SofaScore Edukacija
Završni projekt
League Manager

Alen Murtić, alen.murtic@sofascore.com
Karlo Knežević, knezevic.karlo1@gmail.com

3. lipnja 2021. - 23. lipnja 2021. u 12:00

1 Uvod

Završni projekt SofaScore Edukacije prikazat će razumijevanje gradiva svih predavanja te upoznaje polaznike sa SofaScore domenom. Projektni je zadatak simulacija zadatka kakvog bi programeri dobili u početnom stadiju razvoja jedne aplikacije. Za razliku od prvog projekta, ne samo da je dozvoljeno, već je preporučljivo koristiti javno dostupne biblioteke.

1.1 Model podataka - 30 bodova

> Slug je transformacija stringa tako da može uključiti u simbolički URL bez enkodiranja. Nije nužno jednoznačan, npr. Đoković se može pretvoriti u Djokovic ili Dokovic, to su implementacijski detalji programske knjižnice koja se koristi (iako kao poznavatelji hrvatskog jezika znamo da je djokovic preciznije). U sklopu ovog projekta pretpostavit ćemo da slug sadrži samo mala slova i znak minus. Npr. očekivani slug stringa "Marin Čilić" će biti "marin-cilic". Da bi se taj string mogao uključiti u URL bez transformacije u slug, bilo bi ga potrebno enkodirati PHP funkcijom urlencode čiji bi rezultat bio sasvim nešto drugo. U ovom projektu slugifikacija ne mora biti jako pametna, važno je da je implementirana. Možete koristiti neku vanjsku knjižnicu koja to radi, sami napisati nešto ili iskoristi rješenje s interneta. U slučaju da je slugifikacija dio projekta (znači druga ili treća opcija iz prošle rečenice), potrebno je napisati testove za nju.

Potrebno je podržati dva sporta - nogomet i košarku.

1.1.1 Sport

Sport je osnovni razred modela, a sadrži sljedeće podatke:

- identifikator (int)
- ime (string) - npr. Football, Basketball, itd.
- slug (string) - npr. football, basketball, itd.

Slug je **jedinstven samo nad sportom** i samo za sport se koristi umjesto identifikatora u rutama, tj. ruta koja prima sport parametar neće biti izvedena kao /sport/id (npr. sport/1), nego /sport/slug (npr. /sport/football ili samo /football).

1.1.2 Category (kategorija)

Category je entitet koji modelira prostornu ili organizacijsku značajku pojedinog natjecanja. Njezina funkcionalnost unutar projekta je samo semantičko grupiranje natjecanja. Primjeri kategorija su: Hrvatska, Engleska, Europa, Formula 1, WTA, ATP, ITF, Davis Cup. Važno je iz modela ispod uočiti da je kategorija vezana uz sport pa tako postoji kategorija Hrvatska za nogomet, Hrvatska za košarku, Hrvatska za rukomet i slično. Model Category treba izgledati ovako:

- identifikator (int)
- ime (string) - npr. Croatia, England, WTA, ATP
- slug (string)
- sport (Sport) - sport kojoj kategorija pripada

1.1.3 Competitor (natjecatelj)

Competitor je entitet koji se natječe protiv drugih Competitora. U nogometu, košarci rukometu i sličnim sportovima to je klub ili reprezentacija, tj. momčad. U sportovima poput tenisa, pikada ili stolnog tenisa, competitor je osoba.
Nadalje, competitor može biti više od jedne osobe koji ipak nisu klub, npr. parovi u tenisu, stolnom tenisu ili badmintonu.
Da bi se osigurala jednostavnost implementacije Match modela, **svi tipovi natjecatelja se spremaju u jednu tablicu, competitor**. Ta tablica mora sadržavati atribute dovoljne za opis svih navedenih tipova competitora. Oni su:

- identifikator (int)
- ime (string) - ime kluba ili osobe, za parove treba biti konkatenacija imena osoba odvojenih crticom
- slug (string)
- sport (Sport) - sport natjecatelja

- država (Country) - država kluba ili reprezentacije, državljantsvo osobe ili parova (ako oba imaju isto državljanstvo). Iskoristiti embedded entitete. Country nad natjecateljem nije nullabilan, ali kod države unutar Country objekta je.
- type (int ili string) - određuje je li competitor, tim/momčad, osoba ili par (npr. parovi u tenisu). U slučaju da je competitor modeliran apstraktnim razredom Competitor i trima konkretnim razredima (Team, Person i Pair) u obliku single table nasljeđivanja, tada je type string. Ako postoji samo konkretan razred Competitor, tada je type int.

1.1.4 Competition (natjecanje)

Competitori se jedni protiv drugih natječu u sklopu nekog natjecanja - Competitiona. Svako natjecanje pripada jednoj kategoriji, npr. UEFA Champions League pripada kategoriji Europe, 1. Hrvatska nogometna liga pripada kategoriji Croatia. Natjecanje na sebi nema sport jer ga ima kategorija. Dakle, model natjecanja je:

- identifikator (int)
- ime (string)
- slug (string) - transformacija imena tako da se ono može uključiti u simbolički url
- kategorija (Category)
- koliko puta natjecatelji igraju jedni protiv drugih u sezoni (int)

U sklopu ovog projekta možete potpuno ignorirati natjecanja od više faza, više grupa, ona koja imaju sustav na ispadanje, doigravanja i isključivo razmatrati ligaška natjecanja poput engleske Premier lige ili HNL-a (bez doigravanja za ispadanje). Zato je ključno imati informaciju koliko puta timovi igraju međusobno u sezoni (npr. za Premier ligu je to 2, a za HNL 4).

Napomena: *Alternativni naziv za Competition je Tournament. Iako mislimo da je Competition s jezičnog aspekta ispravniji naziv, u projektu možete koristiti Tournament iz banalnog razloga da kad krenete pisati "Comp" u PHPStormu, on ponudi samo Competitor razred, a ne i Competition te si time ubrzate pisanje koda.*
Ako se pitate što mi u SofaScoreu koristimo - kod nas su svi Competitori u klasi Team (da, Roger Federer je momčad), faze Competitiona (npr. regularna sezona u NBA, grupa u Ligi Prvaka) su nam Tournament, a NBA i Liga Prvaka UniqueTournament. Zašto tako čudno? Takvo imenovanje je davno preuzeto od vanjskih suradnika i sad je potpuno neisplativo refaktorirati.

1.1.5 Season (sezona)

- identifikator (int)
- ime (string) - npr. "2018-19" ili "2020"
- opcionalno: datum početka (datetime) - datum i vrijeme početka prve utakmice u sezoni
- opcionalno: datum završetka (datetime) - datum i vrijeme početka zadnje utakmice u sezoni

Prilikom implementacije slobodni ste odabrati postoji li u sustavu samo jedna "2019-20" sezona ili svako natjecanje koje traje od datuma u 2019. do datuma u 2020. ima svoju "2019-20" sezonu. Ako odaberete prvu opciju, treba napraviti entitet **CompetitionSeason** koji mora ima datum početka i datum završetka sezone, a za drugu opciju opcionalni atributi sezone postaju obvezujući. Dakle, oni moraju negdje biti zapisani, samo je pitanje gdje. U drugom slučaju, sezona mora imati i referencu na Competition.

1.1.6 Match (utakmica)

Utakmica, dvoboj natjecatelja unutar natjecanja u nekoj sezoni. Postoji jedan apstraktan razred Match koju nasljeđuju izvedenice za nogomet i košarku. Dosadašnji model je bio nezavisan o sportu, ali Match ne može biti jer se nogomet igra u dva poluvremena, a košarka u četvrtinama. Slobodno možete zanemariti košarku koja se igra u poluvremenima (NCAA). Model Matcha je:

- identifikator (int)
- domaći natjecatelj (Competitor)
- gostujući natjecatelj (Competitor)
- početak (datetime)
- status (int, konstante u kodu) - opcije: nije počela, 1. poluvrijeme, 2. poluvrijeme, pauza, 1.-4. četvrtina, otkazana, završena, produžeci (košarka)
- natjecanje (Competition)
- sezona (Season)
- domaći rezultati po periodima (Score)
- gostujući rezultati po periodima (Score)
- kod pobjednika (int) - null ako utakmica nije završena, 1 ako je pobijedio domaći natjecatelj, 2 gostujući, 3 neodlučeno

Budući da ovaj projekt pokriva samo ligaški sustav natjecanja, nogomet nema produžetaka.
Možete primijetiti da se neka natjecanja igraju na neutralnom terenu te nemaju gostujućeg i domaćeg natjecatelja, ali to je iz perspektive nazivlja sretnija opcija nego natjecatelj A i B. Ako smislite bolje nazivlje, slobodno ga koristite.

1.1.7 Score (raspisani rezultat)

Score je embedded entitet koji sadrži rezultate po periodima i konačni rezultat kao što je opisano u 10. predavanju.
Atribut Score razreda koji se nije dogodio (npr. svi prije početka utakmice ili rezultat produžetaka ako je utakmica završila u regularnom vremenu) **mora biti null**.

1.1.8 Standings i StandingsRow, tablica i redak u tablici

Standings je tablica, poredak timova za natjecanje u sezoni. Osim natjecanja i sezone, sadrži retke, entitete StandingsRow za svaki od natjecatelja koji igra u sezoni. Sadrži njegov broj pobjeda, poraza i neodlučnih rezultata te broj bodova i postotak pobjeda. Standings može biti ukupni (total), domaći i gostujući. Dakle, svako natjecanje u sezoni ima tri različita standingsa.
VAŽNA NAPOMENA: Čvrsto preporučamo da Standingsi budu implementirani jednim razredom Standings koji ima string "type" za razlikovanje jesu domaći, gostujući ili ukupni. I "single table", i "class table" nasljeđivanje bi učinilo korištenje standingsa značajno kompliciranijim.

Budući da projekt obrađuje samo ligaški sustav, retke standingsa koristite i kao popis natjecatelja u sezoni natjecanja.
Ako se odlučite za model u kojem sezona ima Competition, startDate i endDate, onda standings ne mora imati natjecanje jer to određuje sezona.

Dakle, mogući model standingsa je:

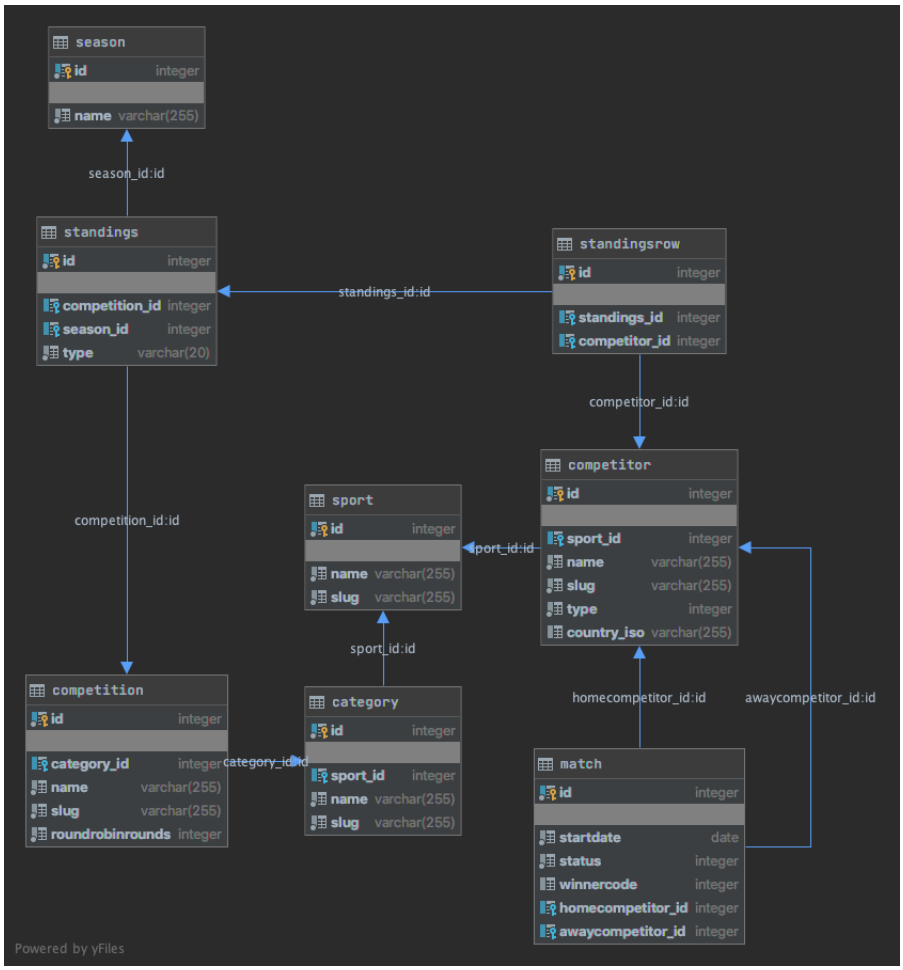
- identifikator (int)
- natjecanje (Competition)
- sezona (Season)
- type (string) - vrijednosti: home, away, total

A retka u standingsima:

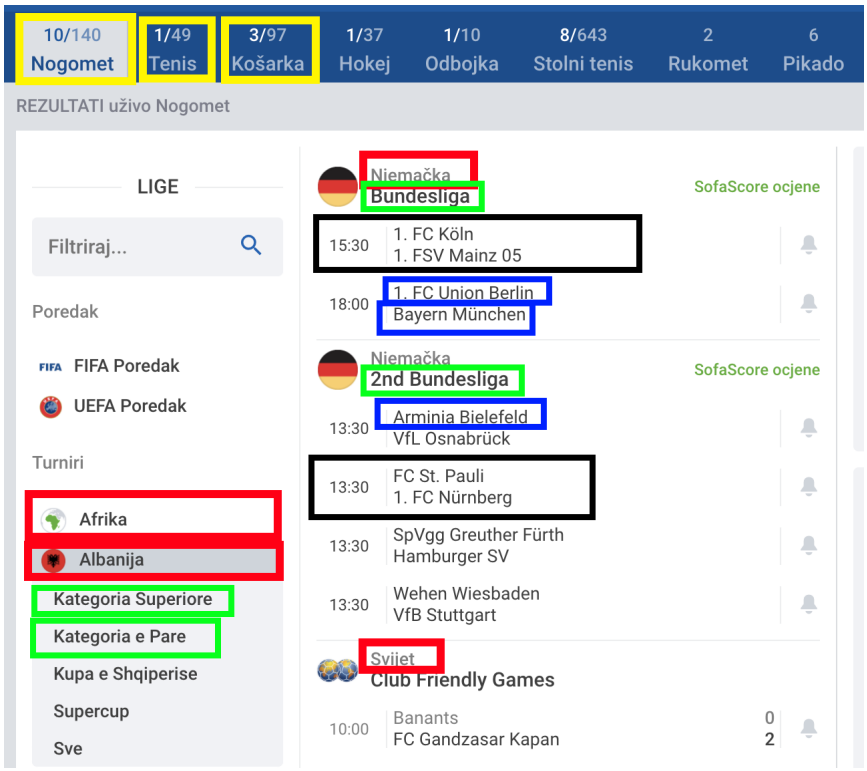
- identifikator (int)
- natjecatelj (Competitor)
- standings (Standings)
- matches (int) - broj završenih utakmica - incijalna vrijednost 0

- wins, losses, scoresFor, scoresAgainst (int, inicijalna vrijednost im je 0)
- draws, points (nullabilni int) - ne postoje u košarci
- percentage (nullabilni float) - udio pobjeda u ukupnom broju utakmica, ne postoji u nogometu

Na slici 1 nalazi se predloženi E-R model podataka s izostavljenim stupcima nad Match i StandingsRow entitetima da bi bio lakše čitljiv.
Slike 2 i 3 prikazuju označene entitete na SofaScore stranici. Entiteti su označeni bojama: žuta - Sport, crvena - Category, crna - Match, tamno plava - Competitor, zelena - Competition, svijetlo plava - Standings, ružičasta - StandingsRow, narančasta - Season.



Slika 1: Predloženi E-R model podataka



Slika 2: Stranica sporta na SofaScore web stranici

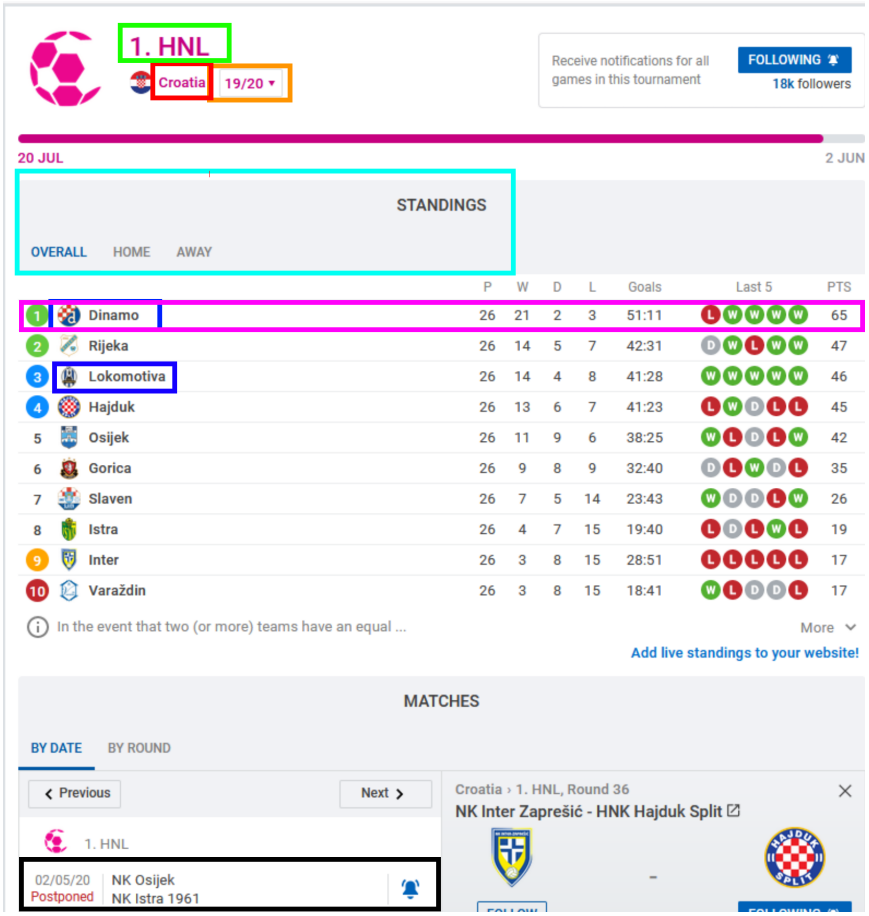
1.2 Aplikacijska logika - 50 + 10 bodova

Aplikacijska logika treba biti organizirana u servise. Kao što je istaknuto na predavanjima, i repozitoriji, i komande, i controlleri su servisi. No, ako se ista logika koristi i u komandi i u controlleru, potrebno je izdvojiti logiku u neki treći servis, ne napraviti tako da controller prima komandu kroz konstruktor ili obrnuto.

1.2.1 Komanda za popunjavanje sintetičkih (dummy) podataka - 10 bodova

Budući da ovaj projekt nema izvor stvarnih podataka, potrebno je napraviti komandu koja će generirati i spremati podatke. Ona je ujedno i dobar test ispravnosti modela. U bazu je potrebno ručno unijeti nogomet i košarku kao sportove. Svi nasumični stringovi koji će biti generirani trebaju biti slova engleske abecede ili razmak. Možete generirati samo mala slova i razmak pa nad tim stringom pozvati ucwords(trim(string)).

Algoritam komande je sljedeći:



Slika 3: Stranica natjecanja na SofaScore web stranici

1. Nasumičan odabir jednog od sportova u bazi.
2. Generiranje nasumičnog stringa između 5 i 7 znakova koji će biti ime kategorije. Ako kategorija s tim imenom za taj sport postoji u bazi, koristi nju. Inače, kreiraj novu.
3. Generiranje nasumičnog stringa između 5 i 7 znakova koji će biti ime natjecanja. Ako natjecanje s tim imenom postoji u bazi za tu kategoriju, dohvati njega. Inače, kreiraj novo.
4. Proizvoljni algoritam stvaranja dummy sezone i njenin početnih i završnih datuma. Sezona mora trajati između 7 i 11 mjeseci.
5. Stvaranje između 10 i 16 novih timova proizvoljnog imenovanja.

6. Stvaranje standingsa tipova total, home i away i standings redaka za svaki tim u svakom od tih triju standingsa.

7. Stvoriti raspored utakmica korištenjem servisa u odjeljku ispod.

1.2.2 Stvaranje rasporeda - 10 + 10 bodova

Stvaranje Match entiteta na temelju popisa natjecatelja u natjecanju u sezoni (tj. Standingsa). Prva utakmica mora početi kada počinje sezona, zadnja kad sezona završava. Ostatak raspoređivanja po vremenu je proizvoljan, uz zahtjev da svaki natjecatelj mora imati bar 12 sati između početaka dvije svoje utakmice.

Dobro/kreativno vremensko raspoređivanje utakmica može se nagraditi s dodatnih 10 bodova.

1.2.3 Stvaranje nove sezone dosadašnjem natjecanju - 5 bodova

Kao dva argumenta prima početni i završni datum nove sezone te naziv iste (ili ga može generirati na temelju datuma). Iz baze dohvati najbližu prošlu sezonu, uzme iste timove, kreira standingse za novu i pozove servis za stvaranje rasporeda.

1.2.4 Računanje Standingsa na zahtjev - 15 bodova

Servis koji računa standingse na temelju svih dosadašnjih utakmica u sezoni. To se računanje izvodi za sve natjecatelje koji su igrali u sezoni. Očekuje se da postoje reci za svakog natjecatelja u toj sezoni i natjecanju, inače treba baciti iznimku ili pustiti da logika pukne sama (npr. poziv metode na null vrijednosti) da bi netko mogao ručno pogledati zašto se to dogodilo. Paziti da se različito računanju tablice za košarku i nogomet.
ScoresFor i scoresAgainst je broj zabijenih i primljenih golova, odnosno koševa. Budući da je ovo ligaški sustav, u nogometu nemamo produžetke i penale pa je to samo suma konačnih rezultata utakmica.

1.2.5 Promjena standingsa na temelju utakmice - 10 bodova

Kada utakmica postane završena, trebaju se automatski izračunati standingsi za nju zbrajanjem rezultata na pripadne retke u standingsima (ne računanjem cijelih standingsa ispočetka). To ne treba implementirati uz pomoću listenera, već direktno na mjestima gdje se postavlja status da je završena (sljedeći odjeljak i na API-ju).

Dodatno, treba napisati listener za slučaj da se na otprije završenoj utakmici promijeni score, tada treba pokrenuti logiku računanja tablica na zahtjev, tj. logiku prošlog odjeljka. Listener ne smije dodatne promjene direktno pozivati u svojem kodu nego treba na neki način dodati novi posao u red poslova (npr. Redisom i komandom koja gleda stanje u Redisu, kao na predavanju).

1.2.6 Punjenje jedne utakmice nasumičnim rezultatom - 5 bodova

Za neko natjecanje i sezonu, servis treba uzeti prvu vremenski sljedeću utakmicu, postaviti joj završeni status i popuniti adekvatne atribute Score entiteta (one koji pripadaju tom sportu) nekim realnim rezultatima po sportu (nogomet neće završiti 70-70, niti košarka 4-2).

1.3 JSON API - 40 bodova

Za svaku rutu treba odrediti koji je njen tip - GET, POST ili PUT - te je implementirati. Koristiti ParamConverter za svaki dohvat entiteta čiji se identifikator prima u URL-u.

Sve rute koje primaju json podatke (json body) ili rade izmjene u sustavu su POST ili PUT. Json podacima (bodyju requesta) se može pristupiti pozivom $request->getContent() gdje je $request instanca Symfony\Component\HttpFoundation\Request razreda koju prima svaka akcija u controlleru ako se zada kao argument metode. Razlika između POST i PUT ruta je da su PUT idempotentne, tj. ako se više puta izvrše s istim podacima, stanje u bazi će biti isto.

Sve rute koje rade dohvat podataka su GET.

U sustavu treba napraviti sljedeće rute:

1. Ruta za promjenu podataka utakmice - 5 bodova - Za dani identifikator prima json koji sadrži status, datum početka i trenutne rezultate po timovima te podatke sprema u bazu.
2. Ruta za rekalkulaciju standingsa - 5 bodova - Za natjecanje i sezonu treba pozvati servis za rekalkulaciju standingsa ili napraviti job u pripadnom redu poslova.
3. Rute za izmjenu podataka o natjecatelju, natjecanju i sezoni - ukupno 10 bodova - svaka u URL-u ima samo identifikator entiteta (i nalazi se naravno u adekvatnom controlleru).
4. Ruta za dohvat zadnjih 5 završenih utakmica svakog od natjecatelja - 10 bodova - rezultat može biti mapa u kojoj su ključevi identifikatori natjecatelja, a vrijednosti liste utakmica za tog natjecatelja.
5. Ruta za dohvat standingsa u natjecanju i sezoni - 2.5 boda - bez serijaliziranih StandingsRow entiteta
6. Ruta za dohvat StandingsRow entiteta za zadani standings id - 2.5 boda - pojedini redak ima serijalizirane natjecatelje.

7. Ruta koja za natjecanje i sezonu u zahtjevu prima listu identifikatora natjecatelja i kreira novu sezonu s tim natjecateljima - 5 bodova - ona dijeli kod sa zadnje dvije točke algoritma komande za izradu sintetičkih podataka.

1.3.1 Autentikacija i autorizacija u sustavu - 20 bodova

Tko želi, može za 20 bodova dodati autentikaciju i autorizaciju u sustav (prvenstveno na API rute) korištenjem Symfony Security Bundlea kratko objašnjenog na zadnjem predavanju. Primitivno korisničko sučelje iz sljedeće točke ne smije imati ograničen pristup.

1.4 Korisničko sučelje - 5 bodova

1.4.1 Korisničko sučelje za čitanje podataka - 5 + 10 bodova

Treba napraviti korisničko sučelje koje se sastoji od 4 HTML stranice: popis kategorija za sport, popis natjecanja u kategoriji, popis sezona u natjecanju te stranica sezone u natjecanju s ukupnim standingsima i popisom utakmica s rezultatima. Dozvoljeno je da bude najprimitivnije moguće (znači red teksta ispod reda teksta) jer bit zadatka nije na njemu. Preporuka da bude napisano u twigu.

Dodatni trud uložen u izradu lijepog korisničkog sučelja bit će nagrađen s maksimalno 10 bodova.

1.4.2 Sučelje za unos podataka - 0 + 20 bodova

Ako netko želi i zna napraviti interaktivnu web stranicu koja koristi API iz prethodnog zadatka, slobodan je to napraviti za 20 bodova. Ostali mogu preskočiti ovaj zadatak. Za rad na tome bit će potrebno implementirati još neke rute, ponajprije GET za navigaciju.

1.5 Testovi - dodatnih 50% bodova po funkcionalnosti

Svaka funkcionalnost može donijeti dodatnih 50% bodova ako su za nju napisani korisni testovi. Npr. ako funkcionalnost nosi 10 bodova, s testovima je njen maksimum 15.

1.6 Ukupno bodovanje

Polaznici SofaScore akademije uspješnim će rješavanjem ovog projekta pokazati da razumiju teme prikazane na predavanjima, znaju koristiti PHP i Symfony dovoljno da mogu napraviti jednu umjereno složenu aplikaciju te mogu riješiti osnovne probleme iz SofaScore domene.

Za prolazak ovog projekta očito je potrebno napraviti model jer sve ovisi o njemu te ukupno skupiti 50 bodova. Ne postoje drugi pragovi. Maksimalni broj bodova koji polaznik može skupiti na projektu je 100, iako je ukupna suma

veća od 100 zbog opsežnosti projektnog zadatka. Svatko može odabrati kako želi doći do tih 100 bodova. Naravno, preporuka je riješiti što više zadataka i skupiti što više bodova.

2 Literatura

Za uspješno rješavanje ove domaće zadaće potrebno je razumijevanje predavanja vezanih uz Symfony i Doctrine. Dodatno, Symfonyjeva službena dokumentacija je izvrstan resurs jednostavnih programskih primjera i objašnjenja. Ona se nalazi na poveznici: `https://symfony.com/doc/current/index.html`.