SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**Seminar**

Škola košarke

Ivana Vanjak

0036460333

Zagreb, veljača 2015.

***Opis problema***

Administracija Škole košarke želi unaprijediti kvalitetu usluge za svoje članove tako što će izraditi informacijski sustav koji će im omogućiti praćenje treninga, utakmica i evidenciju članova. Škola košarke broji pedesetak aktivnih članova i treneru je postalo preteško samostalno voditi bilješke o svim igračima.

Članovi škole košarke su učenici osnovne škole. Prilikom upisa nekog djeteta uzimaju se njegovi podaci i uz to podaci o roditeljima/skrbnicima.

Škola košarke ima više različitih grupa za treninge. Ovisno o godištu igrača i njegovom napretku, trener odlučuje u koju grupu pripada. Na svakom treningu potrebno je provoditi evidenciju igrača i voditi zabilješke o njihovom napretku. To je potrebno da bi roditelji prilikom posjeta treneru mogli dobiti širi uvid u igrački razvoj svog djeteta, te kako bi trener prilikom savjetovanja igrača mogao imati pregled najbitnijih stavki na proteklim treninzima.

Igrači sudjeluju na raznim natjecanjima. Natjecanje može biti liga, turnir ili samo prijateljska utakmica. Potrebno je voditi evidenciju o rezultatima svih utakmica i statistici svih igrača na njima.

Također je važno imati podatke o protivničkim klubovima s kojima se igrači susreću, kao što su adresa i kontakt, kako bi sve potrebne stvari prilikom organizacije utakmice bile dostupne unutar aplikacije.

***Rješenje problema***

Središte problema su igrači koji sudjeluju na natjecanjima i pohađaju treninge. To nam daje najvažnije elemente domene. Aplikacija će biti organizirana oko entiteta *Igrač*. *Natjecanje* i *Trening.* Za svakog igrača pratit će se na kojim je treninzima bio, na kojim natjecanjima je sudjelovao te njegov učinak na njima.

Kako bi se olakšalo praćenje napretka, omogućit će se vođenje bilješki kako bi trener mogao pratiti ponašanje svakog igrača posebno.

Omogućit će se i praćenje plaćanja članarina. Moći će se definirati članarina za određeni mjesec, te označiti koji igrači su je platili. Prikazivat će se oni igrači koji je nisu platili.

Pratit će se i podaci o protivničkim klubovima.

***Use cases***

1. Upisivanje novog člana u klub

2. Uplata mjesečne članarine

3. Evidencija članova na treninzima i vođenje bilješki o igračima

4. Dodavanje novog natjecanja

5. Dodavanje nove utakmice

6. Dodavanje novog protivničkog kluba

7. Unos statistike za igrača na nekoj utakmici

8. Pregledavanje statistike za nekog igrača na nekoj utakmici

9. Pregledavanje svih utakmica na nekom natjecanju

***Use case 1: Upisivanje novog člana u klub***

***Primary actor:*** Administracijski djelatnik

***Stakeholders:***

*Administracijski djelatnik:* Želi uvid u broj upisanih članova.

*Trener:* Želi uvid u broj članova kako bi znao organizrati treninge i kako bi pojedinog igrača mogao prijavljivati za utakmice.

*Roditelj:* Želi upisati dijete.

*Igrač:* Želi trenirati.

***Precondition:*** Dijete je učenik osnovne škole.

***Postcondition:*** Igrač je upisan u sustav.

***Glavni uspješni scenarij:***

1. Roditelj dolazi upisati dijete.

2. Administracijski djelatnik unosi podatke o djetetu, budućem igraču.

3. Ovisno o godištu, odbire početnu grupu i termin treninga.

4. Igrač je upisan u sustav.

5. Trener na prvom treningu utvrđuje da je igrač u dobroj grupi.

6. Igrač nastavlja trenirati.

***Alternativni scenarij:***

5a. Trener na prvom treningu uviđa da igrač nije u dobroj grupi.

1. Trener mijenja grupu kojoj igrač pripada.

***Use case 4: Evidencija članova na treninzima i vođenje bilješki***

***Primary actor:***Trener

***Stakeholders****:*

*Trener* – Prilikom savjetovanja igrača želi imati pregled najbitnijih stavki na preteklim treninzima.

*Administracija* – Žele znati je li netko izostao sa svih treninga u mjesecu kako bi ga oslobodili od plaćanja članarine.

*Roditelji* – Žele imati uvid u broj treninga na kojima im je dijete bilo i u igrački razvoj svog djeteta.

*Igrač –* Želi uvid u svoj napredak i savjete od trenera.

***Preconditon:*** Igrač je upisan u klub.

***Postcondition:*** Igraču je evidentiran dolazak na trening.

***Glavni uspješni scenarij:***

1. Trener evidentira prisutnost igrača na treningu.

2. Nakon treninga upisuje bilješke za pojedinog igrača.

3. Podaci se pohranjuju u sustav.

***Use case 6: Dodavanje nove utakmice***

***Primary actor:*** Trener

***Stakeholders:***

*Trener –* Želi imati pregled utakmica koje je njegov tim odigrao.

*Roditelji –* Žele vidjeti koliko je utakmica njihovo dijete odigralo.

***Precondion:*** Utakmica je odigrana.

***Postcondition:*** Utakmica je spremljena u sustav sa svim podacima.

***Glavni uspješni scenarij:***

1. Trener odabire natjecanje kojem utakmica pripada.

2. Trener odabire klub protiv kojeg je njegov tim igrao.

3. Trener odabire igrače koji su sudjelovali u toj utakmici.

4. Trener unosi rezultat.

***Alternativni scenarij:***

1a. Nije definirano natjecanje kojem utakmica pripada.

1. Trener izvodi slučaj korištenja 5. “Dodavanje novog natjecanja”.

2a. Nije definiran protivniči klub.

1. Trener izvodi slučaj korištenja 7. “Dodavanje novog protivničkog kluba”.

***Use case 8: Unos statistike za igrača na nekoj utakmici***

***Primary actor:*** Trener

***Stakeholders:***

*Trener:* Želi imati uvid u rezultate pojedinca i pratiti napredak.

*Roditelj:* Želi imati uvid u rezultate i napredak svog djeteta.

*Igrač*: Želi imati informaciju od trenera o svom napretku.

***Precondition:*** Igrač je sudjelovao u utakmici.

***Postcondition:*** Podaci o statistikama su pohranjeni u sustavu.

***Glavni uspješni scenarij:***

1. Trener odabire utakmicu za koju unosi statistiku.

2. Trener odabire igrača za kojeg unosi statistiku.

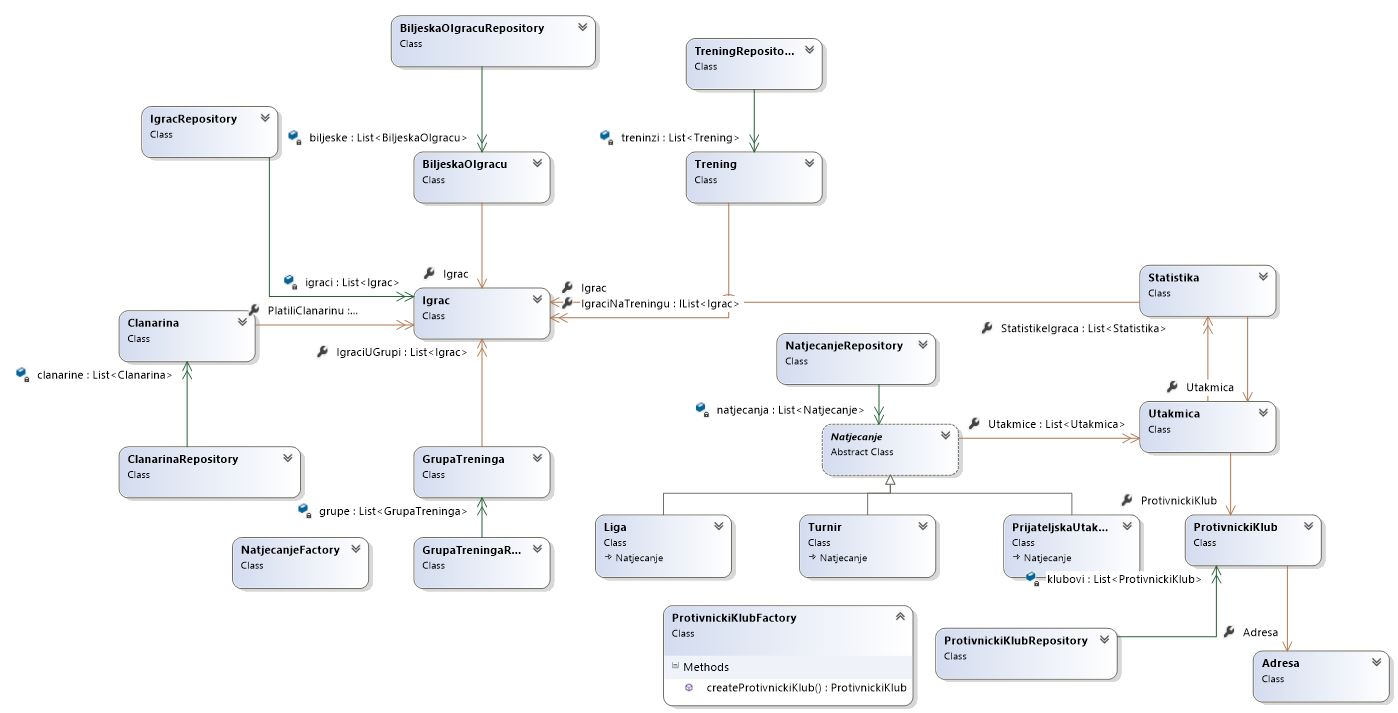
3. Unosi statistiku.

***Alternativni scenarij:***

1a. Nije definirana utakmica.

1. Trener izvršava slučaj korištenja 6. “Dodavanje nove utakmice”.

***Opis objektnog modela***

**Slika 1. Dijagram razreda**

Model sustava je odgovoran za reprezentiranje osnovnih koncepata sustava, informacija o njegovom stanju i skupa njegovih pravila. On opisuje i kontrolira stanje objekata. Da bi arhitektura mogla podržavati domain-driven dizajn, potrebno je postići izolaciju domain-related koda na način koji dopušta kohezivan dizajn domene koji je slabo vezan s ostatkom sistema. Kreiramo cjelokupni sloj objekata koji modeliraju poslovnu domenu softvera. Neki objekti predstavljaju podatke, a neki poslovna pravila u domeni, uz primjenu standardnih principa objektno orijentirane paradigme. Model izgleda slično kao model baze podataka.

Kao središte interesa u ovoj aplikaciji imamo igrače koji sudjeluju na raznim natjecanjima i pohađaju treninge. Igrači su predstavljeni entitetom u kojem su pohranjene sve osnovne informacije o samom igraču. Igračima se upravlja kroz repozitorij u kojem postoji lista svih igrača koji su trenutno u sustavu. On ima sve metode potrebne za kreiranje i dohvaćanje igrača. *IgracRepository* je *sigleton*, što znači da u svakom trenutku postoji samo jedna instanca te klase.

Kroz entitet *Trening* pratimo koji su igrači bili na pojedinom treningu. Treninzi se mogu odvijati u više grupa,, pa igrače na treningu biramo ovisno o grupi igrača koja ima određeni trening. U entitetu *GrupaTreninga* imamo listu igrača koji joj pripadaju. Za svakog igrača se mogu voditi bilješke koje će omogućiti prikaz napretka igrača. Treninzima, grupama treninga i bilješkama se također upravlja kroz istoimene repozitorije koji su realizirani isto poput repozitorija igrača.

Natjecanje je prikazano kroz hijerarhiju objekata. Kao apstraktnu klasu imamo entitet *Natjecanje* kojeg zatim nasljeđuju *Liga*, *Turnir* i *PrijateljskaUtakmica*. Uz atribute *Natjecanja*, *Liga* i *Turnir* definiraju dodatno svoje atribute.

*Natjecanje* ima listu objekata *Utakmica*. Svakom natjecanju se pridružuju utakmice koje su se održale na tom natjecanju, za koje se pamte ukupan rezultat i protivnički klub. Za svaku utakmicu definiramo skup igrača koji su na njoj prisustvovali kroz klasu *Statistika. Statistika* ima referencu na igrača za kojeg se piše statistika i referencu na utakmicu na kojoj je postigao statistiku te podatke o koševima, skokovima, asistencijama i osobnim pogreškama. Natjecanja, utakmice i statistike se prate u repozitoriju *NatjecanjeRepository* u kojem su defnirane sve metode potrebne za dodavanje i dohvaćanje svakog od njih. Tamo imamo listu natjecanja, preko koje se zatim traže utakmice, a preko utakmica statistike. Konkretno natjecanje se kreira preko *NatjecanjeFactory* klase, koja pruža statičke metode za izradu natjecanja. Pošalje joj se konkretni tip i potrebni parameri, a ona vraća objekt tipa *Natjecanje*.

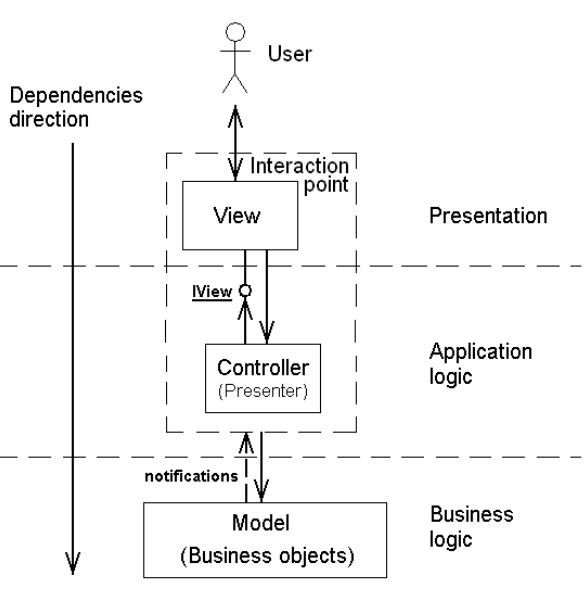
U aplikaciji je potrebno pratiti i podatke o protivničkim klubovima. U entitetu *ProtivnickiKlub* pratimo sve podatke o klubovima, poput naziva, telefonskog broja te adrese koja je implemetirana kao vrijednosni objekt *Adresa.* Protivničkim klubovima se pristupa preko pripadnog repozitorija. Objekti se stvaraju preko *ProtivnickiKlubFactory* razreda.

U sustavu se još prati plaćanje članarina. Definiranjem mjeseca, godine i vrijednosti članarine opisana je jedna mjesečna članarina. U tom entitetu se još prate svi igrači koji su je platili. Članarine se prate kroz repozitorij.

***Arhitektura/GUI***

Aplikacija je ostvarena kroz MVP obrazac koji omogućava odvajanje sučelja i podataka. MVP je varijanta MVC-a. U MVC-u imamo tri uloge: *Model*, *View* i *Controller*. *Model* sadrži podatke koji predstavljaju problem, *Controller* odgovara na korisnikove akcije govoreći *Modelu* kako da se promijeni i komunicira s *View*-om koji te podatke prikazuje.

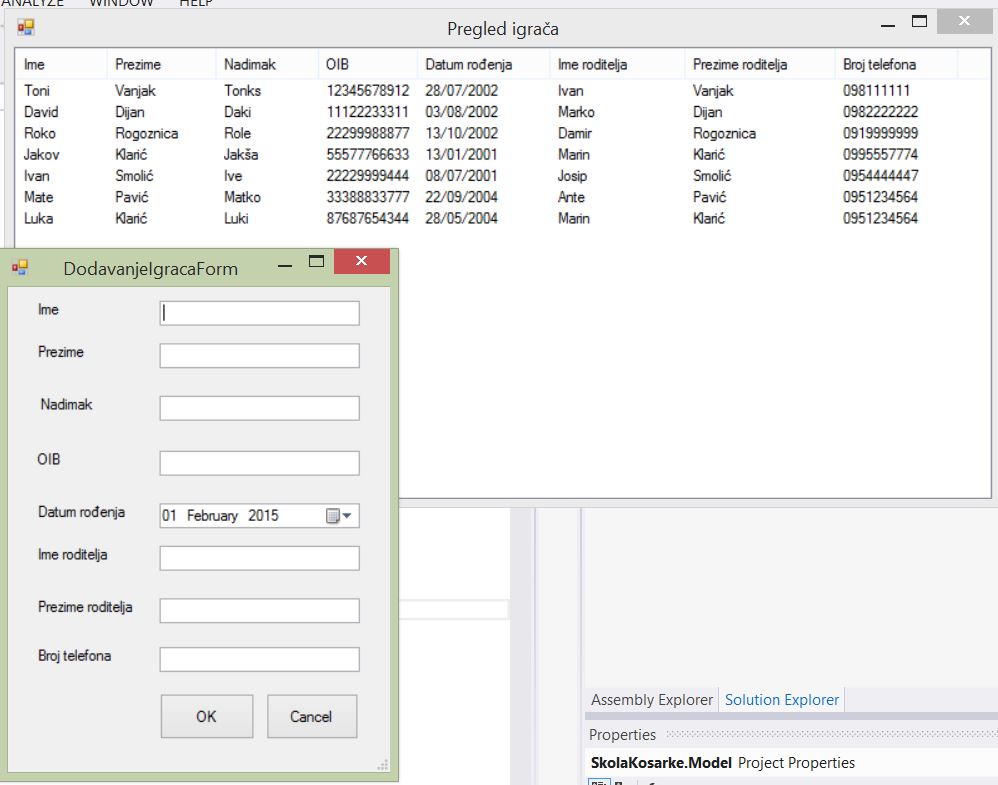
U MVP-u *View* ima direktnu referencu na *Controllera*(*Presentera*), a *Controller* s *View*-om komunicira preko sučelja *IView*, ne ovisi o konkretnom *View*-u. *Controller* ima referencu na *Model*.



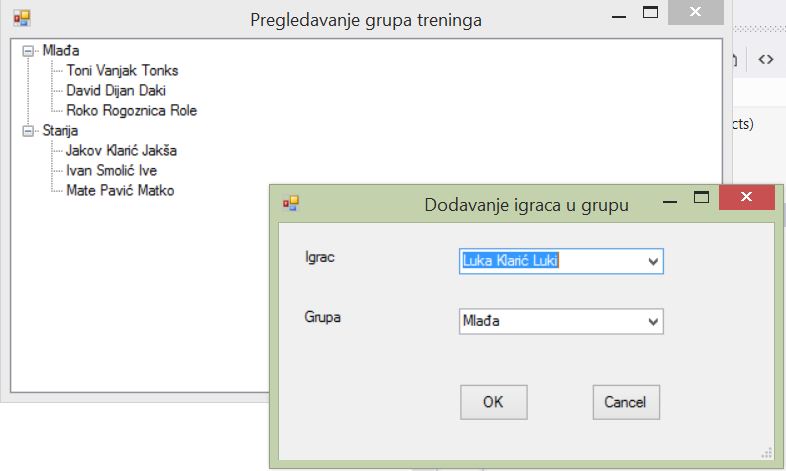
*View* prima korisnikove upite i samo ih prosljeđuje *Controlleru*, koji ih izvodi u suradnji s *Modelom*. *Controller* preko *IView* sučelja poziva metode *View*-a koji samo prikazuje podatke bez komunikacije s *Modelom*.

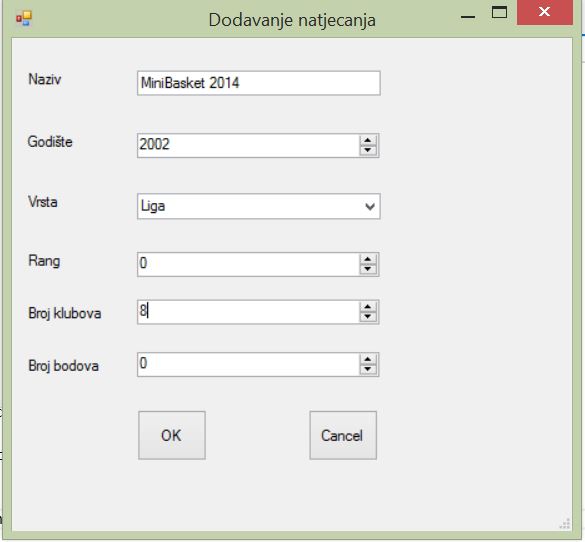
Promjena u *Modelu* se javlja *Viewu* preko *Observer* obrasca. *Controller* je *Subject*, a forme *Observeri*. *Controller* prilikom prilikom pozivanja neke metode iz *Modela* kojom se *Model* mijenja, obavještava sve forme koje ga prate, tj. koje su u njegovoj listi *Observera*. Forma prilikom osvježavanja dohvaća najnovije podatke preko *Controllera* i prikazuje ih.

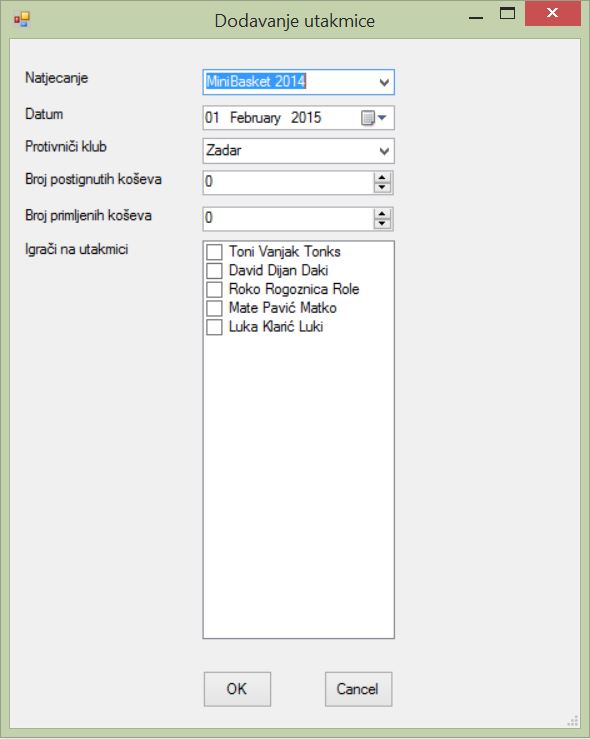
Kad korisnik odabere neku formu na alatnoj traci, u *MainView*-u se pozove odgovarajuća metoda *Controllera.* Ukoliko je to forma koja će zahtijevati osvježavanje nakon promjene modela, *Controller* je dodaje u listu svojih *Observera*. Preko *IView* sučelja, govori formi da se prikaže. Forma zatim preko direktne reference na *Controllera*, pozva njegove metode za dohvat podataka. Controller preko reference na određeni repozitorij dohvaća podatke i prosljeđuje ih formi koja ih zatim prikazuje.

Pregled i dodavanje igrača. Kad se igrač doda, ažuriraju se otvorene forme za prikaz igrača.

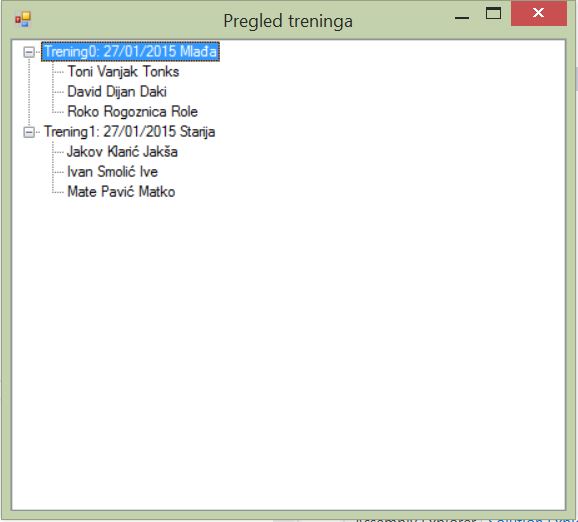
Isto vrijedi i kod svih ostalih formi.

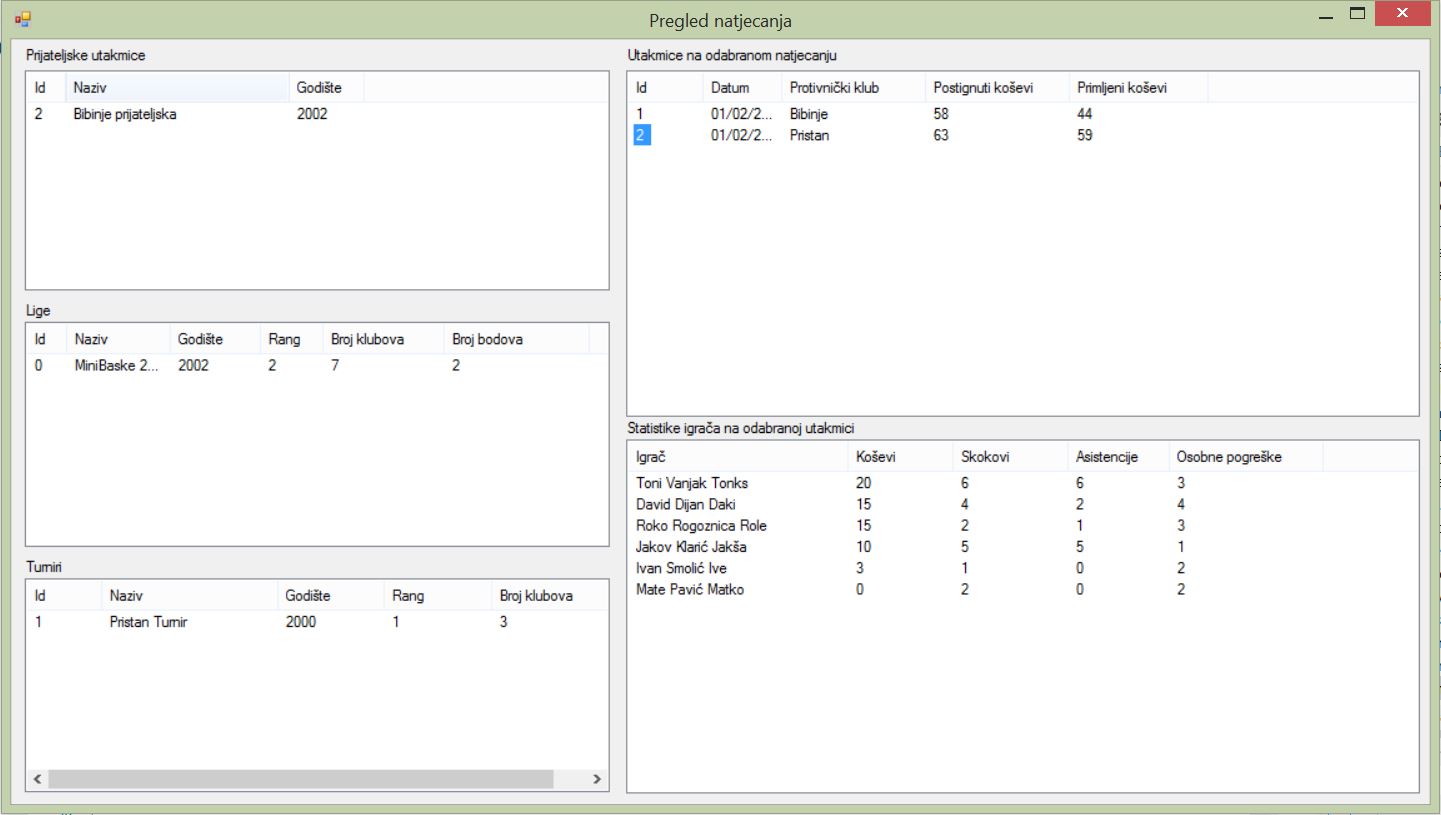


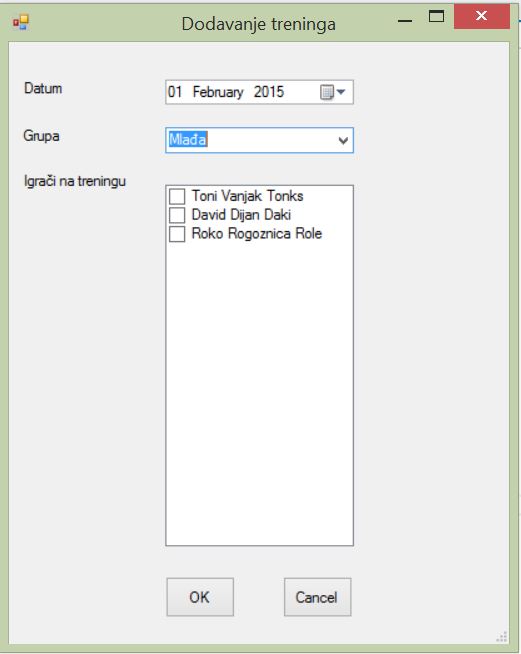




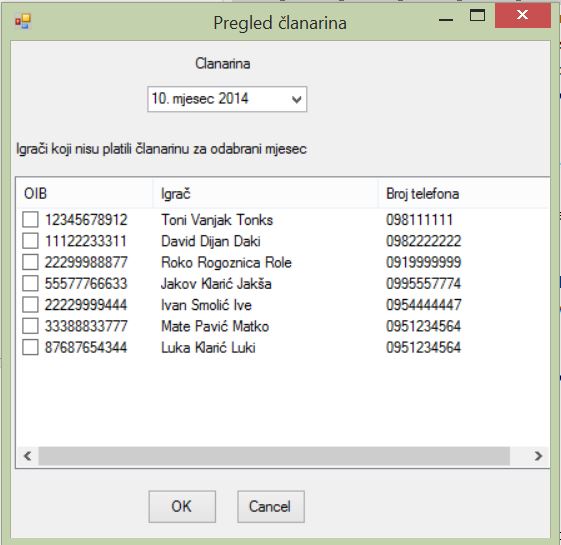
Prikazuju se samo igrači koji su mlađi ili jednaki od godišta koje je definirano kod stvaranja natjecanja na kojeg se dodaje utakmica. Natjecanje se bira u prvom *combobox*-u. Kod promjene vrijednosti *combobox*-a, mijenja se lista igrača koji se prikazuju.







Prilikom odabira grupe u *combobox*-u, prikazuju se samo oni igrači koji pripadaju odabranoj grupi



Ako se doda nova članarina dok je ova forma za pregled otvorena, padajući izbornik se odmah osvježi te sadrži i novu vrijednost. Odabirom *checkboxa* za pojedinog igrača i pritiskom na OK gumb, označava se da je igrač platio odabranu članarinu. Forma se osvježava i više ga ne prikazuje na popisu.