika za implementaciju igre Blackjack je da igraći špil bude dostupan u svakom trenutku, uz onemogućavanje višestrukog istovremenog pristupa resursu. Programski jezik koji zadovoljava ovaj uvjet visoke dostupnosti je **Erlang** [2]. Dodatno, korišten je alat Rebar3 [3], koji je pružio funkcionalnosti potrebne za lakše pokretanje i izgradnju modula.

Iako je uvjet visoke dostupnosti vrlo važan, nije jedini potreban za programsku implementaciju igre. Kako bi igra uopće mogla biti pokrenuta, potrebno je omogućiti komunikaciju djelitelja (poslužitelja) i igrača (klijenata). Za to smo koristili Erlangov *gen_server* modul ponašanja, koji pruža funkcionalnosti generičkog **klijent-poslužitelj** odnosa [4].

B. Komunikacija između modula

Za komunikaciju djelitelja i igrača korištene su procedure za obradu zahtjeva *'call'* (omogućuje **sinkronu** komunikaciju) i *'cast'* (omogućuje **asinkronu** komunikaciju). Obje procedure služe za poziv funkcija ili obradu zahtjeva unutar istog modula (na primjer, kada poslužitelj šalje zahtjev koji se obrađuje unutar istog poslužitelja, ili analogno za klijenta) te između različitih, međusobno povezanih, modula (na primjer, kada funkcija s klijenta poziva funkciju na poslužitelju, ili obrnuto). Prilikom korištenja *sinkrone* komunikacije, program unutar modula nastavlja s izvršavanjem nakon završetka obrade zahtjeva ili izvođenja funkcije. Nasuprot tome, kod *asinkrone* komunikacije program odmah nastavlja s izvršavanjem, ne čekajući završetak obrade.

C. Zaštita od višestrukog pristupa

Stanje modula (*engl.* **state**) omogućuje pohranu podataka relevantnih za upravljanje tokom igre. Tako poslužitelj čuva špil igraćih karata, ažurirajući ga po izvlačenjem svake karte, kao i zbroj vrijednosti karata u vlastitoj ruci i broj igrača koji više ne sudjeluju u igri. Broj neaktivnih igrača koristi za računanje kako bi znao jesu li svi preostali igrači spremni. Igrač također mora ažurirati zbroj vrijednosti karata u ruci te pratiti balans i uloge. Kada igrač želi zatražiti novu kartu, ne može špilu pristupiti izravno, već to čini slanjem zahtjeva poslužitelju.

U Erlangu nije moguć paralelan pristup podacima spremljenima u *stanje*, već se pristup i obrada istih obrađuje sekvencijalno. Time je onemogućen istovremeni pristup špilu od strane više igrača, što rješava ključni problem paralelne implementacije igre Blackjack.

Ovaj pristup nije ograničen isključivo na implementaciju igre Blackjack; sličan pristup može se primijeniti u širokom spektru domena s problemima zajedničkog pristupa resursima.

Implementacijom ponašanja navedenih u poglavlju *Funkcionalnosti djelitelja* u modul servera te onih navedenih u *Funkcionalnosti igrača* u modul klijenta i njihovim povezivanjem, uspjeli smo ispuniti sve zahtjeve potrebne za provođenje igre Blackjack.

V. Rezultati

A. Simulacija igre Blackjack

Na samome kraju, prije zaključka, proveli smo nekoliko simulacija igre Blackjack s različitim ishodima. Budući da je djelitelj automatiziran, na konzoli djelitelja nije bilo ispisa. Sve djeliteljeve poruke su poslane igračima te prikazane na njihovim korisničkim sučeljima, stoga su na slikama prikazane korisničke konzole.

B. Povezivanje igrača s djeliteljem

Kada igrač pokrene igru, pokušava se kao klijent spojiti na poslužitelja, odnosno djelitelja. Nakon uspješnog spajanja, igrač treba unijeti svoj balans te ulog za prvu igru. Opisani proces pokretanja igre prikazan je na *slici 4*.

```
player console
Eshell V13.1.5  (abort with ^G)
(player@DESKTOP-ED8HE0U)1> Povezivanje s dealerom ...
(player@DESKTOP-ED8HE0U)1> Povezano!
(player@DESKTOP-ED8HE0U)1> Upisite svoj balans: 10000
(player@DESKTOP-ED8HE0U)1> Upisite svoj ulog: 100
(player@DESKTOP-ED8HE0U)1> Dobrodosli!
Balans: 10000
Ulog:   100
```

Slika 4. Povezivanje igrača s poslužiteljem (djeliteljem) i početak igre

Budući da se igra provodi naredbama u konzoli, implementirana je naredba *'help'*, koja korisniku ispisuje sve implementirane naredbe i njihova značenja. Ispis na konzoli nakon poziva naredbe *'help'* prikazan je na *slici 5*. Također, nakon naredbe *'help'*, pozvana je naredba *'stake'*, čime je igraču omogućena promjena uloga.

```
(player@DESKTOP-ED8HE0U)1> Za pomoc u igri napisite help!

(player@DESKTOP-ED8HE0U)1> INPUT:   help
(player@DESKTOP-ED8HE0U)1>    HELP:

   ---------------------------------
   start --- pocetak igre

   hit   --- vuci kartu

   stand --- dosta

   stake --- promjena uloga

   quit  --- izlaz

(player@DESKTOP-ED8HE0U)1> INPUT:   stake
(player@DESKTOP-ED8HE0U)1>
Promijenite svoj ulog: (player@DESKTOP-ED8HE0U)1> 200
```

Slika 5. Poziv naredbe *'help'* te promjena uloga

C. Igrač gubi

Prvi prikazani ishod igre je gubitak igrača, preciznije gubitak premašivanjem maksimalne dozvoljene vrijednosti zbroja sume karata u igračevoj ruci. Na *slici 6*, prikazan je slučaj kada je zbroj vrijednosti karata u igračevoj ruci bio jednak 24. Budući da je maksimalna dozvoljena vrijednost ruke jednaka 21, igrač je ostvario *'bust'*, što je i prikazano na konzoli. Također, vidimo da se balans igrača smanjio za iznos jednak ulogu.

```
player console
(player@DESKTOP-ED8HE0U)1> INPUT:   start
(player@DESKTOP-ED8HE0U)1>
KARTE:           [4,"Q"]
(player@DESKTOP-ED8HE0U)1>
SUMA:            {14,hard}
(player@DESKTOP-ED8HE0U)1>
DEALEROVA KARTA: 3

(player@DESKTOP-ED8HE0U)1> INPUT:   hit
(player@DESKTOP-ED8HE0U)1> NOVA KARTA:   "Q"
(player@DESKTOP-ED8HE0U)1> KARTE:        ["Q",4,"Q"]
(player@DESKTOP-ED8HE0U)1> SUMA:         {24,hard}

(player@DESKTOP-ED8HE0U)1>
BUST!

(player@DESKTOP-ED8HE0U)1>
STAND!

KARTE:  ["Q",4,"Q"]
(player@DESKTOP-ED8HE0U)1> SUMA:   24
(player@DESKTOP-ED8HE0U)1>
CEKAMO DEALEROVE KARTE ...

(player@DESKTOP-ED8HE0U)1>
DEALEROVA SUMA:  17
(player@DESKTOP-ED8HE0U)1>
VASA SUMA:       24
(player@DESKTOP-ED8HE0U)1>
BUST.
(player@DESKTOP-ED8HE0U)1>
BALANS:   9800
ULOG:     200
```

Slika 6. Simulacija igre i gubitak igrača

D. Igrač pobjeđuje

Na *slici 7*, prikazana je simulacija igre u kojoj je na kraju suma vrijednosti karata u igračevoj ruci bila veća od djeliteljeve sume, čime je igrač pobijedio u igri i osvojio iznos jednak ulogu.

```
player console
BALANS:   9800
ULOG:     200

(player@DESKTOP-ED8HE0U)1> INPUT:   start
(player@DESKTOP-ED8HE0U)1>
KARTE:           [2,9]
(player@DESKTOP-ED8HE0U)1>
SUMA:            {11,hard}
(player@DESKTOP-ED8HE0U)1>
DEALEROVA KARTA: 8

(player@DESKTOP-ED8HE0U)1> INPUT:   hit
(player@DESKTOP-ED8HE0U)1> NOVA KARTA:   "J"
(player@DESKTOP-ED8HE0U)1> KARTE:        ["J",2,9]
(player@DESKTOP-ED8HE0U)1> SUMA:         {21,hard}

(player@DESKTOP-ED8HE0U)1> INPUT:   stand
(player@DESKTOP-ED8HE0U)1>
STAND!

KARTE:  ["J",2,9]
(player@DESKTOP-ED8HE0U)1> SUMA:   21
(player@DESKTOP-ED8HE0U)1>
CEKAMO DEALEROVE KARTE ...

(player@DESKTOP-ED8HE0U)1>
DEALEROVA SUMA:  17
(player@DESKTOP-ED8HE0U)1>
VASA SUMA:       21
(player@DESKTOP-ED8HE0U)1>
POBJEDA!
(player@DESKTOP-ED8HE0U)1>
BALANS:   10000
ULOG:     200
```

Slika 7. Simulacija igre i pobjeda igrača

E. Zaključak

Koristeći programski jezik Erlang te funkcionalnosti gen_server modula, implementirali smo aplikaciju koja igračima omogućuje igranje kartaške igre Blackjack. U igri postoje dvije uloge: djelitelj, koji je implementiran kao modul poslužitelj, i igrač, koji ima ulogu klijenta. Za razliku od tradicionalne igre s fizičkim kartama, igrači ne moraju čekati red na potez, već svi igrači igraju svoj potez istovremeno, čime je trajanje jedne igre ubrzano približno n puta, gdje je n broj igrača koji sudjeluju u igri.

Htjeli smo da simulacija igre bude što vjerodostojnija tradicionalnoj igri, zbog čega smo napravili zajednički špil karata koji je univerzalan za sve igrače. Budući da je igraći špil pohranjen u djeliteljevom stanju (*engl*. state), onemogućen je višestruki istovremeni pristup špilu.

F. Budući rad

Kako bismo unaprijedili aplikaciju, prvi korak bio bi izrada grafičkog korisničkog sučelja, koje bi zamijenilo konzolu te igračima pružilo bolje korisničko iskustvo.

Trenutno ne postoji vremenski period unutar kojeg igrač mora odigrati svoj potez, no kako bi spriječili zastoje zbog neaktivnih igrača, planiramo postaviti vremenska ograničenja na trajanje poteza.

Trenutno je aplikaciju moguće pokrenuti na računalima unutar lokalne mreže (*engl*. LAN). Jedan od ciljeva za budućnost je omogućiti igračima da igraju zajedno neovisno o njihovoj lokaciji. Kako bismo to postigli, aplikaciju je potrebno objaviti na webu.

REFERENCES

[1] "Blackjack," https://www.en.wikipedia.org/wiki/Blackjack, pristup: 20. Siječanj 2024.

[2] "Erlang programming language," https://www.erlang.org, pristup: 21. Siječanj 2024.

[3] "Rebar3: Erlang build tool," https://www.rebar3.org/, pristup: 21. Siječanj 2024.

[4] "gen_server - erlang/otp," https://www.erlang.org/doc/man/gen_server.html, pristup: 21. Siječanj 2024.