

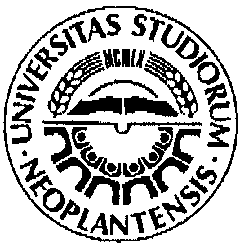
**UNIVERZITET U NOVOM SADU**

**PRIRODNO-MATEMATIČKI**

**FAKULTET**

**DEPARTMAN ZA MATEMATIKU**

**I INFORMATIKU**



Filip Vasić, 50/19

Aplikacija za pregled i vođenje statistike jednog

košarkaškog kluba tokom jedne sezone

- seminarski rad iz predmeta Skript jezici-

Novi Sad, 2020.

**Sadržaj**

[Sadržaj 2](#_Toc29296161)

[1.Uvod 3](#_Toc29296162)

[2.Opis programa 4](#_Toc29296163)

[Prijava referenata 4](#_Toc29296164)

[Unos novog igrača 6](#_Toc29296165)

[Unošenje statistike igrača 6](#_Toc29296166)

[Dodavanje novih utakmica 7](#_Toc29296167)

[Prikaz igrača u klubu 7](#_Toc29296168)

[Prosečna statistika igrača 8](#_Toc29296169)

[Pregled statistike sa utakmice 9](#_Toc29296170)

[Pregled odigranih utakmica 9](#_Toc29296171)

[Prikaz grafikona “pobede-porazi” 10](#_Toc29296172)

[Prikaz grafikona “igrač-indeks korisnosti” 11](#_Toc29296173)

[3.Zaključak 13](#_Toc29296174)

# Uvod

Za izradu programa koristio sam programski jezik *Python* u editoru *Spyder*.

Program služi za vođenje statistike igrača neke košarkaške ekipe tokom jedne sezone kroz odigrane utakmice. Program se sastoji iz pet zasebnih programskih fajlova i četiri tekstualna fajla.

U tekstualnim fajlovima se čuvaju podaci o referentima, odigranim utakmicama, košarkašima i njihovoj statistici.

U fajlu **referenti.txt** čuvaju se podaci o referentima i oni se u sklopu programa ne mogu menjati, u njemu se nalaze imena i prezimena zaposlenih referanata kao i njhova korisnička imena i lozinke za pristup programu.

Marko|Markovic|marko|123abc

Izgled linije tekstualnog fajla sa podacima referenata

U fajlove **igraci.txt**, **utakmice.txt**, **statistika.txt** će se upisivati podaci uneti u programu od strane referenata.

Program se pokreće iz **glavni.py** i čita podatke iz ostalih programskih fajlova **Igraci.py**, **Statistika.py**, **Utakmice.py**, **Referenti.py**, koji su povezani sa tekstualnim fajlovima. U tim fajlovima se već nalaze neke jednostavne funkcije koje upravljaju tekstualnim dokumetima, funkcije u njima su uglavnom funkcije dodavanja unetih podataka i čuvanja u listi, kao i pretrage nekih podataka potrebnih za izvršavanje funkcija u glavnom program.

Te funkcije će biti često pozivane zbog dodavanja podataka u liste.

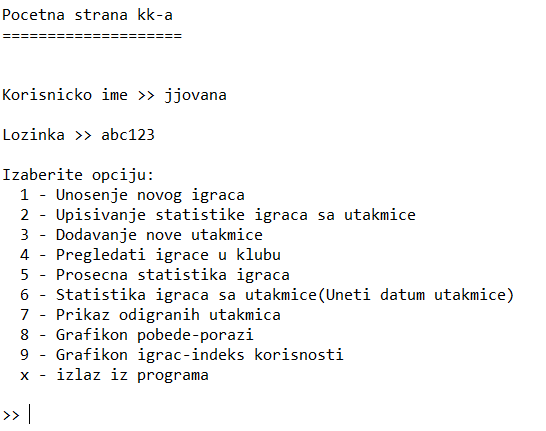
U programu će biti prikazano kako se mogu odraditi neke osnovne operacije programa koje svaki košarkaški klub mora da zbog vođenja učinka svojih igrača u sezoni.

# Opis programa

Program se sastoji iz dela za prijavu referenata i opcija koje oni mogu da izvršavaju.

Referenti imaju opcije da dodaju nove igrače i utakmice i beleže statistiku igrača sa utakmice u neku bazu podataka, mogu da izlistavaju sve igrače u klubu, njihovu prosečnu statistiku tokom sezone kao i statistiku sa jednog meča, imaju uvid u odigrane utakmice i u broj pobeda i poraza kao i uspešnost igrača tokom sezone koja se meri prosečnim indeksom korisnosti.

Na slici 1 prikazano je kako izgleda početni meni programa nakon prijave referenta koji treba da obavi neku od ponuđenih operacija.



Slika 1. Izgled menija nakon prijave referenta u program

## Prijava referenata

Pri pokretanju programa otvara se prozor za prijavu referenata u kojem referent unosi svoje korisničko ime i lozinku, iz **referenti.txt**, za pristup programu.

U listingu 1 prikazan je deo koda koji inicijalizuje početnu stranu za prijavu i dalje izvršavanje toka programa u zavisnosti od unosa ispravnog ili neispravnog korisničkog imena i lozinke.

U listingu 2 prikazana je funkcija za prijavu referenta. Funkcija je pozvana prilikom pokretanja programa a njen zadatak je da pozove funkciju za prijavu iz fajla **Referenti.py** koja potom proverava poklapanje unetih podataka sa podacima u tekstualnom dokumentu i vraća vrednost **True** ili **False** u zavisnosti da li su uneti podaci ispravni.

Ukoliko referent unese pogrešno tj. nepostojeće ime ili lozinku vraća se vrednost **False**, prekida se izvršavanje programa i štampa poruka “Niste uneli postojece ime i lozinku!”. Kada su uneti ime i lozinka ispravni vraća se vrednost **True** i nastavlja se dalje izvršavanje programa, otvara se novi meni iz kojeg referent bira redni broj (od 1 do 9 ili X za prekid rada programa) u zavisnosti koju operaciju želi da izvrši.

def **main**():

print()

print( "Pocetna strana kk-a")

print( "====================")

print()

if not login():

print( "\nNiste uneli postojece ime i lozinku!")

return

komanda = '0'

while komanda != 'X':

komanda = menu()

if komanda == '1':

addPlayer()

elif komanda == '2':

addStats()

elif komanda == '3':

addGame()

elif komanda == '4':

printAllPlayers()

elif komanda == '5':

averageStats()

elif komanda == '6':

playersStats()

elif komanda == '7':

printGames()

elif komanda == '8':

winlose()

elif komanda == '9':

playerIndexRating ()

print( "Dovidjenja.")

Listing 1. Funkcija kojom počinje izvršavanje programa

def **login**():

username = input("Korisnicko ime >> ")

password = input("Lozinka >> ")

return Referenti.login(username, password)

Listing 2. Funkcija za prijavu referenata

## Unos novog igrača

Kada referent izabere opciju ‘1’ tj. opciju za dodavanje novog igrača u tim, poziva se funkcija koja je prikazana u listingu 3.

Referent unosi ime, prezime, datum rodjenja, visinu i težinu igrača. Uneti podaci se čuvaju kao lista u fajlu **igraci.txt** odakle se kasnije pozivaju za neke nove naredbe.

Utakmica se dodaje pomoću poziva funkcija za dodavanje i čuvanje podataka koje se nalaze u fajlu **Igraci.py**.

def **addPlayer**():

print("Unos novog igraca\n")

i = {}

i['ime'] = input("Unesite ime >> ")

i['prezime'] = input("Unesite prezime >> ")

i['datum'] = input("Unesite datum rodjenja >> ")

i['visina'] = input("Unesite visinu igraca u centimetrima >> ")

i['tezina'] = input("Unesite tezinu igraca u kilogramima >> ")

Igraci.addPlayer(i)

Igraci.savePlayer()

Listing 3. Funkcija za dodavanje novog igrača u listu

## Unošenje statistike igrača

Kada referent izabere opciju ‘2’ tj. opciju za upisivanje statistike igrača sa utakmice, poziva se funkcija prikazana u listingu 4.

Referent unosi datum kada je utakmica odigrana, ime i prezime igrača, broj poena, asistencija, skokova, blokada, ukradenih lopti i faulova koje je igrač na tom meču zabeležio. Uneti podaci se čuvaju kao lista u fajlu **statistika.txt** odakle se kasnije pozivaju za neke druge operacije nad njima.

Utakmica se dodaje pomoću poziva funkcija za dodavanje i čuvanje podataka koje se nalaze u fajlu **Statistika.py**.

def **addStats**():

print("Unos statistike igraca sa utakmice \n")

s = {}

s['datum'] = input("Unesite datum utakmice(DD.MM.YYYY.) >> ")

s['igrac'] = input("Unesite ime i prezime igraca>> ")

s['poeni'] = input("Unesite broj poena igraca >> ")

s['asistencije'] = input("Unesite broj asistencija igraca >> ")

s['skokovi'] = input("Unesite broj skokova igraca >> ")

s['blokade'] = input("Unesite broj blokada igraca >> ")

s['ukrlopte'] = input("Unesite broj ukradenih lopti igraca >> ")

s['faulovi'] = input("Unesite broj faulova igraca >> ")

Statistika.addStats(s)

Statistika.saveStats()

Listing 4. Funkcija za upisivanje statistike

## Dodavanje novih utakmica

Kada referent izabere opciju ‘3’ tj. opciju za dodavanje nove utakmice, poziva se funkcija prikazana u listingu 5.

Referent unosi datum kada je utakmica odigrana, protivnika i ishod utakmcie(w – za pobedu ili l – za poraz). Uneti podaci se čuvaju kao lista u fajlu **utakmice.txt** odakle se kasnije pozivaju za neke druge operacije nad njima.

Utakmica se dodaje pomoću poziva funkcija za dodavanje u čuvanje podataka koje se nalaze u fajlu **Utakmice.py**.

def addGame():

print("Dodavanje nove odigrane utakmice \n")

g = {}

g['datum'] = input("Unesite datum utakmice(DD.MM.YYYY.) >> ")

g['protivnik'] = input("Unesite protivnika >> ")

g['ishod'] = input("Unesite ishod utakmice(w za pobedu, l za poraz >> ")

Utakmice.addGames(g)

Utakmice.saveGame()

Listing 5. Funkcija za dodavanje nove utakmice

## Prikaz igrača u klubu

Kada referent izabere opciju ‘4’ tj. opciju za prikaz svih igrača koji se trenutno nalaze u klubu, poziva se funkcija prikazana u listingu 6.

Funkcija pretrazuje sve redove u tekstualnom dokumentu u kojem se čuvaju podaci o igračima i u konzoli ispisuje sve prethodno unete podatke o igračima sa adekvatnim zaglavljem na slici 2 prikazano je kako izgleda izvršenje ove funkcije za neku prethodno formiranu bazu podataka o igračima.

Funkcija radi pomoću for petlje u kojoj se vrši štampanje potrebnih podataka, dok se zaglavlje štampa pre ulaska u petlju pomoću poziva funkije iz fajla **igraci.py**.

def **printAllPlayers**():

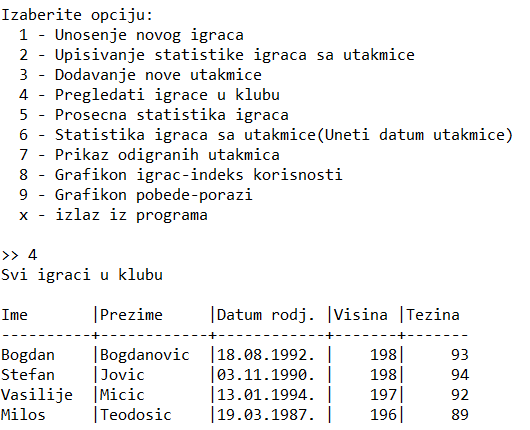
print("Svi igraci u klubu \n")

print(Igraci.formatHeader())

for igrac in Igraci.igraci:

print(Igraci.formatPlayer(igrac))

Listing 6. Funkcija za prikaz igrača u klubu



Slika 2. Primer prikaza igrača nekog kluba

## Prosečna statistika igrača

Kada referent izabere opciju ‘5’ tj. opciju za prikaz prosečne statistike igrača tokom cele sezone, poziva se funkcija prikazana u listingu 7.

Funkcija štampa zaglavlje tako što poziva funkciju iz fajla **Statistika.py** koja u sebi ima već ispisano zaglavlje pa potom pravi novu listu “allplayers” u kojoj se nalaze imena i prezimena svih igrača u klubu, samo jednom. Pomoću for petlje ulazi se u formiranu listu i računa prosečna statistika za svakog igrača po sledećem principu:

Formira se nova lista “playerstats” pomoću poziva funkcije za pretragu igrača iz **Statistika.py**. U listi se nalazi statisitika igrača iz liste “allplayers” sa svih utakmica, potom se za svaku kategoriju računa prosek tako što se sabiraju sve vrednosti iz te kategorije pa dele sa brojem tih vrednosti.

**Objašnjenje na primeru poena:**

Pravi se nova lista “poeni”. U napravljenu listu pomoću for petlje upisuju se svi poeni igrača za kojeg se u tom trenutku vrši račun. Kada je lista formirana tj. for petlja stigne do kraja uvodimo promenljivu “prosek\_poena” koja dobija vrednost sume svih poena iz liste “poeni” podeljenih sa brojem elemenata liste. Princip je isti za svaku narednu kategoriju.

Kada se izračuna prosek svih kategorija (poeni, asistencije, skokovi, blockade, ukradene lopte, faulovi) svi se štampaju u tabelu zajedno sa imenom igrača, a izvršavanje funkcije se nastavlja za sledećeg igrača.

Na slici 3 prikazano je kako izgleda izvršenje funkcije za neku bazu podataka.

def **averageStats**():

print("Prosecna statistika igraca tokom sezone \n")

print(Statistika.formatHeader())

allplayers = set([i['igrac'] for i in Statistika.statistika])

for igrac in allplayers:

playerstats = Statistika.findPlayer(igrac)

poeni = [int(i['poeni']) for i in playerstats]

prosek\_poena = sum(poeni)/len(poeni)

asist = [int(i['asistencije']) for i in playerstats]

prosek\_as = sum(asist)/len(asist)

skok = [int(i['skokovi']) for i in playerstats]

prosek\_skok = sum(skok)/len(skok)

blok = [int(i['blokade']) for i in playerstats]

prosek\_blok = sum(blok)/len(blok)

ul = [int(i['ukrlopte']) for i in playerstats]

prosek\_ul = sum(ul)/len(ul)

faul = [int(i['faulovi']) for i in playerstats]

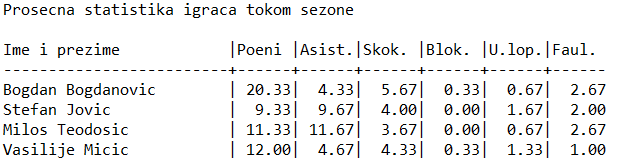
prosek\_faul = sum(faul)/len(faul)

print("{0:25}|{1:>6.2f}|{2:>6.2f}|{3:>6.2f}|{4:>6.2f}|{5:>6.2f}|

{6:>6.2f}".format(igrac,prosek\_poena,prosek\_as,

prosek\_skok,prosek\_blok,prosek\_ul,prosek\_faul))

Listing 7. Funkcija za štampanje prosečne statistike igrača u sezoni



Slika 3. Primer tabele sa prosečnom statistikom igrača

## Pregled statistike sa utakmice

Kada referent izabere opciju ‘6’ tj. opciju za prikaz statistike igrača sa neke utakmice, poziva se funkcija prikazana u listingu 8.

Referent unosi datum utakmice kao ulazni podatak. Funkcija poziva drugu funkciju iz fajla **Statistika.py** koja proverava da li se uneti datum nalazi u bazi podataka, ukoliko se pronađe poklapanje vraća se vrednost **True** odnosno **False** ako nije pronađeno poklapanje.

Ako je vraćena vrednost **False** funkcija štampa “Za uneti datum ne postoji odigrana utakmica.”, ako je vraćena vrednost **True** nastavlja se izvršavanje programa, štampa se zaglavlje i ulazi u for petlju koja vrši pretragu unetog datuma u fajlu **statistika.txt** i štampa redove bez datuma, kada u koloni za datume pronađe podudaranje.

Na slici 4 je prikazan izgled konzole posle izvršenja funkcije za neku zadatu bazu podataka.

def **playersStats**():

utakmica = input("Unesite datum utakmice(DD.MM.YYYY.) >> ")

print(' ')

print("Statistika igraca sa utakmice odigrane",utakmica,"\n")

if not Statistika.gamedata(utakmica):

print("Za uneti datum ne postoji odigrana utakmica.")

else:

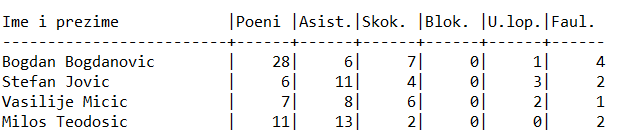
print(Statistika.formatHeader())

for i in Statistika.statistika:

if utakmica in i['datum']:

print(Statistika.formatPlayer(i))

Listing 8. Funkcija za štampanje statistike sa utakmice



Slika 4. Primer izgleda tabele za prikaz statistike sa neke odigrane utakmice

## Pregled odigranih utakmica

Kada referent izabere opciju ‘7’ tj. opciju za prikaz svih odigranih utakmica ekipe u sezoni, poziva se funkcija prikazana u listingu 9.

Funkcija radi pomoću for petlje u kojoj se vrši štampanje potrebnih podataka, štampa se tabela sa dve kolone u kojima se nalaze datum kada je utakmica odigrana i ime protivničke ekipe, dok se zaglavlje štampa pre ulaska u petlju pomoću poziva funkije iz fajla **Utakmice.py**.

def **printGames**():

print("Sve odigrane utakmice \n")

print(Utakmice.formatHeader())

for utakmica in Utakmice.utakmice:

print(Utakmice.formatPlayer(utakmica))

Listing 9. Funkcija za štampanje odigranih utakmica

## Prikaz grafikona “pobede-porazi”

Kada referent izabere opciju ‘8’ tj. opciju za prikaz broja pobeda i poraza ekipe tokom sezone pomoću grafikona, poziva se funkcija prikazana u listingu 10.

def **winlose**():

allgames = [i['ishod'] for i in Utakmice.utakmice]

pobede=0

porazi=0

for i in allgames:

if i == 'w':

pobede+=1

else:

porazi+=1

width = 0.1

fig, ax = plt.subplots()

rects1 = ax.bar(1 - width/2, pobede, width, label = 'Pobede')

rects2 = ax.bar(1 + width/2, porazi, width, label = 'Porazi')

ax.set\_ylabel('Broj pobeda i poraza')

ax.set\_title('Broj pobeda i poraza u sezoni')

ax.legend()

plt.show()

Listing 10. Funkcija za štampanje grafikona “pobede-porazi”

Na početku programa je u program je importovana Python biblioteka *matplotlib* i njen podmodul *pyplot*, pomoću koda prikazanom u listingu 11. *Matplolib* biblioteka služi za crtanje 2D grafika.

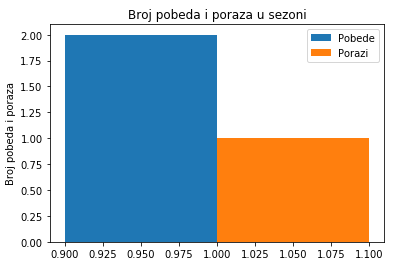
import matplotlib.pyplot as plt

Listing 11. Importovanje matplotlib biblioteke kao plt

Funkcija za prikaz grafika “pobede-porazi” radi tako što prvo napravi novu listu “allgames” u koju pomoću for petlje upiše sve vrednosti ‘w’ za pobedu i ‘l’ za poraz koje su prethodno već unete u program. Upisivanje se vrši tako što for petlja prolazi kroz sve redove i kolone u fajlu **utakmice.txt**.

Kada je lista formirana uvode se dve nove promenljive i njihova vrednost se postavlja na nulu pošto će služiti kao brojači pobeda ili poraza. Kada je to urađeno pokreće se for petlja koja prolazi kroz prethodno formiranu listu i prebrojava koliko kojih članova ima, ako naiđe na ‘w’ poveća brojač za pobede za jedan, u suprotnom poveća brojač za poraze za jedan.

Grafikon se forimira pomoću već unapred ugrađenih funkcija u Python, samo im se prosleđuje vrednost pobeda i poraza, jedan grafikon za neku bazu podataka prikazan je na slici 5.



Slika 5. Primer grafikona “pobede-porazi” za neku proizvoljnu bazu podataka

## Prikaz grafikona “igrač-indeks korisnosti”

Kada referent izabere opciju ‘9’ tj. opciju za korisnosti igrača tokom sezone pomoću grafikona, poziva se funkcija prikazana u listingu 12.

Kao i kod prikaza grafikona “pobede-porazi” potrebno je importovati *matplotblib* bioblioteku. To je već urađeno na početku programa, na koji način prikazano je u [listingu 11](pobede-porazi#_Prikaz_grafikona_).

def **playerIndexRating**():

x\_podaci = []

y\_podaci = []

print("Prosecan indeks korisnosti \n" )

allplayers = set([i['igrac'] for i in Statistika.statistika])

for igrac in allplayers:

playerstats = Statistika.findPlayer(igrac)

poeni = [int(i['poeni']) for i in playerstats]

asist = [int(i['asistencije']) for i in playerstats]

skok = [int(i['skokovi']) for i in playerstats]

blok = [int(i['blokade']) for i in playerstats]

ul = [int(i['ukrlopte']) for i in playerstats]

faul = [int(i['faulovi']) for i in playerstats]

brojutakmica = [i['datum'] for i in playerstats]

index = (sum(poeni)+sum(asist)+sum(skok)+sum(blok)+sum(ul)-sum(faul))/len(brojutakmica)

print("{0} {1:.2f}".format(igrac,index))

x\_podaci.append(igrac)

y\_podaci.append(index)

plt.bar(x\_podaci, y\_podaci)

plt.xlabel('igraci')

plt.xticks(rotation=90)

plt.ylabel('index')

plt.ylim(ymin=0, ymax=50)

plt.show()

Listing 12. Funkcija za prikaz grafikona “igrač-indeks korisnosti”

Indeks korisnosti nekog igrača se računa tako što se sabere suma svih njegovih poena, asistencija, skokova, blokada i ukradenih lopti na odigranim utakmicama, od zbira svih suma oduzima se suma napravljenih faulova na tim utakmicama i na kraju dobijena razlika se deli sa brojem odigranim utakmica.

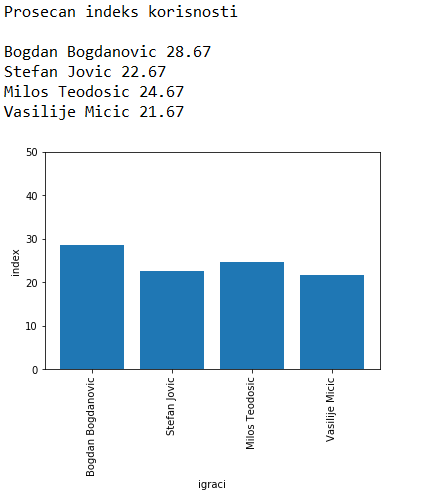
Funkcija se zasniva upravo na računanju prosečnog indeksa korisnosti igrača, pa se to potom predstavlja na grafiku pomoću funkcija koje već postoje u *matplotlib* biblioteci.

Pre početka računanja prosečnog indeksa u funkciji se prave dve nove liste “x\_podaci” u koju ce biti smeštena imena igrača koja će posle biti ispisana ispod stubića na grafikonu i “y\_podaci” u koju ce biti smeštene vrednosti prosečnog indeksa za te igrače i određivati visinnu stubića.

U narednom koraku se pravi još jedna lista “allplayers” u koju se smeštaju svi igrači bez ponavljanja sto se postiže pomoću komande ‘set’ i for petlje koja prolazi kroz sve redove u **statistika.txt**. Kada je ta lista formirana nova for petlja prolazi kroz nju i za svakog igrača pronalazi sve njegove statističke parametre i beleži u novu listu “playerstats” pomoću poziva funkcije za pronalaženje podataka o igraču iz **Statistika.py** tako što joj prosleđuje ime igrača za kojeg se računa indeks.

Kada je to urađeno formiraju se nove posebne liste za svaku kategoriju i u njih se beleže potrebni podaci za računanje indeksa koji se potom računa po već ranije navedenoj formuli. Kada je indeks izračunat dodaje se u listu “y\_podaci” a igrač u listu “x\_podaci”

Izgled grafikona napravljenog pomoću ove funkcije prikazan je na slici 6. Grafikon iznad sebe ima odštampane vrednosti indeksa za svakog igrača.



Slika 6. Primer grafikona “igrač-indeks korisnosti” za neku proizvoljnu bazu podataka

# Zaključak

Prikazan je jedan jednostavan sistem za vođenje evidencije o učinku igrača nekog košarkaškog tima tokom sezone, napisan u programskom jeziku *Python*. Program ima prostora za doradu u vidu dodavanja nekih novih funkcija i interfejsa u savisnosti od potrebe korisnika. Ovako napisan program bez ikakvih dorada može pronaći primenu u bilo kom klubu koji nema neke velike zahteve u vidu naprednih statističkih prikaza.

U ovom programu imali smo mogućnosti da prikazujemo evidenciju igrača u klubu, tj. osnovne podatke o njima u vidu imena, prezimena, visine, težine i datuma rođenja, kao i njihove učinke u najbitnijim kategorijama tokom utakmica poeni, asistencije, skokovi, blokade, ukradene lopte i faulovi. Imali smo opciju za unos novih igrača u klub kao i dodavanje novih utakmica i učinka igrača na njima, kao i opcije za prikaz dva grafikona za neke naprednije podatke kao što je bilo računanje prosečnog indeksa korisnosti. Program je izvršavao neko ko već od ranije ima pristup sistemu tako što se prijavljivao pomoću korisničkog imena i lozinke.

Dakle program ima prostora za napredak i dodavanje novih funkcija, ali je ovako napisan sasvim dovoljan za rad na nekom normalnom nivou informacija i vođenje evidencije stanja kluba ali ipak za rad sa većom količinom podataka potrebne su neke dorade.