

Simulacija Fudbala

Seminarski rad iz predmeta Programske Paradigme

Nemanja Mićović¹ Lazar Ranković²

¹nmicovic@outlook.com
www.alas.matf.bg.ac.rs/~mi13283
Matematički Fakultet

²lrankovic@outlook.com
www.alas.matf.bg.ac.rs/~mi13268
Matematički Fakultet

Prezentacija projekta, 2. februar 2016

O programu

Struktura programa

- ▶ Python engine za generisanje podataka
- ▶ Baza podataka
- ▶ Java program baziran na MVC-u

Motivacija za temu:

- ▶ Primena verovatnoće i statistike
- ▶ Duboko poznavanje samog fudbala
- ▶ Kompleksnost same tematike

Paradigme koje su javile

Korišćene paradigme:

- ▶ Skript paradigma (Python)
- ▶ Paradigma upitnih jezika (SQL)
- ▶ Objektna paradigma (Java)
- ▶ Komponentna paradigma (JavaFX)
- ▶ Konkurentna paradigma (Java niti)
- ▶ Funkcionalna paradigma (Java lambda izrazi)
- ▶ Vizuelna paradigma (UML dijagrami i konstrukcija automata)

Python engine za generisanje podataka

Koliko je podataka potrebno (na nivou sloga baze podataka) za simulaciju fudbalske utakmice?

1 tim → ime lige, ime tima, logo tima

1 igrač → ime, prezime, datum rođenja, tim, slika, 20 ocena

Uzmimo: 10 timova, 11 igrača Ukupno:

$10 * 11 * (20 + 6) + 10 * 4 = 2900$ slogova!

Python engine za generisanje podataka

Pitanje od milion dolara?

Šta radi programer kada treba da uradi više od 5 (ili 2900) puta istu stvar, iako mu pada teško i da samo jednom to radi?

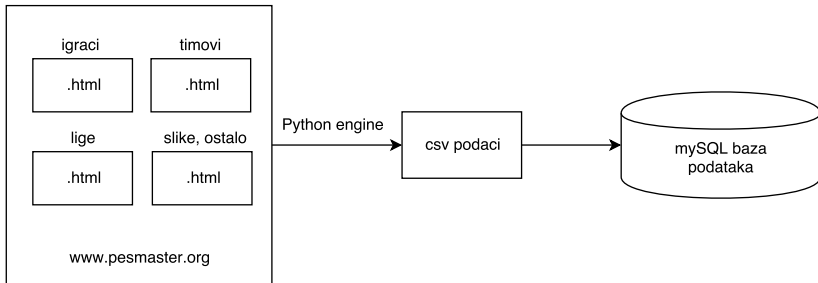
Python engine za generisanje podataka

Pitanje od milion dolara?

Napravi program koji to uradi za njega.

Python engine za generisanje podataka

Struktura



Python engine za generisanje podataka

Python kod

```

1 class Player:
2     def __init__(self, name, surname, stats, overall):
3         self.stats = stats
4         self.name = name
5         self.surname = surname
6         self.overall = overall
7     def get_tuple_stats(self):
8         return tuple(self.stats.keys())
9
10    def __str__(self):
11        return self.name.ljust(20) + self.surname.ljust(20) + str(self.overall).ljust(4)

```

classes/player.py python utf-8[unix] 27% 3: 1

```

22 # stats we have for our players
23 stats = [
24     "Attacking Prowess", "Dribbling", "Low Pass", "Lofted Pass", "Finishing", "Place Kicking", "Header",
25     "Defensive Prowess", "Ball Winning", "Kicking Power", "Speed", "Explosive Power", "Body Balance", "Jump"
26 ]
27
28 # Returns v1 for player_stat regex
29 # @return str
30 def generate_regex_for_stats1(stat_name):
31     regex = r"<tr>\s*"
32     regex += r"<td><span class='\"A-Za-z '\"*>([0-9]*)</span></td></td>\s*" # backreference to stat value
33     regex += r"<td>(\" + stat_name + r\")\s*</tr>" # backreference to stat name
34     return regex
35
36 # Returns v2 for player_stat regex
37 # @return str
38 def generate_regex_for_stats2(stat_name):
39     regex = r"<tr>\s*"
40     regex += r"<td><span class='\"A-Za-z '\"*>([0-9]*)</span></td>\s*" # backreference to stat value
41     regex += r"<td>(\" + stat_name + r\")\s*</td></tr>" # backreference to stat name
42     return regex
43
44 # @return str
45 def generate_regex_for_overall():
46     regex = r"\s*<h1 class='\"top-header('\"><span class='\"'\">ovr [a-zA-Z]*('\"><([0-9]*)</span>\s*"
47     return regex
48
49 # @returns Player
50 def get_player_object(data, stats, name, surname):
51     dragon_stats = {}
52     for stat in stats:
53         res = re.search(generate_regex_for_stats1(stat), data)

```

NORMAL > PASTE generate_players_csv.py python utf-8[unix] 42% 53: 1

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

Python engine za generisanje podataka

Baza podataka

Applications localhost / localhost: yto 4e6 2 13:22 localhost / localhost / seminarski / players | phpMyAdmin 4.0.10deb1 - Google Chrome

localhost/phpmyadmin/index.php?token=815c2d1b58d73dd82a5ecc0f48a1b5a2#PMAURL-2:sql.php?db=seminarski&table=players&server=1&target=...

Server: localhost - Database: seminarski - Table: players

	team_id	player_id	name	surname	position	date_of_birth	picture
Edit Copy Delete	1	1	Luis	Suarez	st	1987-01-24	[BLOB - 12.1 KiB]
Edit Copy Delete	1	10	Sergio	Busquets	dm	1988-07-16	[BLOB - 11.9 KiB]
Edit Copy Delete	1	11	Andres	Iniesta	cm	1984-05-11	[BLOB - 12.1 KiB]
Edit Copy Delete	2	12	Edgar	Barreto	cm	1984-07-15	[BLOB - 12.7 KiB]
Edit Copy Delete	2	13	Mattia	Cassani	rb	1983-08-26	[BLOB - 11.9 KiB]
Edit Copy Delete	2	14	Lazaros	Christodouloupoulos	rw	1986-12-19	[BLOB - 11.6 KiB]
Edit Copy Delete	2	15	Eder	Lopez	st	1986-11-15	[BLOB - 12.4 KiB]
Edit Copy Delete	2	16	Antonio	Cassano	st	1982-07-12	[BLOB - 12.9 KiB]
Edit Copy Delete	2	17	Vasco	Regini	lb	1990-09-09	[BLOB - 11.2 KiB]
Edit Copy Delete	2	18	Niklas	Moisander	cb	1985-09-29	[BLOB - 12 KiB]
Edit Copy Delete	2	19	Emiliano	Viviano	gk	1985-12-01	[BLOB - 11 KiB]
Edit Copy Delete	1	2	Gerard	Pique	cb	1987-02-02	[BLOB - 12.4 KiB]
Edit Copy Delete	2	20	Matias	Silvestre	cb	1984-09-25	[BLOB - 13 KiB]
Edit Copy Delete	2	21	Fernando	Martins	cm	1992-03-03	[BLOB - 11.7 KiB]
Edit Copy Delete	2	22	Roberto	Soriano	am	1991-03-08	[BLOB - 12 KiB]
Edit Copy Delete	2	23	Eden	Hazard	lw	1991-01-07	[BLOB - 11.6 KiB]
Edit Copy Delete	3	24	Oscar	Santos	am	1991-09-09	[BLOB - 12.5 KiB]
Edit Copy Delete	3	25	Diego	Costa	st	1988-10-07	[BLOB - 13.5 KiB]
Edit Copy Delete	3	26	Willian	Borges	rw	1988-08-09	[BLOB - 15.9 KiB]
Edit Copy Delete	3	27	Gary	Cahill	cb	1985-12-19	[BLOB - 11.2 KiB]
Edit Copy Delete	3	28	Cesc	Fabregas	dm	1987-05-04	[BLOB - 12.5 KiB]
Edit Copy Delete	3	29	John	Terry	cb	1980-12-07	[BLOB - 12.1 KiB]

Baza podataka

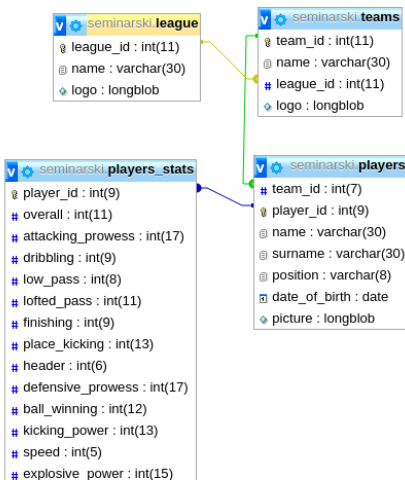
Baza podataka

Zašto baza podataka?

- ▶ performanse
- ▶ sigurnost
- ▶ prenosivost
- ▶ manipulacija podataka
- ▶ paralelna obrada

Baza podataka

Schema baze



Baza podataka

Slika baze

Applications localhost / localhost / seminarski / players | phpMyAdmin 4.0.10deb1 - Google Chrome

localhost/phpmyadmin/index.php?token=815c2d1b58d73dd82a5ecc0f48a1b5a2#PMAURL-2:sql.php?db=seminarski&table=players&server=1&target=...

Server: localhost - Database: seminarski - Table: players

	team_id	player_id	name	surname	position	date_of_birth	picture
Edit Copy Delete	1	1	Luis	Suarez	st	1987-01-24	[BLOB - 12.1 KiB]
Edit Copy Delete	1	10	Sergio	Busquets	dm	1988-07-16	[BLOB - 11.9 KiB]
Edit Copy Delete	1	11	Andres	Iniesta	cm	1984-05-11	[BLOB - 12.1 KiB]
Edit Copy Delete	2	12	Edgar	Barreto	cm	1984-07-15	[BLOB - 12.7 KiB]
Edit Copy Delete	2	13	Mattia	Cassani	rb	1983-08-26	[BLOB - 11.9 KiB]
Edit Copy Delete	2	14	Lazaros	Christodouloupoulos	rw	1986-12-19	[BLOB - 11.6 KiB]
Edit Copy Delete	2	15	Eder	Lopez	st	1986-11-15	[BLOB - 12.4 KiB]
Edit Copy Delete	2	16	Antonio	Cassano	st	1982-07-12	[BLOB - 12.9 KiB]
Edit Copy Delete	2	17	Vasco	Regini	lb	1990-09-09	[BLOB - 11.2 KiB]
Edit Copy Delete	2	18	Niklas	Moisander	gb	1985-09-29	[BLOB - 12 KiB]
Edit Copy Delete	2	19	Emiliano	Viviano	gk	1985-12-01	[BLOB - 11 KiB]
Edit Copy Delete	1	2	Gerard	Pique	cb	1987-02-02	[BLOB - 12.4 KiB]
Edit Copy Delete	2	20	Matias	Silvestre	cb	1984-09-25	[BLOB - 13 KiB]
Edit Copy Delete	2	21	Fernando	Martins	cm	1992-03-03	[BLOB - 11.7 KiB]
Edit Copy Delete	2	22	Roberto	Soriano	am	1991-03-08	[BLOB - 12 KiB]
Edit Copy Delete	2	23	Eden	Hazard	lw	1991-01-07	[BLOB - 11.6 KiB]
Edit Copy Delete	3	24	Oscar	Santos	am	1991-09-09	[BLOB - 12.5 KiB]
Edit Copy Delete	3	25	Diego	Costa	st	1988-10-07	[BLOB - 13.5 KiB]
Edit Copy Delete	3	26	Willian	Borges	nw	1988-08-09	[BLOB - 15.9 KiB]
Edit Copy Delete	3	27	Gary	Cahill	cb	1985-12-19	[BLOB - 11.2 KiB]
Edit Copy Delete	3	28	Cesc	Fabregas	dm	1987-05-04	[BLOB - 12.5 KiB]
Edit Copy Delete	3	29	John	Terry	cb	1980-12-07	[BLOB - 12.1 KiB]

Baza podataka

Korišćena tehnologija

Korišćene tehnologije:

- ▶ mySQL
- ▶ PhpMyAdmin

Java program

Zašto java?

Zašto java?

- ▶ portabilnost
- ▶ kvalitetan interfejs za rad sa bazom
- ▶ kvalitetna dokumentacija
- ▶ velika zajednica
- ▶ kvalitetno razvojno okruženje
- ▶ velika količina dostupnih biblioteka
- ▶ IntelliJ Idea

Java program

MVC patern

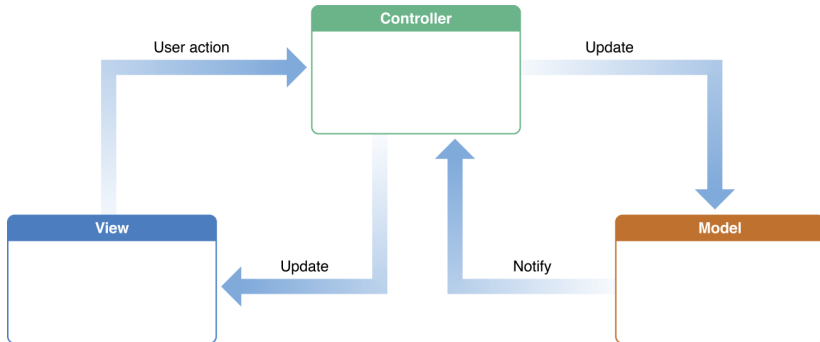
MVC (Model, View, Controller) patern (engl. pattern) razdvaja problem na 3 osnovne komponente.

Osnovni delovi:

- ▶ model (engl. model)
- ▶ pogled (engl. view)
- ▶ kontroler (engl. controller)

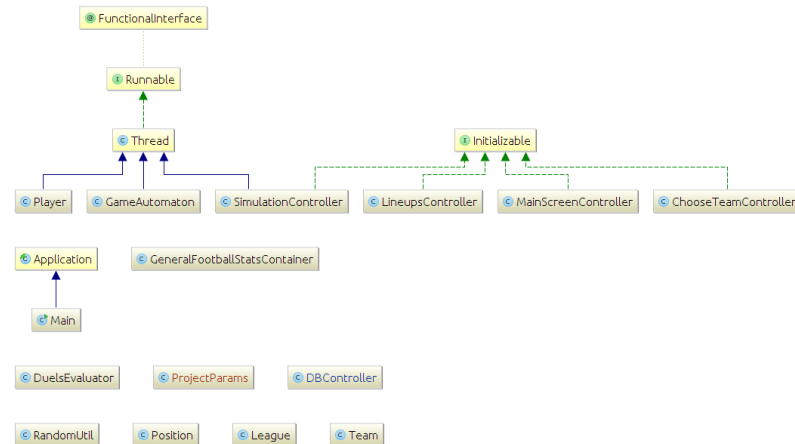
Java program

MVC patern



Java program

UML dijagram klasa



Powered by yllies

Java program

Paradigme u javi

Koje se to paradigme javljaju u Javi?

- ▶ Objektna paradigma
- ▶ Komponentna paradigma
- ▶ Konkurentna paradigma
- ▶ Funkcionalna paradigma
- ▶ Paradigma upitnih jezika

Java program

Objektna paradigma

Korišćeni elementi objektne paradigme:

- ▶ enkapsulacija objekata
- ▶ nasleđivanje
- ▶ polimorfizam
- ▶ biblioteke (yodatetime, jdbc)

Java program

Objektna paradigma

```

private void simulate(int n)
{
    if (gameMinutes == 90) {
        automatonState = state.finalState;
        return;
    } else if (gameMinutes == 45) {
        automatonState = state.halfTimeState;
        System.out.println("Half time!");
        return;
    }

    for (int i = 0; i < n; i++) {
        automatonIterations++;
        switch (automatonState) {
            // INITIAL
            case initialState:
                // At the start of game, teamA has the ball
                automatonState = state.possessionA;
                break;

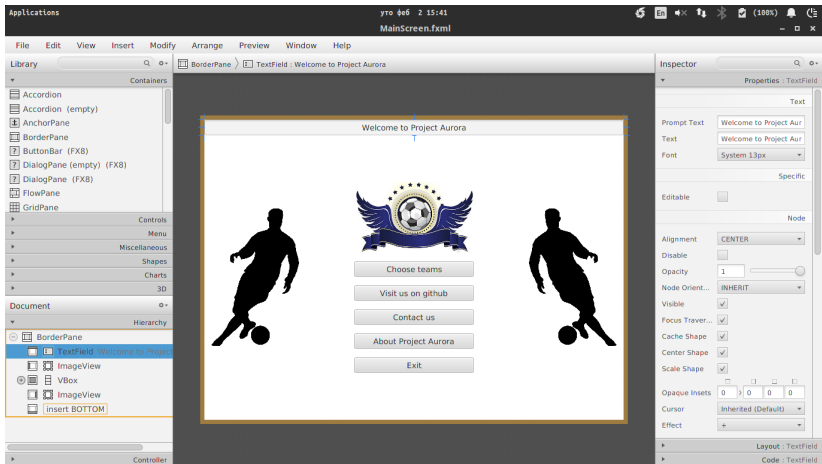
            // HALF TIME
            case halfTimeState:
                // At half time, teamB has the ball
                automatonState = state.possessionB;
                break;

            // POSSESSION (most important, controls most of the automaton)
            case possessionA:
                automatonState = handlePossessionState(automatonState); // switches global variable automatonState
                break;
            case possessionB:
                automatonState = handlePossessionState(automatonState);
                break;

            // GOALS
            case goalA:
                handleGoalAction(1);
                break;
            case goalB:
                handleGoalAction(2);
                break;
        }
    }
}
    
```

Java program

Komponentna paradigma



Java program

Konkurentna paradigma

Program koristi java klasu Thread koju nasleđuju neke od implementiranih klasa.

Java niti omogućavaju kontrolu brzine simulacije, dodelu niti samom automatu koji se izvršava odvojeno od niti za iscrtavanje interfejsa i slično.

Java program

Funkcionalna paradigma

Primena lambda izraza sa sortiranje.

```
// Gets somebody most likely to dribble from a team
public Player getDribbler()
{
    Vector<Player> homies = new Vector<>(players);
    Collections.sort(homies, (o1, o2) -> o2.getStats().get("dribbling") - o1.getStats().get("dribbling"));

    RandomUtil rand = new RandomUtil();
    float val = rand.runif();
    if (val <= 0.7f) {
        val = rand.runif();
        if (val <= 0.5f)
            return homies.get(0);
        else
            return homies.get(1);
    } else {
        val = rand.runif();
        if (val <= 0.5f) {
            // 3 4
            return homies.get(rand.getFromInterval(2, 3));
        } else {
            return homies.get(rand.getFromInterval(4, 10));
        }
    }
}
```

Java program

Funkcionalna paradigma

Primena lambda izraza sa sortiranje.

```
public void sortPlayersByPosition()
{
    HashMap<String, Integer> valuators = new HashMap<>();
    valutors.put("gk", 0);
    valutors.put("lb", 1);
    valutors.put("cb", 2);
    valutors.put("rb", 3);
    valutors.put("dm", 4);
    valutors.put("cm", 5);
    valutors.put("am", 6);
    valutors.put("lw", 7);
    valutors.put("rw", 8);
    valutors.put("st", 9);
    Collections.sort(players, (o1, o2) ->
        valutors.get(o1.getPosition()) - valutors.get(o2.getPosition())
    );
}
```

Java program

Paradigma upitnih jezika

Korišćenje jezika SQL za komunikaciju sa bazom podataka.

```
private void setChosenTeams()
{
    DBController query = new DBController();
    String query1 = "select * from players  p join players_stats ps on p.player_id = ps.player_id ";
    query1 += "join teams t on t.team_id = p.team_id ";
    query1 += "join league l on l.league_id = t.league_id where t.name =' " + lb_tean1.getText() + "'";

    String query2 = "select * from players  p join players_stats ps on p.player_id = ps.player_id ";
    query2 += "join teams t on t.team_id = p.team_id ";
    query2 += "join league l on l.league_id = t.league_id where t.name =' " + lb_tean2.getText() + "'";

    ResultSet players_team_1 = query.sendQuery(query1);
    ResultSet players_team_2 = query.sendQuery(query2);

    players_team_A = new Vector<Player>();
    players_team_B = new Vector<Player>();
}
```

Konačni deterministički automat

Formalna definicija

Definicija

Nedeterministički konačni automat je uređena petorka $(\Sigma, Q, I, F, \Delta)$.

Gde su:

- ▶ Σ azbuka
- ▶ Q skup stanja
- ▶ I početna stanja
- ▶ F završna stanja
- ▶ Δ skup prelaza po stanjima

Konačni deterministički automat

Formalna definicija

Definicija

Deterministički konačni automat zadovoljava prethodno datu definiciju i važi:

- ▶ *Ne postoje ε prelazi*
- ▶ *$|I| = 1$*
- ▶ *Prelaz iz svakog stanja u naredno je jedinstveno određen.*

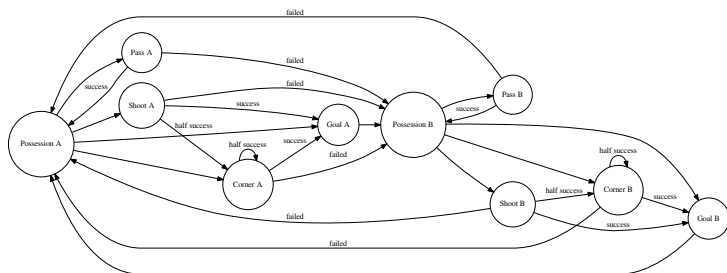
Algoritam za simulaciju

Automat se koristi za simulaciju utakmice.

Stanja automata su događaji iz fudbala poput:

- ▶ Tim A ima posed lopte
- ▶ Igrač A iz tima B šutira
- ▶ Korner za tim A
- ▶ Igrač A dodaje loptu igraču B
- ▶ Igrač A oduzima loptu igraču B
- ▶ Gol za tim A

Algoritam za simulaciju



Algoritam za simulaciju

Prelazak automata iz stanje u stanje

Automat menja stanja na osnovu ishoda događaja koji se simuliraju.

Kada tim A ima posed lopte, generišu se verovatnoće (zasnovane na podacima iz realnog života) na osnovu kojih se donosi odluka šta tim A pokušava da uradi sa loptom.

Ishod bilo kakve akcije tima A se zasniva na pozivanju evaluacione funkcije koja vraća ishod događaja koji se izračunava na osnovu ocena igrača koji učestuju u duelu.

Algoritam za simulaciju

Kako poboljšati nivo realnosti simulacije?

- ▶ Bolje modelovanje evaluacionih verovatnoća
- ▶ Dodavanja novih stanja u automat (aut, penal, izmena...)

Dalji razvoj

Budućnost samog projekta?

Aplikacija je:

- ▶ skalabilna
- ▶ dokumentovana
- ▶ na GitHub-u
- ▶ prati MVC
- ▶ lako proširiva

Dalji razvoj

Budućnost algoritma za simulaciju?

Šta je potrebno dodati:

- ▶ uopštavanje
- ▶ primena na druge sportove
- ▶ optimizacija
- ▶ formalna verifikacija

Pitanja

Pitanja?

Zahvalnica

za seminarski rad

Ovom prilikom bi smo se zahvalili dr. Mileni Vujošević Janičić na pruženoj prilici da kreiramo prikazani seminarski rad, kao i na znanju koje smo stekli tokom slušanja kursa *Programske paradigme*.

Zahvalili bismo i kolegi Čedomiru Dimiću koji nam je pomogao pri unosu podataka u bazu podataka.