SVEUČILIŠTE U ZAGREBU,
PRIRODOSLOVNO-MATEMATIČKI FAKULTET -
MATEMATIČKI ODSJEK

RAČUNARSKI PRAKTIKUM 2

Seminar - Briškula

Izradili:
Mirjana Jukić-Bračulj
Mia Filić
Gregor Boris Banušić
Ivan Laković

Profesor:
Doc. dr. sc. Zvonimir Bujanović

7. srpnja 2016.

Sadržaj

1. Uvod

Briškula (Briscola) je talijanska kartaška igra udomaćena na hrvatskom priobalju i otocima. Iznimno je popularna i lako se uči. Najčešće se igra s tršćanskim kartama (tal. Carte Triestine). U špilu ima 40 karata. Po istim osnovnim pravilima igra se obična i dupla briškula. Običnu igraju dva igrača ili 4 igrača od kojih svaki ima 3 karte. Dupla briškula se igra u dvoje. Svaki igrač ima 4 karte. Na kraju dijeljenja karata igračima okreće se jedna karta licem prema gore i na nju se stavlja preostali špil karata. Okrenuta karta se zove briškula i ona predstavlja tip (boju) karte u koju se igra. Postoje 4 tipa karata: kupe, baštone, špade i dinari. Karte su označene brojevima 1-7 i 11-13. Karte poredane po veličini, od najjače do najslabije su redom: as (1), trica (3), kralj (13), konj (12), fanta (11) i dalje od sedmice (7) prema dvici (2). As se broji kao 11 punata, trica 10, kralj donosi 4, konj 3, fanta 2, a ostale karte ne donose punte. Igru započinje igrač kojem su prvom podijeljene karte. Bačena karta predstavlja tip karte u koju se igra ta runda. Igrač s najjačom kartom te runde, kupi karte, prvi uzima novu kartu sa špila i započinje novu rundu. Prva bačena karta se može pobijediti jačom kartom istog tipa ili bilo kojom briškulom. U igri ukupno ima 120 punata, pobjednik je onaj tko ima više od 60. U slučaju da obje strane imaju 60 punata tada nema pobjednika. Igra se na 4 dobivene igre.

Napravili smo web verziju igre tako da smo serversku stranu implementirali u PHP-u korištenjem MVC arhitekture, a klijentsku stranu u JavaScript-u koristeći dodatne biblioteke jQuery i Bootstrap.

Igra je implementirana za 2 i 4 igrača, te ukoliko igraju 2 igrača može se igrati i dupla. Serverska strana pri komunikaciji s klijentima koristi json pozive pomoću ajax-a stoga je lako u budućnosti napraviti mobilnu aplikaciju koja će moći igrati zajedno s igračima koji igraju preko browsera.

2. O aplikaciji

Aplikacija služi korisnicima za online multiplayer igranje briškule. Igrači mogu birati s kim žele igrati, te pozivati prijatelje da igraju s njima. Osim toga igrači mogu izabrati žele li igrati u 2 ili 4 igrača te običnu ili duplu briškulu. Osim toga pamti se i omjer pobjeda i poraza tako da pri izabiranju protivnika možemo odabrati odgovarajućeg.

2.1. Prijava

Bilo tko se može registrirati tako da upiše korisničko ime (koji mora biti jedinstven), lozinku i valjanu e-mail adresu na kojoj će korisnik potvrditi račun. Račun prije potvrde nije aktivan te se na njega ne može prijaviti. Nakon potvrde računa korisnik se sa svojim podacima može prijaviti u igru.

2.2. Stvaranje igre

Nakon prijave na stranicu korisnik može čekati poziv u igru ili sam konstruirati sobu u koju će pozivati suigrače. Ukoliko čeka poziv bilo koja osoba u sobi može ga pozvati u sobu te on mora odgovoriti hoće li pristati ili ne. Ukoliko korisnik napravi ili uđe u sobu može pozivati aktivne (online) igrače koji još ne igraju te nisu u niti jednoj sobi. Nakon što se soba napuni igrači u sobi više ne mogu pozivati nove igrače, ali im se omogućuje konstrukcija igre. Ukoliko jedan od igrača konstruira igru, svim ostalim igračima u sobi se pokreće igra.

2.3. Igra

Sama igra je realizirana tako da što više liči pravom igranju briškule. Prilikom igranja svakog igrače se čeka maksimalno 2 minute da odigra potez, ukoliko ne odigra u tom vremenu gubi (tj. gubi njegov tim ukoliko se igra u 4). Osim toga igra je identična igri uživo. Na kraju igre svakom se igraču promjeni omjer pobjeda i odigralih ovisno o završetku igre, osim toga daje se mogućnost igračima za revanž (ponavljanje igre). Ukoliko igrači više ne žele međusobno igrati vraćaju se u stvaranje igre.

2.4. Odjava

Korisnik se u svakom trenutku može odlučiti odjaviti se, te tako napušta trenutnu i varaća se na početnu stranicu.

3. MVC

Projekt je organiziran po uzoru na Model-View-Controller arhitekturu.

3.1. Controller

Tijekom implementacije igre pojavila se je potreba za 3 controllera (izuzevši index controller koji preusmjerava na prijavu). Svaki controller se brine za jedan dio aplikacije, te odgovara na korisničke upite sukladno time. Tako imamo iduće controllere:

- Prijava
- Online
- Igra

3.1.1. Prijava

Ako korisnik nije prethodno ulogiran u aplikaciju, ispisuje se ekran koji mu omogućava stvaranje novog računa ili logiranje s već postojećim računom. Pri odabiru korisničkog imena kod registracije, provjerava se njegova duljina (mora biti duljine od 3 do 10 slova) i sprječava odabir već postojećeg korisničkog imena. Pri registraciji korisnik mora unijeti mail adresu na koju će dobiti mail kojeg mora potvrditi kako bi njegov račun postao aktivan. Logiranje se provodi tako da se iz baze usporedi da li uneseni username odgovara unesenoj lozinki. Ovaj controller se brine i o odjavi iz aplikacije, tako da se korisnika vrati na početni ekran za logiranje, a pripadni session se uništi.

3.1.2. Online

Provjerava se da li je korisnik ulogiran. Ako nije onda ga preusmjerimo na ekran za logiranje. Ako korisnik već sudjeluje u nekoj igri proslijedimo ga na ekran koji iscrtava stanje igre. Inače, ubaci korisnika u tablicu aktivnih ljudi i provjerava da li je korisnik kreirao novu sobu i sukladno tome šaljemo pripadnom view-u (create_game) podatak koji dio stranice trebamo iscrtati. Ako korisnik nije u niti jednoj sobi onda se nalazi u tablici aktivnih i ostali igrači ga mogu pozvati u sobu. Ulaskom u sobu korisnika se briše iz tablice aktivnih i postavlja ga se u jednu od tablica za sobe kao sudionika odgovarajuće sobe. Iz sobe je moguće izići, pozvati druge igrače u sobu ili, ako je odgovarajući broj igrača u sobi, pokrenuti igru. Komunikaciju između igrača omogućuju funkcije koje odgovaraju na ajax pozive od strane klijenta šaljući mu podatke zapisane iz tablica “onlines”, “sobe2”, “sobe4”.

3.1.3. Igra

Provjerava da li je korisnik ulogiran, ako nije preusmjeri ga na ekran za logiranje. Inače provjeravamo da li je korisnku u session postavljen id igre u kojoj sudjeluje. Ako nije onda pokušamo pročitati iz baze da li je igra u kojoj on sudjeluje već stvorena. Ako igra postoji onda iščitamo pripadni id igre inače stvorimo igru s imenima igrača koji su u korisnikovoj sobi. Na taj način se brine o tome da se ne stvore dvije igre za istu skupinu igrača. Sadrzi funkciju koja se brine o prosljeđivanju bacanja karata koje dobije

od klijenta preko ajax poziva modelima koji sukladno tome mijenjaju stanje igre u bazi. Osim toga, klijentu šalje podatke o trenutnom stanju igre kako bi se korisniku prikazali potezi protivnika, karte koje su mu na izboru za bacanje, stanje bodova...

3.2. Model

Tijekom implementacije igre pojavila se je potreba za 3 controllera (izuzevši index controller koji preusmjerava na prijavu). Svaki kontroler se brine za jedan dio aplikacije. Tako imamo iduće controllere:

- Db
- User
- UserService
- OnlineService
- Soba
- SobaService
- Briskula
- BriskulaService

3.2.1. Db

Klasa dizajnirana kao single pattern koja predstavlja vezu na bazu. Ima statičku funkciju koja uspostavlja tu vezu.

3.2.2. User

Klasa čuva osnovne podatke o korisniku: korisničko ime, lozinku, broj pobjeda i broj odgralih igara. Omogućava konstrukciju instanci klase. Koristi se za slanje polja svih aktivnih korisnika aplikacije klijentu kao odgovor na ajax poziv.

3.2.3. UserService

Klasa brine o provjeri ispravne prijave i registracije korisnika tako da usporedi podatke koje dobije od controllera i podatke koji se nalaze u bazi. Čita podatke iz iz tablice “onlines” za dobivanje svih aktivnih i slobodnih korisnika aplikacije.

3.2.4. OnlineService

Klasa koja brine o tablici “onlines”. Ubacuje korisnika koji se ulogirao u tablicu, izbacuje ga pri odjavi, provjerava da li je igrač pozvan u sobu tako da provjeri da li u pripadnom retku postavljen id_sobe. Ako igrač ne želi u sobu onda se ponuda briše, inače mu se pridružuje sastav sobe u koju je pozvan.

3.2.5. Soba

Klasa čuva podatke o sobi. Klasa se brine o ispravnosti funkcionalnosti: ubacivanje i izbacivanje igrača iz sobe, provjera da li su igrači spremni na igru i je li soba prazna.

3.2.6. SobaService

Klasa se brine o izmjenama tablica "soba2" i "soba4". Omogućava stavanje nove sobe, dodaje igrače u sobu i izbacuje ih, čita iz tablice sobu sa zadanim igracem i id-om.

3.2.7. Briskula

Klasa čuva podatke o jednoj igri briskule. Čuva podatke o broju igrača, stanju karti i bodova. Omogućava osvježavanje stanja bodova na kraju svake runde, podjelu novih karti igračima na kraju svake runde i obrađivanju poteza igrača.

3.2.8. BriskulaService

Klasa sprema izmjene stanja igre u tablicu "igre" te pronalazi i čita retke iz iste prema zadanom parametru (username, id).

3.3. View

U sklopu MVC arhitekture izradili smo 3 view-a koji po funkcionalnosti prate logiku aplikacije (controllere) i nekoliko pomoćnih koji se uvijek uključeni poput _header-a koji je zadužen za crtanje zaglavlja. Unutar view komponente nalaze se i dodatne slike i gifovi koji služe boljoj vizualizaciji i realističnijem dojmu igre.

- 404_index
- _footer
- _header
- create_game
- login
- playGame
- welcome_page

3.3.1. login

Login view se poziva na početku aplikacije i komunicira s prijava controllerom. Login je zadužen za iscrtavanje ekrana za logiranje u igru i registraciju novih korisnika.

3.3.2. create_game

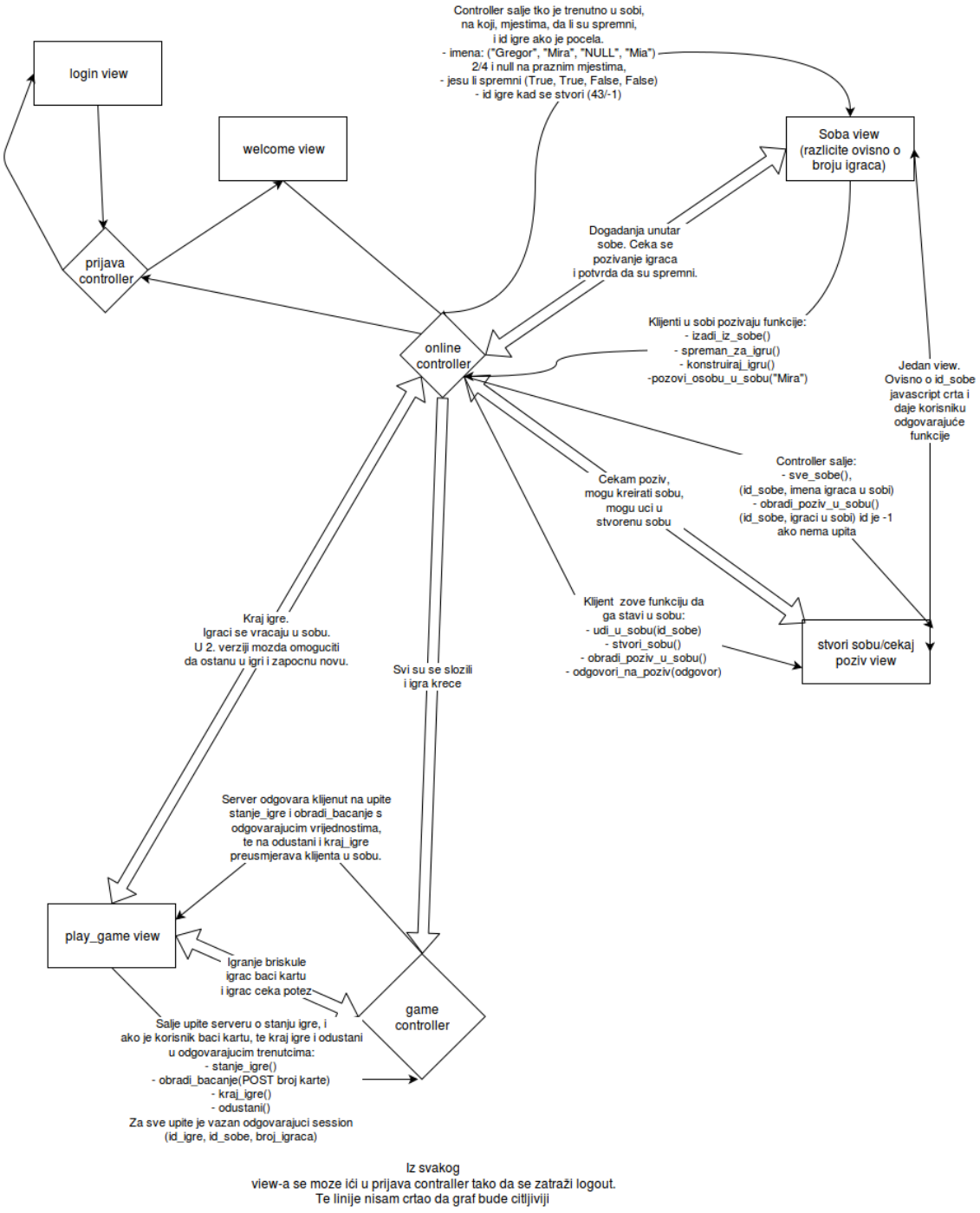
Create_game view vizualizira korisnicima trenutno stanje aktivnih igrača, te im omogućuje izradu sobe u koju će pozvati igrače i čekanje na poziv u igru od drugih igrača.

Create_game komunicira s online controllerom te ovisno o želji korisnika kontaktira neke njegove funkcije, te automatski osvježava trenutno stanje u sobi/lobiju i da li ima zahtjev za pridruživanje igri. U slučaju da je igrač u sobi, osvježava se trenutno stanje u sobi te ukoliko ima slobodnih mjesta omogućuje se pozivanje slobodnih igrača, te kad se sva mjesta popune, ako su svi igrači spremni moguće je pokrenuti igru.

3.3.3. playGame

View playGame je ključni dio aplikacije i on je odgovoran za samu igricu, to jest za njezino igranje. View pomoću javascripta (bootstrap, i jquery) dohvaća trenutno stanje u igri od servera komunicirajući s igra controllerom te pomoću malo logike (javascript) crta trentno stanje igre i omogućava igraču igranje igre (bacanje karata). PlayGame na omogućava predaju, a ukoliko se igra završi ispisuje rezultat i pobjednika te varaća igrače natrag u sobu (create_game) view.

4. Komunikacija unutar aplikacije



5. Zaključak

Uz korištenje uzastopnih ajax poziva ostvarili smo komunikaciju među igračima te tako postigli multplayer igranje. Korisnici mogu odabrati hoće li igrati u paru ili u igri s 4 igrača. Po ulasku u igru iscrtava se popis svih aktivnih igrača i omogućuje stvaranje sobe. Po ulasku u sobu korisnici mogu pozivati druge igrače u sobu. Kada se popuni broj mjesta u sobi, moguće je pokrenuti igru.
Postoji mogućnost dodavanja još jedne funkcionalnosti, a to je da svaki igrač mora potvrditi da li je spreman na igru. Pri pokretanju igre potrebno je paziti da se stvori točno jedna igra. Korisnicima se iscrtavanju karte. Odabir karte za bacanje postiže se klikom na kartu, koja se potom iscrtava na polju za bačene karte. Ovaj dio se može poboljšati uvođenjem nekog oblika komunikacije među igračima, npr. chat prozora. Na kraju igre proglašava se pobjednik i bilježi se njegova pobjeda. Igrači se vraćaju u sobu iz koje je stvorena njihova igra.
Veliki dio podataka se od servera dobiva u obliku json stringova pa se klijentski dio aplikacije može lako implementirati i u drugim tehnologijama, npr. android.