UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Uroš Zarić

SISTEM ZA PRUŽANJE STATISTIKE ONLINE BORBENE IGRE

- seminarski rad iz predmeta NoSQL baze podataka -

Novi Sad, 2019.

SADRŽAJ

UVOD

Višekorisničke borbene igare za cilj imaju odigravanje mečeva između timova sačinjenih od igrača odnosno korisnika na mreži. Svaki meč, tim i individualni igrač zahtevaju praćenje i čuvanje parametara njihove igre kao što su dužina trajanja, broj pobeda i slično. Analizom navedenih podataka igračima se pružaju značajni podaci o njihovom uspehu sa ciljem njihovog usavršavanja u igri.

Sistem za pružanje navedenih statističkih informacija računarske igre sastoji se iz baze podataka i web aplikacije.

Baza podataka treba da omogući lak pristup radi pružanja brzih informacija i skladištenje velike količine podataka o svim odigranim mečevima na neograničeni period. Povezanost mnogobrojnih igrača, timova i mečeva upućuje na veliki značaj i važnost veza između njih. Grafovske baze podataka se nameću kao rešenje koje najviše odgovara navedenim potrebama jer akcenat upravo stavljaju na jednakom značaju veza kao i samih entiteta. Mogu se predstaviti grafovima odnosno skupom čvorova i veza između njih.

Više reči u Poglavlju 2 - Model

Neo4J je konkretna implemetacija takve baze podataka pisana u Java programskom jeziku. Služi se specijalizovanim Cypher upitnim jezikom koji olakšava pisanje upita nad ovakvom strukturom podataka.

Web aplikacija igračima omogućava prezentaciju željenih informacija, pristupajući i obrađujući podatke iz baze podataka. Korišćene su Java web tehnologije i Spring framework za efikasniju izradu ovakvih aplikacija.

Više reči u Poglavlju 3 - Implementacija

Igračima se omogućava jednostavan pristup putem web browser-a bez potrebe za instalacijom dodatne aplikacije. Neophodna je preglednost i jednostavnost korišćenja radi korisničke ugodnosti ovakve aplikacije.

Više reči u Poglavlju 4 - Izvršavanje

BAZA PODATAKA

2.1 Grafovske baze podataka

Strukturu grafovske baze podataka čine čvorovi i veze između njih. Oni imaju jednaku važnost što je bitna razlika u odnosu na relacione baze gde su veze predstavljene presečnim tabelema koje povezuju entitete (zasebne tabele) što otežava "kretanje" kroz njih. Grafovske baze omogućavaju vrlo lako i efikasno "kretanje" kako kroz čvorove tako jednako i kroz njihove veze što je značajno za baze u kojima se čuva znatno više veza od eniteta.

Poput relacionih baza i u grafovskim bazama mogu se naznačiti tipovi čvorova i veza što se postiže labelama (:NAZIV_LABELE). Čvorovi i veze mogu imati svoje atribute određenih tipova (naziv atributa: tip) koji ih opisuju i određuju.

2.2 Model baze podataka online borbene igre

Osnovna potreba baze podataka ovog sistema je skladištenje odigranih mečeva (*Slika 1*). Svaki meč igraju dva suprotstavljena tima sa po 5 igrača na 5 različitih pozicija (gornja, sredina, donja, džungla i podrška). Igrači pre početka meča biraju pozicije i heroje sa kojima će odigrati meč, ne postoje ograničenja pri odabiru.

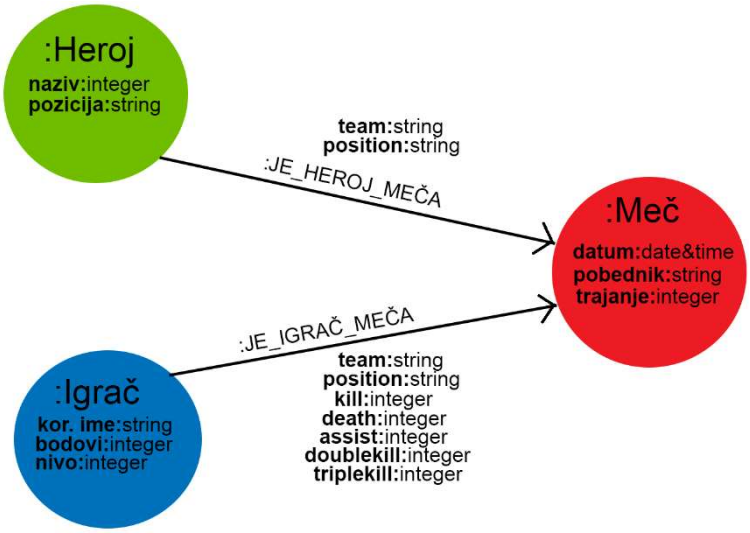
Čvorovi:

Igrač ima jedinstveno korisničko ime, pored kojeg se čuvaju sakupljeni bodovi i nivo.
Heroji su određeni nazivom i preporučenom pozicijom.
Mečevi su predstavljeni datumom i vremenom odigravanja, pobednikom i trajanjem.

Radi bržeg pretraživanja samim time i pristupa bazi, čvorovi su indeksirani na osnovu određenih atributa (Igrač-ime, Heroj-naziv, Meč-datum).

Veze:

Potrebno je zapamtiti igrača i heroja na svakoj poziciju u oba tima. Pozicije i timovi čuvaju se kao atributi veza između igrača/heroja i mečeva na kojima su igrali. Igrači sa odabranim herojima tokom meča postižu određene parametre kao što je broj asistencija, zbog izbegavanja redundantnosti podataka oni se čuvaju samo kao atributi veza između igrača i mečeva. Čuvanje različitih tipova veza za svaku različitu poziciju i tim bila bi beskorisna odluka, razlog je činjenica da atributima lako pristupamo i proveravamo njihove vrednosti umesto da informacije čuvamo u labelama veza. Čuvanje dostignuća na meču za svakog igrača zasebnim čvorovima je takođe suvišno.



Slika 1 *- Metamodel Grafovske Baze*

2.3 Pristup i administracija grafovske baze podataka

Kreiranje nove baze zahteva unos novih podataka ili prenos postojećih podataka iz druge baze. Podaci se najčešće čuvaju u fajlovima u tabelarnom formatu (podeljeni u kolone i redove). Prevođenje zasebnih tabela entiteta u čvorove ne iziskuje napore ali prevođenje posrednih tabela veza u veze između čvorova zahtevaju opreznost jer je za grafovske baze karakterističan veliki broj veza gde i najmanja greška može dovesti do napornog ponavljanja procesa ispočetka.

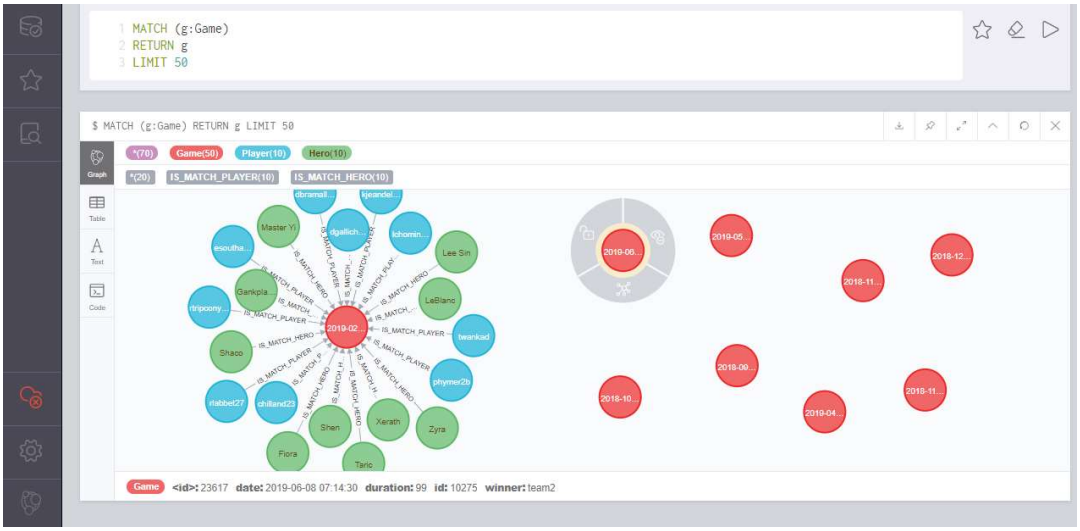
Aplikacija pristupa podacima koji se automatski unose u bazu nakon svakog odigranog meča. Preuzeti podaci se obrađuju da bi se pružile složene informacije o igračima i herojima kao što su najuspešniji heroji određenog igrača, KDA (odnos ubistava,smrti i asistencija) heroja i slično.

Korišćena je najpopularnija implementacija grafovske baze Neo4J.

Pogodnosti:

1. namenjena bilo kom problemu
2. prilagodljiva reprezentacija grafova
3. podrška bezbednosti podataka kroz ACID transakcije
4. brz pristup i kratak upitni jezik

Nudi pregledan administrativni grafički interfejs za podešavanje baze, izvršavanje upita i prikaz rezultata u formi grafa ili dokumenta (*Slika 2*).



***Slika 2** - Izgled grafičkog interfejsa Neo4J baze*

Cypher upitni jezik predstavljen je adekvatnom sintaksom za izvršavanje naredbi nad grafovskim bazama, značajno olakšavajući i ubrzavajući proces razvoja aplikacije (*Kod 1*).

```
MATCH (i:Igrac)-[v:JE_IGRAČ_MEČA]->(m:Meč)
WHERE i.ime=`Mark´and m.pobednik=team1 and v.kill>10
RETURN m
```

***Kod 1** - Ilustracija upitnog jezika Cypher*

WEB APLIKACIJA

3.1 Korišćene tehnologije

Spring je Java framework koji eliminiše konfigurisanje uobičajenih delova novog projekta čime programiranje aplikacije čini efikasnijim omogućavajući programeru da se fokusira na problem. Anotacijama se označavaju delovi koda radi pružanja dodatnih informacija kompajleru jezika sa ciljem skraćivanja koda i efikasnijeg programiranja.
Web aplikacija se izvršava na aplikativnom serveru koji može biti pokrenut na bilo kom računaru poželjno povezanim sa mrežom.

Koristi Model-View-Controller (MVC) organizaciju aplikacije:
1. model - predstavljanje strukture (čvorova i veza) baze podataka
2. repozitorijum - pristup, izvršavanje upita i dobavljanje rezultata iz baze
3. kontrolor - rukovanje korisničkim zahtevima i obrada dobavljenih podataka
4. web app - prezentacija obrađenih podataka kao odgovor korisniku

Neo4J je implementacija grafovske baze podataka, sadrži sve potrebne alate za rad nad bazom. Baza se nalazi na serveru baze podataka, koji softverski može biti pokrenut na bilo kom računaru poželjno povezanim sa mrežom.

Rezultati upita nad bazom se u aplikaciji mogu predstaviti kao:
1. individualni (tačno 1 red i kolona) - proste strukture: Integer, String, Float...
2. tabelarni (više redova ili kolona) - lista (redovi) mapa (kolone za svaki red-zaglavlje i vrednost): List<Map<String, Object>>
3. čvorovi i veze - ručno definisanim klasama iz modela: Player, PlayerGame

3.2 Obrada korisničkih zahteva i podataka iz baze

Karakterističan i čest zahtev aplikaciji ovog sistema je pronalaženje najboljih saigrača određenog igrača, najboljih heroja i slično (*Kod 2*).
Metoda kontrolora prihvata zahteve i poziva odgovarajuće metode repozitorijuma.
Metode repozitorijuma trebaju da vrate saigrače prosleđenog igrača, broj odigranih mečeva i broj pobeda sa tim saigračem. Ovi zahtevi bazi ne mogu se zadovoljiti jednim upitom po dva različita kriterijuma pa su neophodna dva upita. Jedan upit dobavlja saigrače i broj mečeva odigranih sa njim, a drugi dobavlja saigrače i broj pobeđenih mečeva sa njim.

Neophodno je spojiti rezultate istog saigrača prolaskom iteratora kroz dve liste. Prva lista sadrži imena svih saigrača i broj odigranih. mečeva. Igrač ne mora imati pobedu sa određenim saigračem pa druga lista može biti kraća. Potencijalne razlike u broju redova dve liste mogu dovesti do nepreklapanja imena saigrača pa je neophodno voditi računa postavljanjem dodatnih uslova. Pri spajanju dve liste za saigrače je potrebno izračunati procenat pobeda.

Izračunate procente pobeda za svakog saigrača dalje je potrebno sortirati opadajuće radi određivanja nejuspešnijih saigrača pri čemu je neophodno napraviti posebne komparatore.

Moguća je struktura baze u kojoj se za svakog igrača zasebnim tipovima vezama čuvaju pobeđeni i izgubljeni mečevi. Ona bi olakšala postavljanje upita sa prethodnim kriterijumima

ali bi otežala unošenje podataka u bazu i prolazak kroz sve mečeve što je glavni razlog odustajanja od navedenog alternativnog rešenja.

```
@Controller
@RequestMapping(value="/players")
public class PlayerController {
	@Autowired
	PlayerRepository pr;

@RequestMapping(value="/playerStats", method=RequestMethod.GET)
public String playerStats(HttpServletRequest request, String username) {

	List<Map<String, Object>> saigracBrMec = pr.saigraciMec(username);
	List<Map<String, Object>> saigracBrPob = pr.saigraciPob(username);
	Iterator<Map<String, Object>> it1 = saigracBrMec.iterator();
	Iterator<Map<String, Object>> it2 = saigracBrPob.iterator();
	List<ArrayList<String>> list = new ArrayList<ArrayList<String>>();
	boolean prev = false;
	Map<String, Object> map2 = new HashMap<String, Object>();
	while (it1.hasNext() && it2.hasNext()) {
		Map<String, Object> map1 = it1.next();
		if(!prev)
		    map2 = it2.next();
		ArrayList<String> tempList = new ArrayList<String>();
		tempList.add(map1.get("username").toString());
   if(map1.get("username").toString().equals(map2.get("username").toString())) {
		prev = false;
		tempList.add(String.valueOf(Math.round((Float.parseFloat(map2.get("
		brPob").toString())                                                    /
		(Float.parseFloat(map1.get("brMec").toString()) / 100)))));
	} else {
		prev = true;
		tempList.add(String.valueOf(0));
	}
		tempList.add(map1.get("brMec").toString());
		list.add(tempList);
	}
	list.sort(new Comparator<ArrayList<String>>() {
		@Override
		public int compare(ArrayList<String> o1, ArrayList<String> o2) {
		  if(Integer.parseInt(o1.get(1)) == Integer.parseInt(o2.get(1)))
		                        return 0;
		  if(Integer.parseInt(o1.get(1)) < Integer.parseInt(o2.get(1)))
		                        return 1;
		  else
		                        return -1;
		}
	});
	request.getSession().setAttribute("listaSaigraca", list);

	return "Igraci";
}

}
```

***Kod 2** - Metod kontrolora: Određivanje najboljih saigrača*

3.3 Pristup bazi podataka iz programskog jezika

Pristup i izvršavanje upita nad Neo4J bazom iz Java aplikacije omogućava poseban Neo4J drajver koji se navodi pri konfigurisanju projekta i automatski preuzima sa mreže. Poželjno je da upiti nad bazom obrade sve potrebne proračune nad bazom i vrate što jednostavnije rezultate kako bi se smanjila nepotrebna potrošnja memorije aplikativnog servera. Upiti nad bazom se navode u anotacijama iznad zaglavlja metoda repozitorijuma.

Složeni upiti (*Kod 3*) omogućavaju pronalaženje saigrača i heroja prosleđenog igrača. Prvo treba povezati prosleđenog igrača sa svim mečevima na kojima je igrao, potom povezati ostale igrače ili heroje sa tim mečevima. Heroji moraju igrati na istoj poziciji i timu, a saigrači samo u istom timu. vraćamo nazive saigrača ili heroja i broj mečeva na kojima su zajedno učestvovali.

```
public interface PlayerRepository extends Neo4jRepository<Player, String> {

  @Query("MATCH (p:Player)-[r:IS_MATCH_PLAYER]->(g:Game),
  (h:Hero)-[t:IS_MATCH_HERO]->(g)\r\n" +
  "WHERE p.username = {username} and t.position = r.position and
  r.team = t.team\r\n" +
  "RETURN h.name as name, h.image as image, count(g) as brMec\r\n" +
  "ORDER BY h.name")
  List<Map<String,Object>> najHerojiMec(@Param("username") String username);

  @Query("MATCH (p:Player)-[r:IS_MATCH_PLAYER]->(g:Game),
  (h:Player)-[t:IS_MATCH_PLAYER]->(g)\r\n" +
  "WHERE p.username = {username} and r.team = t.team\r\n" +
  "RETURN h.username as username, count(g) as brMec\r\n" +
  "ORDER BY h.username")
  List<Map<String,Object>> saigraciMec(@Param("username") String username);
}
```

***Kod 3** - Metode repozitorijuma: Određivanje heroja i saigrača prosleđenog igrača*

3.4 Odgovor ka korisniku

JSP(Java Server Pages) je Java tehnologija za kreiranje dinamičkih stranica na strani servera.

Metode kontrolora kao odgovor na korisnički zahtev pozivaju JSP stranice i prosleđuju kao parametre zahteva prethodno dobavljene i obrađene rezultate iz baze.
Rezultati najboljih igrača ili heroja se tabelarno prikazuju u box-u sa naslovom pri čemu je potrebno proći for-petljom kroz rezultate i prikazati tri kolone (naziv i slika, procenat pobeda i broj odigranih mečeva).

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>HEROJI</title>
<link rel="shortcut icon" type="image/png" href="/LoL/images/favicon.ico"/>
<link rel="stylesheet" type="text/css" href="/LoL/styles/styles.css">
</head>
<body>
   <div id="container">...</div>
   <div id="header">...</div>
   <div id="body">
      <div class="box">
         <p class="podnaslov">JAKPROTIV</p><hr>
         <c:forEach begin="0" end="4" items="${jakProtiv }" var="items">
            <div id="row">
                <div id="left">
                   <img src="${items[1]}"width="50"height="50">
                 <h3>${items[0] }</h3>
                </div>
                <div id="middle"><h2>${items[2]}%</h2><p>POBEDA</p></div>
                <div id="right"><h2>${items[3]}</h2><p>MEČEVA</p></div>
            </div>
         </c:forEach>
         <c:remove var="jakProtiv"/>
       </div>
    </div>
</body>
</html>
```

***Kod 4** - JSP stranica: Prikaz rezultata*

KORIŠĆENJE

Aplikacije putem mreže omogućavaju korisnicima lak pristup i korišćenje bez potrebe za posebnom aplikacijom i skladištem. Korisnik pristupa aplikaciji jednostavno putem svog web browser-a u vidu web stranica.

4.1 Početna stranica

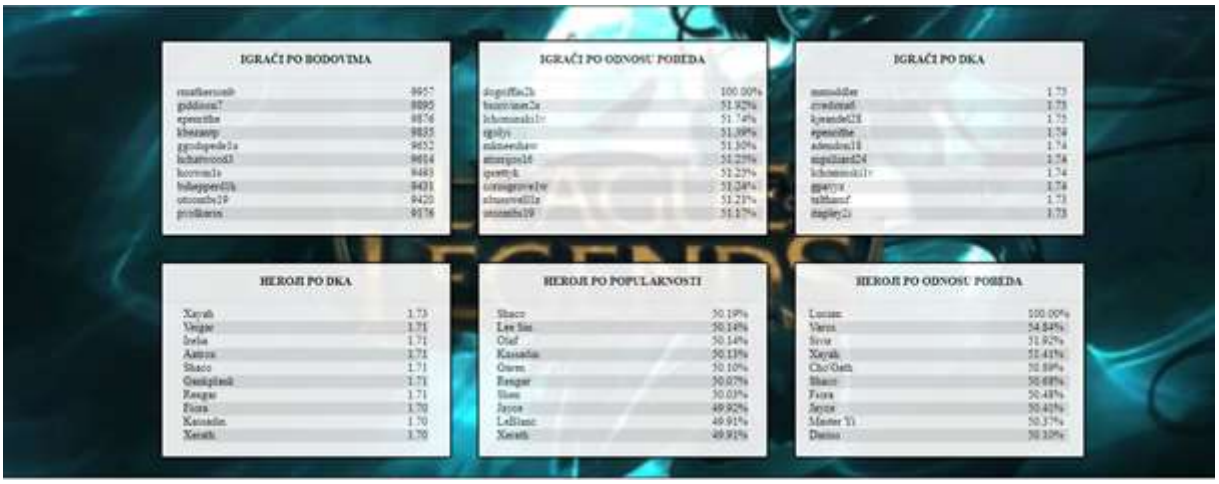
Početna stranica (*Slika 3*) pruža mogućnosti unosa, izmena i brisanja igrača i mečeva u bazu.
Desna Leva forma dodaje igrače i mečeve u bazu.
Srednja forma nudi dugme "Osveži" prikazuje sve igrače i mečeve dodate u bazi. Klikom na određenog igrača (označen plavo) označavaju se mečevi (crveno) na kojima je igrao, klikom na neoznačene mečeve povezuju se označeni igrač sa datim mečevima.
Desna forma omogućava ažuriranje i brisanje označenog igrača.

***Slika 3** - Početna stranica sa CRUD operacijama*

4.2 Stranica opštih podataka svih igrača i heroja

Stranica „Tabele" (*Slika 4*) prikazuje nam opšte statističke podatke igre bazirane na svim igračima, herojima i mečevima.

Kriterijumi tabela:
1 po broju osvojenih bodova (igrači)
2 po procentu odigranih mečeva u odnosu na ukupan broj mečeva (heroji)
3 po procentu pobeda u odnosu na ukupan broj mečeva (igrači i heroji)
4 po odnosu DKA ((death+assist) / kill) (igrači i heroji)



***Slika 4** - Stranica sa statističkim tabelama*

4.3 Stranica podataka o izabranom igraču

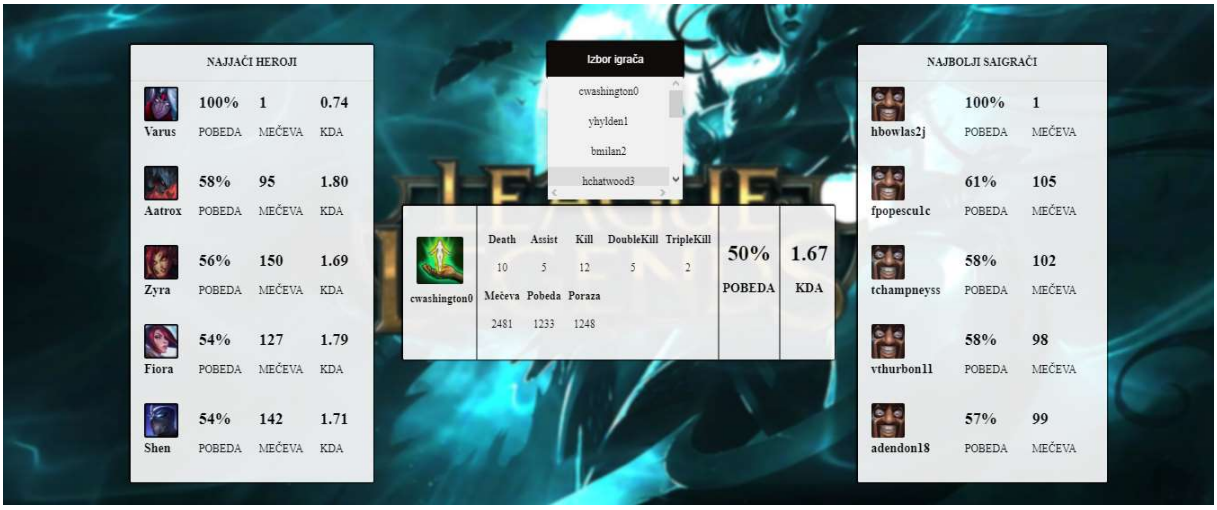
Stranica "Igrači" (*Slika 5*) nudi dugme sa padajućim menijem za izbor igrača. Za odabranog igrača prikazuju se osnovni podaci, odnos sa herojima i drugim igračima.

Osnovni podaci igrača:

1. naziv, broj bodova i nivo
2. broj ubistava, asistencija, smrti, dvostrukih i trostrukih ubistava po meču
3. ukupan broj mečeva, pobeđenih i izgubljenih mečeva
4. procenat pobeda
5. DKA ((death+assist) / kill) odnos na osnovu svih odigranih mečeva

Kriterijumi tabela (s leva na desno):

1. heroji sa najviše pobeda na osnovu svih odigranih mečeva sa datim igračem i ukupan DKA odnos na osnovu tih mečeva
2. saigrači sa najviše pobeda na osnovu svih odigranih mečeva



Slika 5 *- Stranica sa informacijama o igraču*

4.4 Stranica podataka o izabranom heroju

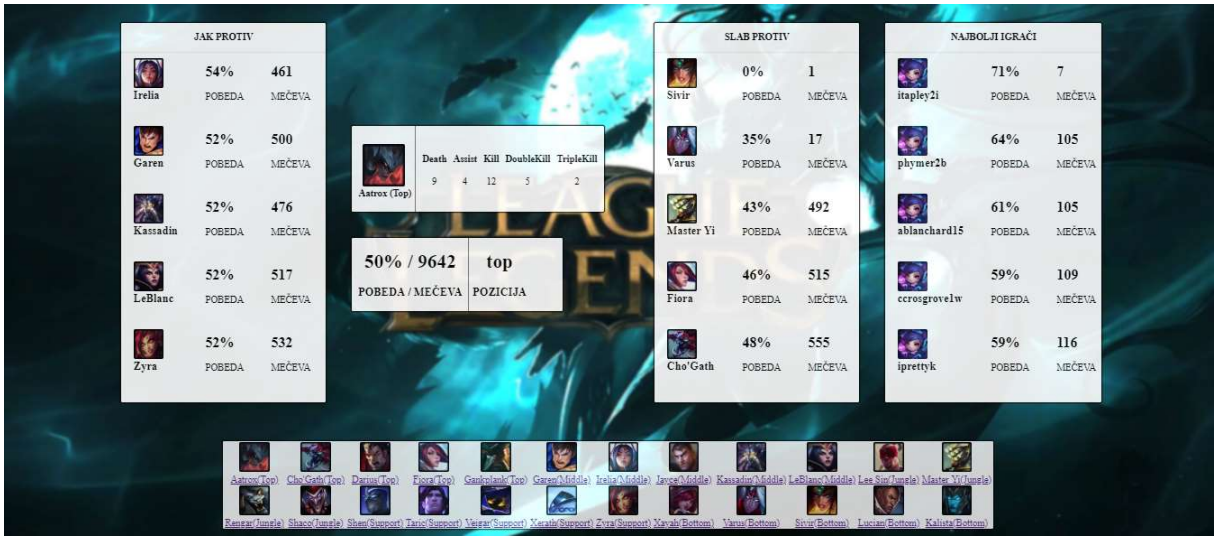
Stranica "Heroji" (*Slika 6*) tabelarno prikazuje sve heroje, klikom na određenog heroja prikazuju se njegovi osnovni podaci kao i odnos sa drugim herojima i igračima.

Osnovni podaci heroja:

1. naziv, preporučena pozicija
2. broj ubistava, asistencija, smrti, dvostrukih i trostrukih ubistava po meču
3. procenat pobeđenih u odnosu na ukupan broj mečeva
4. najčešća pozicija na osnovu svih odigranih mečeva

Kriterijumi tabela (s leva na desno):

1. heroji na istoj poziciji u protivničkom timu protiv kojih je ostvareno najviše pobeda
2. heroji na istoj poziciji u protivničkom timu protiv kojih je ostvareno najmanje pobeda
3. igrači sa najviše pobeda na osnovu svih odigranih mečeva sa datim herojem



Slika 6 *- Stranica sa informacijama o heroju*

ZAKLJUČAK

Izrada ovakvog sistema zahteva dublje poznavanje domena problema i tehnologija implementacije. Najmenje greške u planiranju, naročito početnih faza kao što je model baze podataka, mogu značajno usporiti i otežati izradu projekta jer su ispravke temeljne i obimne. Inicijalizacija baze, prenos podataka i integracija sa drugom bazom nad velikom količinom podataka predstavlja pravi izazov. Složena struktura podataka kao i nemogućnost individualne provere svakog unetog podatka zahteva izuzetnu snalažljivost i pripremljenost administratora.

NoSQL baze pružaju značajne pogodnosti u odnosu na tradicionalne relacione baze, prilagođene su primeni i rešavanju problema iz specifičnih domena. Grafovske baze, kao jedan predstavnik, posmatraju veze jednako kao entitete što značajno koristi kod baza sa velikom međusobnom povezanošću entiteta. Njihova logika i kraći upitni jezik olakšavaju izdvajanje poveznika po određenim kriterijumima koje zadaje programer ili administrator. Efikasnost u izvršavanju upita je od izuzetne važnost i interesa za savremenog korisnika koji svakodnevno u pretražuje veliku količinu informacija. Iako performanse narušavaju samu bezbednost baze podataka, savremene NoSQL baze pružaju poboljšanja u ovom odnosu.

Savremeni razvojni okviri programskih jezika značajno doprinose efikasnijoj izradi aplikacija. Nude gotove šablone i eliminišu standardne korake kreiranja i izvršavanja programa. Okviri omogućavajući programeru da se usredsredi na zadati problem čime postaje produktivniji i pažljiviji. Web aplikacije omogućavaju pristup putem web browser-a što korisnicima značajno štedi vreme i resurse.

LITERATURA

[1] “Cypher Manual,” 2019. [Online]. Available: https://neo4j.com/docs/cypher-manual/current/.

[2] “Developer Guides,” 2019. [Online]. Available: https://neo4j.com/developer/get-started/.

[3] “Documentation,” 2019. [Online]. Available: https://spring.io/docs/reference.

[4] “Guides,” 2019. [Online]. Available: https://spring.io/guides.

[5] “Operations Manual,” 2019. [Online]. Available: https://neo4j.com/docs/operations-manual/3.5/.