SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Sakač

Razvoj web aplikacija korištenjem mikro okvira

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Sakač

Matični broj: 45998

Studij: Informacijski sustavi

Razvoj web aplikacija korištenjem mikro okvira

ZAVRŠNI RAD

Mentor:

dr. sc. Matija Novak

Varaždin, rujan 2021.

Martin Sakač

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada je izrada web aplikacije korištenjem mikro okvira u skriptnom jeziku PHP. U radu će se prikazati kratka povijest, značajke i prednosti PHP-a. Pojasnit će se pojam okviri te prikazati prednosti i nedostaci okvira te pojam MVC arhitekture. Nakon okvira slijedi razrada mikro okvira. Objasnit će se na koji način se instaliraju okviri te koji su alati za instalaciju potrebni. Odabrat će se nekoliko mikro okvira, prikazati njihove funkcionalnosti, dokumentacija, prednosti i nedostaci te će se argumentirati odabir dva mikro okvira koja će biti korištena za izradu web aplikacije.

Ključne riječi: mikro okviri, PHP, programiranje, web aplikacija, baza podataka, web server

Sadržaj

1. Uvod 1
2. Skriptni jezik PHP 2
2.1. Okviri u PHP-u 5
2.2. Mikro okviri 7
2.3. Razlika između okvira i mikro okvira 8
3. Primjena mikro okvira 9
3.1. Instalacija mikro okvira 9
3.2. Fat free 10
3.2.1. Usmjeravanje 11
3.2.2. Globalne varijable i konfiguracijske datoteke 12
3.2.3. Predlošci 13
3.2.4. Rad s bazama podataka 13
3.3. Lumen 15
3.3.1. Struktura Lumena 15
3.3.2. Usmjeravanje, kontroleri i pogledi 16
3.3.2. Baze podataka i ostale značajke 17
3.4. Flight 18
3.5. Limonade 19
3.6. PHPixie 20
3.7. Odabir mikro okvira 20
4. Izrada web aplikacije 22
4.1. Opis web aplikacije i korištene tehnologije 22
4.2. ERA model 22
4.3. Korisničke uloge u aplikaciji 24
4.3.1. Neregistrirani korisnik 24
4.3.2. Registrirani korisnik 28
4.3.3. Moderator 31
4.3.4. Administrator 33
4.4. Korištenje mikro okvira 34
4.4.1. Fat Free – Inicijalizacija 34
4.4.2. Fat Free – Kontroleri 35
4.4.3. Fat Free – Pogledi i predlošci 37
4.4.4. Fat Free – Ponavljajući segmenti predložaka 38
4.4.5. Fat Free – Ostale funkcionalnosti 40
4.4.6. Lumen 40

4.4.7. Rad aplikacije sa dva mikro okvira 42
5. Prednosti, nedostaci i problemi u radu sa mikro okvirima 43
5.1. Prednosti korištenja mikro okvira 43
5.2. Nedostaci korištenja mikro okvira 44
5.3. Problemi u radu sa više mikro okvira 44
6. Zaključak 46
Popis literature 47
Popis slika 49
Popis tablica 49

1. Uvod

Izrada složenih web aplikacija, danas je svakodnevica zaposlenih programera. Od implementacije kompleksnih klasa i funkcija, loše strukture i organizacije koda do potrošnje dragocjenog vremena na ne uspjele pokušaje optimizacije koda. Tu u igru ulaze okviri.

Okviri su platforme koje se koriste kao temelj za razvoj aplikacije. Okviri daju temelj programerima kako bi fokus bio, ne na izmišljanju već postojećeg i optimiziranog koda, već na izradi samog projekta. Okviri programeru pružaju gotovu arhitekturu, bogatu alatima, značajkama i bibliotekama koje pomažu u izradi efikasnog i optimiziranog koda s visokim performansama. Postoji veoma mnogo okvira te se svakim danom taj broj povećava. Zbog toga, programeri često moraju podosta vremena posvetiti sakupljanju informacija o okvirima kako bi odabrali najbolji okvir za njihove potrebe i posvetili vrijeme savladavanju odabranog okvira.

Okviri sa sobom nose mnoge prednosti, ali u slučajevima izrade manjih i jednostavnijih web aplikacije, mogu prouzročiti ne potrebne komplikacije i oštetiti performanse projekta. Tada se preporuča korištenje mikro okvira. Mikro okviri su minimalistički web okviri koji sadrže manje komponenata za izradu web aplikacije sa što lakšim i bržim učinkom. Koriste se za razvijanje web aplikacija kada nisu potrebne sve funkcionalnosti koje sa sobom nose pravi okviri već tek dio. Mikro okviri ubrzavaju razvijanje gotovim funkcijama i klasama te drže do jednostavnosti i čitljivosti koda kako bi programer što lakše izradio i održavao web aplikaciju. Funkcionalnost koju većinom svi mikro okviri olakšavaju je usmjeravanje. Prilikom HTTP GET ili POST zahtjeva, okviri pomoću svojih usmjerivača usmjeravaju HTTP zahtjev prema definiranom kontroleru. Taj kontroler obavlja funkcionalnost koju korisnik definira te vraća HTTP odgovor najčešće u obliku korisničkog sučelja. Mikro okviri često su dizajnirani za izradu API-ja za neku uslugu ili aplikaciju te izradu mikro servisa.

2. Skriptni jezik PHP

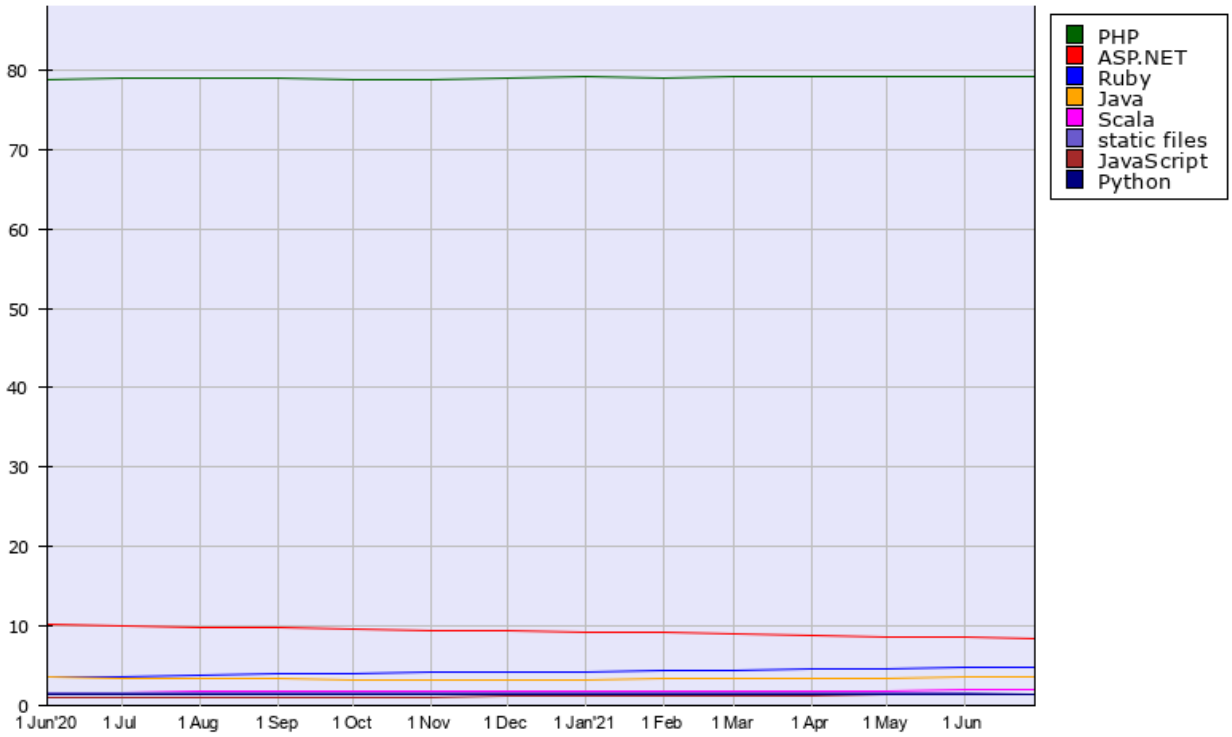
PHP (kratica eng. Hypertext Pre-processor) je skriptni jezik otvorenog koda koji se koristi za izradu HTML sadržaja i za programiranje na strani poslužitelja.

Prva inačica nastala je 1994. kada je Rasmus Lerdorf prikazao PHP kao skup cgi (eng. Common Gateway Interface) izvršnih datoteka koje su bile napisane u programskom jeziku C. CGI je specifikacija sučelja koje omogućava web poslužitelju da izvršava vanjske programe koji su najčešće služili za obradu korisničkih zahtjeva. U to vrijeme, nije bilo planirano da PHP postane skriptni jezik. [1]

U travnju 1996., najavljen je PHP 2.0. U toj najavi, prvi put se spominje da je PHP skriptni jezik. Kao noviteti, uveden je parser koji je puno bolje prevodio i prepoznavao dijelove izvornog programa te je implementirana uvjetova petlja if-else koja je uputila R. Lerdorfa prema stvaranju potpunog skriptnog jezika. [1]

PHP, do tada vođen radom samo jednog čovjeka, privukao je veću pozornost korisnika. Problem je bio u mehanizmu za parsiranje koji nije nudio traženu stabilnost. Tada se Zeev Suraski i Andi Gutmans nude da ponovno napišu temeljni mehanizam za parsiranje koji je temelj treće inačice PHP-a. Ostala zajednica ponudila je svoj rad na drugim dijelovima PHP-a i time je PHP postao projekt otvorenog koda. Godine 1998. nakon izlaska treće inačice, korisnici diljem svijeta uveliko su počeli koristiti jezik i doprinositi razvoju sljedećih inačica.[1]

Mnogi programeri, prema PHP-u kao jeziku za programiranje na strani poslužitelji, imaju odnos ljubavi i mržnje (eng. Love-hate relationship). Svaki programer barem je jednom naišao na članak, sliku ili poruku u kojoj je prikazana netrpeljivost prema jeziku. Usprkos tome, na slici 1, vidljivo je da se PHP kao programski jezik na strani poslužitelja upotrebljava na skoro 80% web stranica. Nakon PHP-a, sljedeći najpopularniji jezik na poslužiteljskoj strani, u ovom slučaju okvir (eng. framework) je Microsoft-ov ASP.NET sa tek 10% pokrivenosti. Možemo zaključiti da PHP definitivno dominira webom. [2]



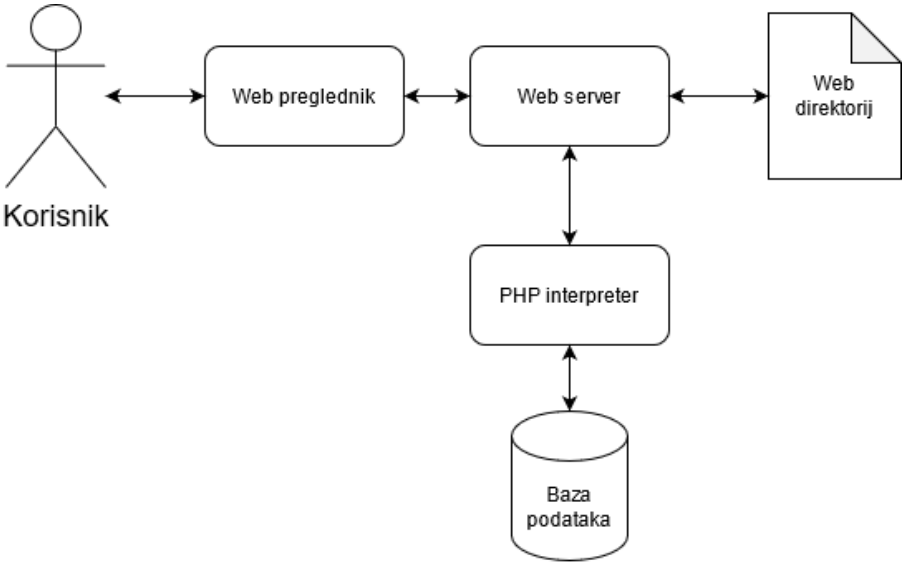
Slika 1: Upotreba programskih jezika na strani poslužitelja (Izvor: [2])

Zašto PHP drži toliko udio weba? Razloga ima mnogo, od širokog spektra odabira operacijskih sustava, web poslužitelja i baza podataka koje PHP podržava do mogućnosti pisanja objektno orijentiranog koda i korištenja raznih dodataka kao što su biblioteke, okviri i mikro okviri. Nadalje će se pisati o nekim od funkcionalnostima koje PHP nudi.

PHP se može koristiti za skriptiranje pomoću naredbenog retka u Unix i Windows operacijskim sustavima i za izradu više platformskih (eng. cross platform) stolnih aplikacija sa grafičkim korisničkim sučeljem pomoću ekstenzije PHP-GTK. Glavna namjena PHP-a je da se koristi za programiranje na strani poslužitelja.

PHP se može koristiti na većini operacijskih sustava od kojih su najrasprostranjeniji Windows, Linux i macOS. Važna značajka je da ima podršku za širok spektar sustava za upravljanje bazama podataka. Povezivanje web stranice sa bazom podataka veoma je jednostavno korištenjem ekstenzija za određenu bazu podataka kao što je MySQL Improved Extension za MySQL. Kao izlaz osim HTML-a, PHP može stvarati slike, PDF i XML dokumente te mnoge druge. Jedan od najjačih aduta PHP-a su okviri i mikro okviri koji nude biblioteke, prakse čistog koda, brži razvoj, veću sigurnost itd. [3,4]

Postoji još mnogo funkcionalnosti i razloga zbog kojih PHP već dugi niz godina sjedi na tronu i dominira kao najbolji jezik za programiranje na poslužiteljskoj strani. U nastavku će se prikazati dijagram arhitekture web aplikacija temeljenih na PHP-u.



Slika 2: Dijagram arhitekture web aplikacija temeljenih na PHP-u (Izvor: [5])

Na slici 2 prikazan je dijagram arhitekture koji prikazuje kako su komponente međusobno povezane te kako se razmjenjuju podaci između istih.

Korisnik koristi web preglednik, tipa Google Chrome, kojim zahtijeva (eng. request) pristup određenoj web stranici. Korisnik upisuje URL web stranice koju želi posjetiti i šalje zahtjev prema web poslužitelju. Taj zahtjev dolazi do web poslužitelja, tipa Apache te taj web poslužitelj uzima tu stranice iz korijena te web stranice. Ako se radi o PHP datoteci, Apache web poslužitelj šalje tu datoteku PHP interpreteru koji taj kod prevodi i izvršava. Ukoliko je u kodu pozvana baza podataka, interpreter uzima potrebne podatke iz baze te ih koristi kako bi generirao odgovor (eng. response). Taj odgovor poslan je Apache web poslužitelju, a Apache šalje isti Google Chromu koji na koncu odgovor prikazuje korisniku na ekranu. [5]

2.1. Okviri u PHP-u

Razvoj web aplikacije od nule, popularan je kod mladih i neiskusnih programera koji žele naučiti jezik i pokazati znanje implementacijom vlastitih aplikacijskih logika i koncepta. Iako je to dobar način da se individualno prikupi znanje, u praksi takve web aplikacije nemaju najbolju budućnost.

Još od samih početaka, jedan od najvećih problema kod izrade PHP web aplikacija je sigurnost. Generalno gledano, sigurnost web aplikacija veliki je problem te je veoma važno spriječiti sigurnosne propuste da treća osoba pridobi pristup osjetljivim korisničkim podacima.[6]

Dva najčešća napada na web aplikacije su [7]:

- XSS – (eng. Cross site scripting) – vrsta napada kojim se zlonamjeran kod ubrizgava putem web obrasca zbog loše provjere valjanosti unešenih korisničkih podataka.
- SQL Injection – ubrizgavanje dijela zlonamjernog SQL koda u URL stranice ili u web obrazac.

Postoji još mnogo načina kojim zlonamjerne osobe mogu penetrirati web aplikaciju. Sigurnost je jedan od važnih aduta okvira. Iako okviri ne pokrivaju svaki način i metodu napada, u znatnoj mjeri olakšavaju zaštitu web aplikacije. [8]

Što je zapravo okvir? Okvir (eng. Framework) se može definirati kao kostur aplikacije kojega programer može koristiti i modificirati prema svojim potrebama. Još jedna definicija govori da je okvir dio ili cijeli dizajn sistema koji se može višekratno koristiti, a predstavljen je skupom apstraktnih klasa i načinom interakcije između njihovih instanci. [9]

Naizgled, ove definicije zvuče različito, no prva definicija opisuje svrhu, dok druga strukturu okvira. Okviri razvojnim inženjerima služe kako bi olakšali izradu same web aplikacije. Inženjeri koji kreiraju okvire, odgovorni su za razvijanje i održavanje tih okvira koji imaju dobro definirane funkcionalnosti i u teoriji ti inženjeri imaju bolje i dublje znanje za specifične funkcionalnosti i zadatke koje ti okviri pokrivaju. U prijevodu, to znači da okviri korisniku garantiraju temeljito razrađenu funkcionalnost nekog dijela koda. Osim toga, okviri sa sobom nose još mnoge prednosti [10, 11, 12]:

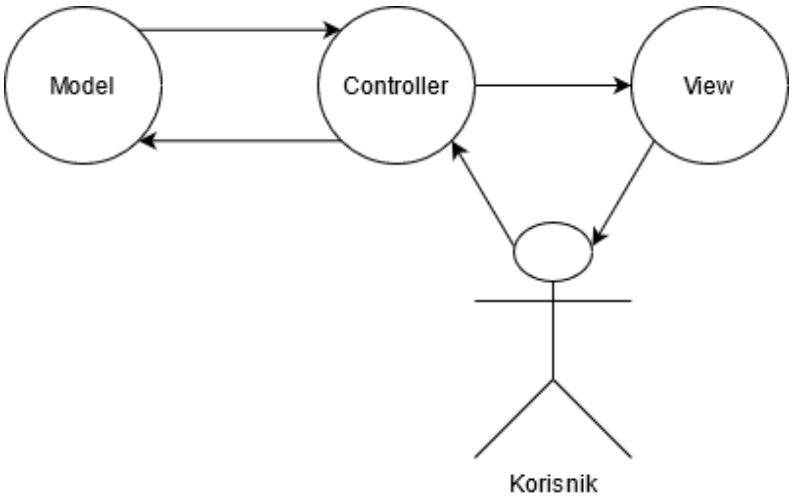
- Brzina razvijanja – kako bi se skratilo vrijeme razvijanja aplikacije, okviri daju razvojnom programeru na korištenje gotove funkcionalnosti i dijelove koda.
- Jednostavnost održavanja – korištenjem arhitektura kao što je MVC (eng. Model-View-Controller), dijelovi aplikacije dijele se u modele (poslovna logika i podaci aplikacije),

kontrolere (korisnički zahtjevi) i poglede (prezentacija podataka korisniku) za lakši pregled, održavanje i implementaciju.

- Automatizacija uobičajenih zadataka – funkcionalnosti i alati koji pomažu da se automatiziraju uobičajeni zadaci kao što su pred memoriranje, rad s URL adresama, sesije, autentikacija itd.
- Sigurnost – korištenje gotovih mehanizama kako bi programer lakše zaštitio web aplikaciju od sigurnosnih prijetnji i propusta.
- Manja cijena – pošto su okviri otvorenog koda, upotrebljavanjem gotovih funkcionalnosti ubrzava se sama izrada aplikacije bez povećavanja troškova.

Iz samih karakteristika, može se uočiti da okviri sa sobom nose mnoge pogodnosti kod korištenja. Svakom inženjeru web aplikacije, same riječi brže i lakše donose rasterećenje. Manje vremena za razvoj, jednostavnije održavanje i veća sigurnost veliki su benefiti.

Postoji veoma mnogo okvira cijelog spektra za razvoj PHP web aplikacija od kojih su neki od najpopularnijih: Laravel, CodeIgniter, Symfony i Phalcon. Phalcon je PHP okvir otvorenog koda optimiziran za visoke performanse. Prije korištenja okvira potrebno je instalirati potrebne programe, pripremiti i konfigurirati radno okruženje te projektnu strukturu. Phalcon, kao i većina okvira koristi MVC arhitekturu te ima mnoge komponente koje je moguće koristiti od kojih su neke: komponente za zahtjev (HTTP GET, POST), spremanje i izbjegavanje grešaka, kriptiranje podataka, preusmjerivač, komponenta sesije i priručne memorije i još mnoge. [13]



Slika 3: Izgled MVC strukture (Izvor: [12])

Prethodno je navedeno da korištenje arhitekturalnog uzorka kao što je MVC donosi prednost kod korištenja okvira. MVC, prema slici 3., sastoji od tri koncepta [12]:

- model – predstavlja indvidualan entitet baze podataka;
- pogled – predstavlja predložak koji prikazuje podatke krajnjem korisniku;
- kontroler – uzima korisničke zahtjeve iz preglednika, dobiva podatke od modela, provjerava unos i šalje korisniku odgovor tj. prikaz.

Okviri temeljeni na MVC uzorka imaju potpunu separaciju korisničkog sučelja i poslovne logike. Kod izrade kompleksnih web aplikacija, takvi okviri su veoma korisni, ali kod manjih web aplikacija, takvi okviri su pretjerani. Pošto su to kompleksni okviri, potrebno je vremena da se savlada i nauči takav okvir, naravno sa razumijevanjem MVC arhitekture. Iako su okviri dizajnirani da unaprijede performanse web aplikacije, količina alata i komponenata zapravo može dovesti do opadanja performansi kod manjih i jednostavnijih web aplikacija. Kod takvih aplikacija, bolji izbor je korištenje mikro okvira. [9, 11, 14]

2.2. Mikro okviri

Kako su se okviri s vremenom razvijali, tako je rasla i njihova opsežnost. Okviri su postajali sve veći i složeniji te su nudili mnoga svojstva. Za kompleksne web aplikacije taj rast nudio je još više olakšica i mogućnosti koje pogrameri mogu koristiti i time olakšati sebi posao. Međutim, nezgodna strana razvoja okvira je korištenje istih kod izrade jednostavnijih web aplikacija. Takvi okviri jednostavno su postali pretjerano “napuhani” sa funkcionalnostima za jednostavnije projekte te su zbog takvih okolnosti nastali mikro okviri. [15]

Mikro okviri su alternative kod izrade web aplikacija sa strukturom i brzinom pravih okvira, no sa manje značajka i alata. Mikro okviri se ponajviše koriste za manje projekte i posebne slučajeve kada je potreban brz razvoj, jednostavna implementacija i testiranje te puštanje same aplikacije u rad. Ne postoji neko pravilo po kojem korisnik odabire korištenje okvira ili mikro okvira. To je u potpunosti osobna odluka koja varira od projekta do projekta, ovisno o funkcionalnostima koje će se implementirati pomoću tih okvira. [15, 16]

Funkcionalnost koju pokriva skoro svaki mikro okvir je usmjeravanje. Sintakse za usmjeravanje, koje se koriste kod pojedinog okvira veoma su slične i koriste isti koncept. Kod najjednostavnijeg usmjeravanja, korisnik upisuje URI, te na temelju tog identifikatora okvir izvršava funkciju koja vraća neki izlaz.

2.3. Razlika između okvira i mikro okvira

Kako bi lakše prikazali razliku okvira i mikro okvira, izrađena je tablica u kojoj su okvirno prikazane razlike između različitih aspekata okvira.

Tablica 1: Razlika okvira i mikro okvira [17]

	Okviri	**Mikro okviri**
Složenost okvira	Kompleksni okviri sa mnogim alatima i modulima	Minimalan kod okvira i jednostavno korištenje
Komponente okvira	Usmjeravanje i obrada zahtjeva, MVC, sigurnost, ORM, validacija i sanitacija, upravljanje sesijama itd.	Ovisno o okviru, sadrži nekolicinu komponenata (usmjeravanje, predlošci, globalne varijable itd.)
Primjena okvira	Srednje i velike web aplikacije	Male web aplikacije i specifični značajke
Krivulja učenja okvira	Zbog složenosti, potrebno je vremena da bi se savladali	Jednostavni za naučiti
Razvoj projekata	Spor razvoj zbog većih zahtjeva	Brz i fleksibilan razvoj

Jedna zanimljiva usporedba okvira i mikro okvira je da se okvir promatra kao veliki SUV automobil sa najmodernijom opremom, a mikro okvir kao mali sportski auto. Okvir ima mnoge opcije i značajke, no isto tako je glomazan i spor, dok mikro okvir nema mnoge značajke na svojoj strani, ali ima lakoću, brzinu i okretnost.

Okviri su relativno složeni te nude veliki repertoar alata. Koriste mnoge komponente i strukture koje su kod velikih web aplikacija korisne jer imaju sve što korisnik treba za izradu dobrih i organiziranih web aplikacija. No, zbog izrazito velikog broja komponenata i strukturiranja projekta, okviri nisu fleksibilni te za učenje okvira treba uložiti vremena i truda kako bi se savladali. Mikro okviri koriste iste koncepte, ali su potpuna suprotnost okvirima. Sadrže puno manje komponenata koje se mogu koristiti u izradi. Nemaju veliku složenost te se mogu brzo naučiti i olakšati programiranje web aplikacija. Najčešće se koriste kod manjih projekata gdje nije potreban veliki temelj aplikaciji i sam okvir ne treba odraditi većinu posla odnosno razvojni programer ima više slobode kod pisanja i organiziranja aplikacije.

Neki mikro okviri koriste MVC strukturu, ali ne striktno već odluku ima korisnik da strukturira projekt po svojoj želji. [15,16,17]

3. Primjena mikro okvira

U ovom poglavlju će se opisati način na koji se instaliraju mikro okviri i koji alati su potrebni za instalaciju ovisno o operacijskom sustavu koji se koristi. Prikazat će se neki mikro okviri i njihove značajke, dokumentacija i zajednica. Na koncu će se napraviti usporedba te argumentirati odabir mikro okvira koji će biti korišteni u izradi web aplikacije.

3.1. Instalacija mikro okvira

Prije nego što se mikro okvir može koristiti potrebno je nekoliko programa i alata. Naravno, potreban je PHP kako bi se uopće mogao interpretirati kod u web aplikaciju. Na Windows operacijskim sustavima često se koristi XAMPP. XAMPP je paket otvorenog koda kojega je razvio Apache te se sastoji od poslužitelja, baze podataka i PHP interpretera za skripte napisane u PHP-u. [18]

Nakon instalacije interpretera za PHP, potreban je alat Composer. Composer je alat koji se koristi za instalaciju biblioteka i okvira. Kod instalacije nekog paketa pomoću navedenog alata, taj paket se ne instalira globalno već u direktorij projekta u kojem se taj paket namjerava koristiti. Postoji više načina instaliranja paketa pomoću Composera ovisno o naredbi. Nadalje će se objasniti i prikazati instalaciju pomoću terminala. [19]

Kako bi se pomoću Composer-a instalirao paket, potrebno je pomoću terminala navigirati se do korijenskog direktorija projekta i upisati sljedeću naredbu:

```
composer init
```

Zatim će se prikazati konfiguracijski generator Composer-a koji traži unos naziva projekta, opisa i autora (svi podaci su neobavezni) te po završetku kreira composer.json datoteku u korijenskom direktoriju projekta. Ispod je prikazan izgled .json datoteke koju kreira Composer. U ključ “require” upisujemo nazive i verzije paketa odnosno okvira koje želimo instalirati u projekt. Zatim u terminal upisujemo naredbu:

```
composer update
```

koja dohvaća okvire sa githuba te u korijenu projekta kreira mapu vendor u koju se ti okviri na koncu i spremaju. [19]

```
{
      “name”: “sakac/”primjer”,
      “authors”: [{
       “name”: “Sakac”,
       “email”: msakac@foi.hr
      }],
      “require”:{
            “laravel/lumen-framework”: ”7.0”,
            “bcasca/fatfree”: “3.7.3”
      }
}
```

3.2. Fat free

Fat free također poznat i kao F3 je mikro okvir koji se ugrađuje u PHP projekte za izradu web aplikacija. Koristi moderne alate i biblioteke koji pomažu izraditi pouzdan i lako održiv PHP kod. Okvir je veoma moćan te se u samo nekoliko linija koda može puno postići. F3 je lak za upotrebu te je prilagođen korisnicima. Uz dobro znanje PHP-a, ovaj okvir može se relativno brzo naučiti i savladati zahvaljujući veoma dobroj korisničkoj dokumentaciji. Također na Youtube dostupni su mnogi profesionalni tutorijali koji olakšavaju proces savladavanja ovog mikro okvira. [20]

Pri izradi aplikacije pomoću F3 okvira, korisnik ima potpunu slobodu odabira načina na koji će strukturirati projekt. F3 sa sobom donosi mnoge funkcionalnosti:

- Usmjerivač (eng. Routing engine);
- F3 globalne varijable i konfiguracijske datoteke;
- Predlošci i pogledi (eng. Template engine);
- Rad s bazama podataka;
- Plug-ins;
- Testiranje.

Fat free također poznat i kao F3 je mikro okvir koji se ugrađuje u PHP projekte za izradu web aplikacija. Koristi moderne alate i biblioteke koji pomažu izraditi pouzdan i lako održiv PHP kod. Okvir je veoma moćan te se u samo nekoliko linija koda može puno postići. F3 je lak za upotrebu te je prilagođen korisnicima. Uz dobro znanje PHP-a, ovaj okvir može se relativno brzo naučiti i savladati zahvaljujući veoma dobroj korisničkoj dokumentaciji. Također na Youtube dostupni su mnogi profesionalni tutorijali koji olakšavaju proces savladavanja ovog mikro okvira. [20]

3.2.1. Usmjeravanje

Usmjeravanje u F3 mikro okviru je veoma jednostavno i prilagođeno programeru za što lakšu upotrebu. Prije korištenja bilo koje funkcije, potrebna je izraditi instancu glavne klase F3 okvira:

```
$f3 = Base::instance();
```

Nakon instanciranja koristimo objekt F3 kako bi pozivali predefinirane funkcije. Jedna od najvažnijih funkcija je “route()”.

```
$f3->route('GET /pocetna', function(){
      echo 'Ovo je pocetna stranica!';
});
$f3->run();
```

Nakon što korisnik u pretraživač upiše neki URI, mikro okvir provjerava da li postoji ta ruta te ako postoji poziva funkciju koja u ovom slučaju ispisuje tekst na ekran. Naravno, nakon što definiramo rute moramo pozvati funkciju “run()” koja pokreće mikro okvir. Instanciranje glavne klase i pokretanje okvira najčešće se radi u korijenu projekta u index.php datoteci. Kako bi bolje strukturirali projekt, pomoću “route()” funkcije možemo u jednoj liniji instancirati klasu i pozvati njezinu metodu koja prikazuje stranicu.

```
require("klase/KlasaPocetna.php");
$f3->route('GET /pocetna', KlasaPocetna -> prikazi);
```

Kako ne bi trebali svaki put dohvaćati direktorij u kojem je spremljena klasa, F3 sadrži Autoloader koji osim što štedi korisnika od konstantnog uključivanja direktorija u kojemu su spremljene klase, štedi memoriju od nepotrebnog uključivanja. Tek kada vidi poziv neke klase autoloader će automatski uključiti tu datoteku u kod. Kako bi koristili funkciju autoloadera, kreira se jedan direktorij u kojem se spremaju sve klase. Jedino ograničenje je da u jednoj datoteki smije biti samo jedna klasa i da naziv datoteke mora biti identičan nazivu klase u toj datoteci.

```
$f3->set('AUTOLOAD', 'klase/');
```

Korištenjem autoloadera kako bi pozivali željene klase već se dobiva bolje strukturiranje objekata. Osim toga korisničko sučelje odnosno HTML sadržaj također možemo staviti u poseban direktorij te time dobivamo odvojen korisnički sadržaj i kontrolere što pozitivno utječe na organizaciju i strukturu cijelog projekta. Postoje još dvije veoma zanimljive funkcije, “beforeroute()” i “afterroute()” koje se pozivaju prije i nakon usmjeravanja. [20]

3.2.2. Globalne varijable i konfiguracijske datoteke

F3 sa sobom donosi vrlo korisnu funkcionalnost definiranja varijabli. Funkcija “set()” inicijalizira varijablu koja može biti bilo koje vrste, niz ili objekt.

```
$f3->set('naziv', vrijednost);
```

Ono što čini ovu funkciju veoma korisnom je to da se inicijalizirane varijable mogu dohvatiti u bilo kojem dijelu koda te ih to čini globalnim varijablama. Također tu su i funkcije get() koja dohvaća varijablu, “clear()” koja briše varijablu iz memorije i “exist()” koja provjerava da li varijabla uopće postoji.

Ovaj okvir također ima veliki popis predefiniranih super globalnih varijabli kao što su SESSION i COOKIE koje su ustvari ekvivalentne PHP-ovim $_SESSION i $_COOKIE varijablama. Osim tih postoji još mnogo predefiniranih varijabli od kojih je već prethodno spomenuta AUTOLOAD. Još neke od važnijih varijabli su UI koja sadrži lokaciju HTML pogleda i predložaka, DEBUG koja daje informacije kod otklanjanja pogrešaka, MODELS koja sadrži lokaciju modela odnosno klase mapirane prema entitetima baze podataka i CACHE koja aktivira korištenje predmemorije.

Konfiguracijske datoteke još su jedan zanimljiv dodatak koji daje na jednostavnosti čitanja i održavanja koda. U F3 okviru postoje 4 predefinirane sekcije:

- [globals] – sekcija za definiranje globalnih varijabli;
- [routes] – sekcija za definiranje ruta za usmjerivač;
- [maps] – sekcija za definiranje mape ruta;
- [redirects] – sekcija za definiranje ruta preusmjeravanja.

Za svaku od tih sekcija možemo izraditi zasebnu .ini konfiguracijsku datoteku te ih dodati u korijensku “index.php” datoteku pomoću funkcije “config()”. [20]

Primjer izgleda datoteke “globalneVarijable.ini” u kojem se definiraju globalne varijable:

```
[globals]
UI = app/views
AUTOLOAD = app/controllers
MODELS = app/models
DEBUG = 3 //vrijednost od 0 do 3, veća vrijednost više informacija
CACHE = true
nekaVarijabla = "varijabla"
```

Primjer izgleda datoteke “rute.ini” u kojem se definiraju rute:

```
[routes]
GET /pocetna=PocetnaKontroler->prikazi
GET /login=LoginKontroler->prikazi
POST /registriraj ->RegistracijaKontroler->registriraj
Primjer izgleda korijenske index.php datoteke:
```

```
$f3 = Base::instance();
$f3->config('globlaneVarijable.ini');
$f3->config('rute.ini');
$f3->run();
```

3.2.3. Predlošci

Korisničko sučelje trebalo bi biti neovisno o poslovnoj logiki projekta. Mješanjem korisničkog sučelja i poslovne logike u jednoj datoteci dobivamo kod koji je težak za održavanje. Zbog toga razloga F3 ima funkcionalnost predložaka.

```
<p>Bok, <?php echo $ime; ?> !</p> //php template
<p>Bok, @{{ime}} !</p> //f3 template, naziv dokumenta bok.html
```

Ovaj predložak kombiniran sa globalnim varijablama donosi veliki prednost kod. Iako i dalje mješamo logiku i sučelje kod se veoma jednostavno čita. Kao što je prethodno rečeno F3 globalne varijable dostupne su u bilo kojem dijelu koda. [20]

```
$f3->set('ime', 'Pero');
$template = new Template();
echo $template->render('bok.html');
```

U navedenom kodu postavili smo varijablu ime na vrijednost Pero te kreirali novu instancu klase “Template()” i funkcijom “render()” prikazali dokument “bok.html” korisniku.

3.2.4. Rad s bazama podataka

F3 je dizajniran na način da čini posao povezivanja i rada sa bazom podataka što lakšim i jednostavnijim. F3 podržava SQL baze podataka kao što su MySQL, SQLite, PostgreSQL itd. Također podržava i MongoDB baze podataka.

Konekcija sa SQL bazom podataka ostvaruje se na sljedeći način:

```
$db = new DB\SQL(
      'mysql:host=localhost;port=3307;dbname=baza',
      'root',
      ''
);
```

Pomoću metode “exec()” izvršavamo upite nad bazom podataka. Jedna korisna funkcija koju F3 nudi je “log()” koja vraća sve napravljene upite nad bazom. Kao što je ranije navedeno SQL ubrizgivanje velika je prijetnja svim web aplikacijama. F3 taj problem rješava parametriziranjem upita.

```
$db->exec(
      ‘SELECT * FROM korisnik WHERE id=?’,
      $f3->get(‘POST.id’)
);
```

Korištenjem ovakvih upita štitimo web aplikaciju od zlonamjernih napada koje mogu imati neželjene posljedice. F3 također podržava veoma jednostavne objektno relacijske mapere između aplikacije i baze podataka koji olakšavaju i ubrzavaju pisanje koda koji se bavi CRUD operacijama odnosno kreiranje, čitanje, ažuriranjem i brisanjem.

```
$korisnik = new DB\SQL\Mapper($db, ‘korisnik’);
$korisnik->load(array(‘id=?’, ‘1’));
```

Prva linija koda kreira instancu mapera koji komunicira sa entitetom korisnik unutar baze podataka. U pozadini, F3 dohvaća strukturu entiteta korisnik i određuje koji su koji atributi. Drugom linijom, odnosno metodom “load()” dohvaćamo podatke iz baze podataka. Kako bi dodali pomoću mapera podatke u bazu podataka, koristimo metodu “save()” te nakon spremanja trebamo obrisati podatke iz mapera pomoću metode “reset()”.

```
$korisnik->id = ‘2’;
$korisnik->ime = ‘Pero’;
$korisnik->save();
$korisnik->reset();
```

Korištenjem F3 mapera, klasa koja se kreira nad određenim entitetom baze podataka, dobiva nazive varijabla iste kao što su navedeni atributi u entitetu. Ukoliko napravimo promjenu nad tim entitetom, odnosno dodamo ili obrišemo redak, Fat Free će automatski sinkronizirati novi strukturu tablice sa objektnim maperom.

```
$korisnik->load(array(‘id=?’, ‘1’));
$korisnik-> erase();
```

Pomoću metode “erase()”, brišemo redak iz entiteta. Nadalje biti će prikazan jedan jednostavan upit pomoću metoda mapera. [20]

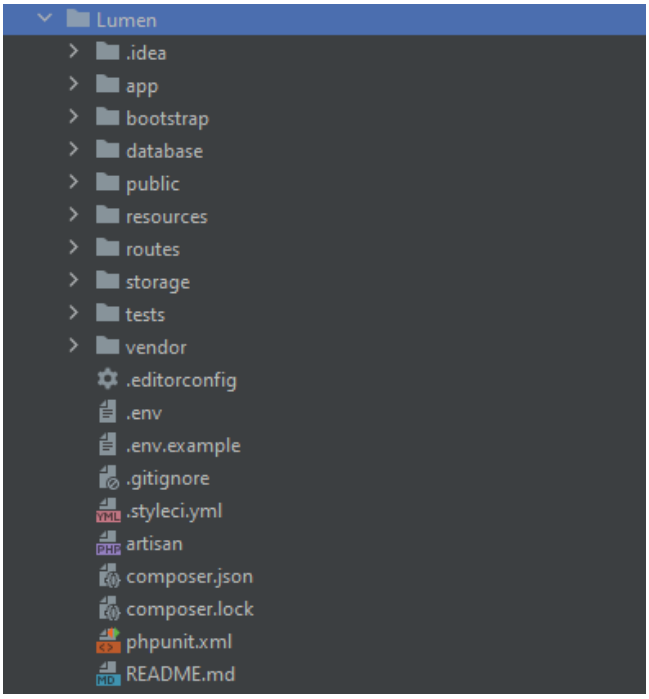
3.3. Lumen

Lumen je brz i popularan PHP mikro okvir kojeg je razvio Taylor Otwell. Isti autor razvio je i okvir Laravel. Lumen nasljeđuje puno funkcionalnosti od svog velikog brata Laravela. Kod razvijanja projekta, ako postoji potreba da se nadogradi na pravi okvir, prelazak sa Lumena na Laravel je veoma jednostavan jer oba okvira koriste iste komponente osim što Laravel nudi puno više nego Lumen. Lumen se također često koristi za izradu mikro servisa i API-ja baziranih na Laravelu.

Lumen sadrži slične funkcionalnosti kao i F3 mikro okvir. Najveća razlika je u načinu strukturiranja projekta. Naime kod instalacije Lumena u korijen projekta, Lumen ima mnoge predefinirane direktorije i datoteke koje se koriste kod izrade aplikacije. Programer nema toliku slobodu strukturiranja projekta na svoj način kao što ima u Fat free mikro okviru. Prednost Lumena je ta da ima veoma dobru dokumentaciju koja uvelike pomaže kod savladavanja okvira. Ukoliko kod izrade aplikacije programer naiđe na frustrirajući problem, Laravel ima veliku zajednicu korisnika spremnih na međusobnu pomoć. [21]

3.3.1. Struktura Lumena

Instalacijom Lumen-a, kao što je prethodno spomenuto, kreira se direktorij vendor u koji je Lumen instaliran i dodaci koji dolaze sa njim. No osim direktorija “vendor”, kreiraju se još mnogi direktoriji i u njima gotove datoteke. Na slici 4 prikazan je izgled strukture projekta u kojem je instaliran Lumen.



Slika 4: Struktura Lumen mikro okvira

Kao što se može vidjeti, Lumen je kreirao još poprilično direktorija. Od kreiranih direktorija važni su:

- app/Http/Controllers/ - direktorij u kojem su spremljene klase kontrolera;
- public/ - korijenski direktorij u kojem se nalazi index.php;
- resources/views/ - direktorij u kojem se nalaze korisnička sučelja;
- routes/ - direktorij u kojem se nalaze rute za usmjeravanje.

3.3.2. Usmjeravanje, kontroleri i pogledi

Kao što je prethodno spomenuto, Lumen dolazi sa gotovim direktorijima u kojem se već nalaze neke datoteke. Lumen dolazi sa prekonfiguriranim Autoloader-om. U direktoriju bootstrap/ nalazi se datoteka “app.php” koja konfigurira Autoloader i direktorije u kojima se nalaze kontroleri, rute i middleware. Middleware su klase odnosno metode unutar klasi koje se mogu pozivati prije i poslije svakog korisničkog zahtjeva ili prema specifičnim rutama, a slične su F3 funkcijama “beforeroute()” i “afterroute()”. U direktoriju “public/”, datoteka “index.php” dohvaća datoteku “app.php” i pokreće Lumen.

Rute se navode unutar direktorija “routes/” u “web.php”:

```
$router->get('/pocetna', function() user ($router){
      return "Pocetna stranica";
});
```

Sintaksa okvira je slična F3 okvira. Za prikaz korisničkog sučelja možemo koristiti kontrolere, no prvo ih treba kreirati. Kao i u F3, naziv kontrolera mora biti jednak nazivu klase koju taj kontroler sadrži te svaki novi kreirani kontroler nasljeđuje glavni kontroler koji je kreiran od strane Lumena.

```
class PocetnaKontroler extends Controller{
      public function prikazi(){
            return 'Pocetna stranica';
            //ispisuje poruku na novoj stranici
      }
      public function prikaziPogled(){
            return view('pocetnaPogled');
            //prikazuje pocetnaPogled.php stranicu
      }
}
```

U datoteci sa rutama:

```
$router->get('/pocetna', 'PocetnaKontroler@prikazi');
$router->get('/pocetna2', 'PocetnaKontroler@prikaziPogled');
```

Sve poglede odnosno korisnička sučelja kreiramo u direktoriju “resources/views/”. Naravno, nije potrebno slijediti strukturu koju je kreirao Lumen, ali kod kreiranja vlastite strukture projekta može doći do neželjenog ponašanja okvira. [21]

3.3.2. Baze podataka i ostale značajke

Lumen trenutno podržava četiri baze podataka:

- MySQL
- PostgreSQL
- SQLite
- SQL Server

Pristup bazi podataka i izrada upita veoma je jednostavna korištenjem gotovih funkcija Lumen-a. Kako bi se dobio pristup bazi podataka, Lumen kreira datoteku “.env.example”. Ta datoteka je samo primjer te programer samostalno izradi datoteku .env sa istim sadržajem te se promjeni konfiguracija za pristup bazi:

```
DB_CONNECTION = mysql
DB_HOST = 127.0.0.1
DB_PORT = 3307
DB_DATABASE = bazaPrimjer
DB_USERNAME = root
DB_PASSWORD =
```

Bazi se može pristupati pomoću na sljedeći način:

```
$results = app('db')->select("SELECT * FROM korisnici);
```

Ukoliko se u “bootstrap/app.php” dadoteku dodaju sljedeće linije koda dobije se pristup pojednostavljenom pristupu bazi podataka pomoću sučelja DB. Druga linija koda omogućava pristup Laravel-ovoj Eloquent ORM funkcionalnosti objektno relacijskog mapiranja baze podataka.

```
$app->withFacades(); //pristup bazi uz jednostavnije sučelje DB
$app->withEloquent(); //korištenje objektno relacijskog mapiranja

$results = DB::select("SELECT * FROM korisnici);
//pojednostavljeni pristup bazi podataka
```

Osim navedenih Lumen nudi još mnoge funkcionalnosti od kojih su neke:

- Predmemoriranje;
- Autorizacija i autentikacija;
- Enkripcija;
- Dnevnici (eng. Logger);
- Testiranje.

3.4. Flight

Flight je brz, jednostavan i proširiv mikro okvir koji se najčešće koristi za izradu RESTful web aplikacija. Flight je dizajniran kako bi bio korisnički proširiv. Dolazi sa predefiniranim metodama i komponentama, ali dopušta izradu vlastitih klasa i metoda te nadjačavanje već postojećih. Kao i svi prethodni okviri, instalaciju vršimo pomoću composera. Instalacijom se kreira samo direktorij vendor u kojoj je sami Flight, što znači da se projekt može strukturirati po želji.

Funkcionalnosti koje se mogu koristiti pomoću ovog mikro okvira su:

- Usmjeravanje;
- Samostalno proširivanje;
- Definiranje globalnih varijabli;
- Pogledi.

Flight ima svega desetak temeljnih i proširivih metoda. Za izradu vlastitog okvira, Flight daje dobar temelj na kojem se može graditi. [22]

Vidljivo je da kod svih mikro okvira postoji uzorak. Flight kao i prethodno navedeni mikro okviri pomažu programeru kako bi mogao koristiti osnovne funkcionalnosti (usmjeravanje, pogledi, varijable) koju svaka web aplikacija ima u svojem kodu.

3.5. Limonade

Limonade je PHP mikro okvir za brzi razvoj web aplikacija i prototipiranje. Inspiriran je okvirom Sinatra u Ruby-u te je cilj da bude što jednostavniji, lakši i fleksibilniji. Nakon instalacije mikro okvira u korijen projekta, kreira se index.php datoteka, dohvati se datoteka "limonade.php" i mikro okvir se može početi koristiti.

```
require_once 'lib/limonade.php';
dispatch('/pocetna', 'poruka');
function poruka(){
      return "Pocetna stranica";
}
run();
```

Funkcija "dispatch()" je ista kao i funkcija "dispatch_get()". Za slanje podataka koristi se funkcija "dispatch_post()". Nakon definiranih ruta mikro okvir se pokreće pomoću funkcije "run()". U rutu možemo staviti i neki parametar koji se onda dohvaća pomoću funkcije "params()".

Također kod usmjeravanja mogu se pozivati statičke klase odnosno instance klasa i njihove metode. Postoje funkcije "before()" i "after()" koje imaju istu ulogu kao i kod F3 okvira. Osim usmjeravanja Limonade ima potporu za poglede i predloške. Za postavljanje lokacije pogleda odnosno korisničkog sučelja koristi se funkcija "option()" s ključem "views_dir". Također možemo postavljati globalne varijable sa funkcijom "set()" odnosno "set_or_default()". [23]

```
option('views_dir', 'direktorij');
dispatch('/bok/:ime', prikazi);
function prikazi(){
      set_or_default('ime', params('ime'), 'Niko');
      //kombinacija funkcije set() i params(), ukoliko nije navedeno
      ime zadana vrijednost je 'Niko'
      return "Bok $ime";
}
```

Mikro okvir Limonade nema najbolju dokumentaciju, ali pošto se radi o malom mikro okviru razvijenom od strane nadahnutih programera, to je razumljivo. Situaciju spašavaju gotovi primjeri koji dolaze zajedno sa mikro okvirom pa je uz korištenje oba lakše savladati ovaj mikro okvir.

3.6. PHPixie

PHPixie nastao je sredinom 2012. godine kao mikro okvir na temelju okvira Kohana, dizajniran za jednostavne web stranice za čitanje sa fokusom na performanse. Ono što je interesantno kod ovog mikro okvira je to da je sa vremenom, postepeno nadrastao granice mikro okvira, te se danas smatra pravim okvirom punog spektra koji je zadržao visoke performanse mikro okvira. Danas, PHPixie nudi mnoge komponente:

- Autentikacija pomoću kriptografski sigurnih praksa;
- Usmjeravanje sa PSR-7 za HTTP zahtjeve i odgovore;
- Proširivi sistem predložaka;
- Biblioteka za manipulaciju slikama itd.

Također PHPixie sa sobom nosi mnoge prednosti kao što su [24, 25]:

- Brzina razvijanja i vrijeme učitavanja stranica;
- Stabilna i prilagodljiva arhitektura projekta;
- Jednostavno učenje i prilagođavanje za korisnike sa iskustvom okvira Codeigniter ili Kohana itd.

No ovdje se postavlja jedno zanimljivo pitanje. Zašto koristiti mikro okvire koji će možda tokom vremena prerasti domenu mikro okvira te postati pravi okviri? Ovime, na neki način, mikro okviri gube svoju smisao, jer korištenjem mikro okvira koji je prerastao u okvir, mogao se koristiti neki dokazani okvir koji ima dobru zajednicu, dokumentaciju te minimalan broj grešaka i propusta u kodu. Naravno ta odluka varira od korisnika do korisnika, te sami razlozi korištenja okvira ili mikro okvira ovise o zahtjevima i veličini projekta koji se nastoji realizirati.

3.7. Odabir mikro okvira

Nakon proučavanja više mikro okvira, odabir pravoga za izradu neke web aplikacije nije jednostavan jer svi mikro okviri manje-više programeru na korištenje daju iste glavne funkcionalnosti. Usmjeravanje, korištenje pogleda i globalne varijable komponente su koje skoro svi mikro okviri podržavaju. No ipak, neki mikro okviri nude veći repertoar funkcionalnosti i priključaka od drugih te bolju dokumentaciju što odmah uveliko utječe na odabir samog mikro okvira.

Za izradu praktičnog dijela odabrani su okviri Fat Free i Lumen. Fat Free je veoma lagan i jednostavan mikro okvir koji nudi dobro razrađenu dokumentaciju, strukturiranje projekta po vlastitoj želji i dobar set alata koje možemo lagano i efektivno koristiti. Jednostavno korištenje

globalnih varijabli i ruta sa kontrolerima te separacija u odvojene konfiguracijske datoteke daje odličnu pregled projekta i jednostavnost u izradi. Lumen nudi još bolju dokumentaciju od F3 mikro okvira, veliku zajednicu te dijeli funkcionalnosti sa Laravelom. Iako nije jednostavan za naučiti i realizirati na projektu zbog svoje strukture i količine funkcionalnost, znanje takvog mikro okvira otvara vrata prema Laravelu koji je iznimno popularan te jedan od najboljih okvira za PHP.

4. Izrada web aplikacije

Za izradu praktičnog dijela ovog rada bilo je potrebno izraditi web aplikaciju korištenjem mikro okvira u PHP-u. U ovom poglavlju opisana je web aplikacija odnosno sama tema web aplikacije, korištene tehnologije za izradu, model baze podataka i korisničke uloge. Također, prikazani su dijelovi koda i funkcionalnosti koje su izrađene pomoću mikro okvira, a na koncu je prikazan osvrt i mišljenje na prednosti i nedostatke u radu sa više mikro okvira.

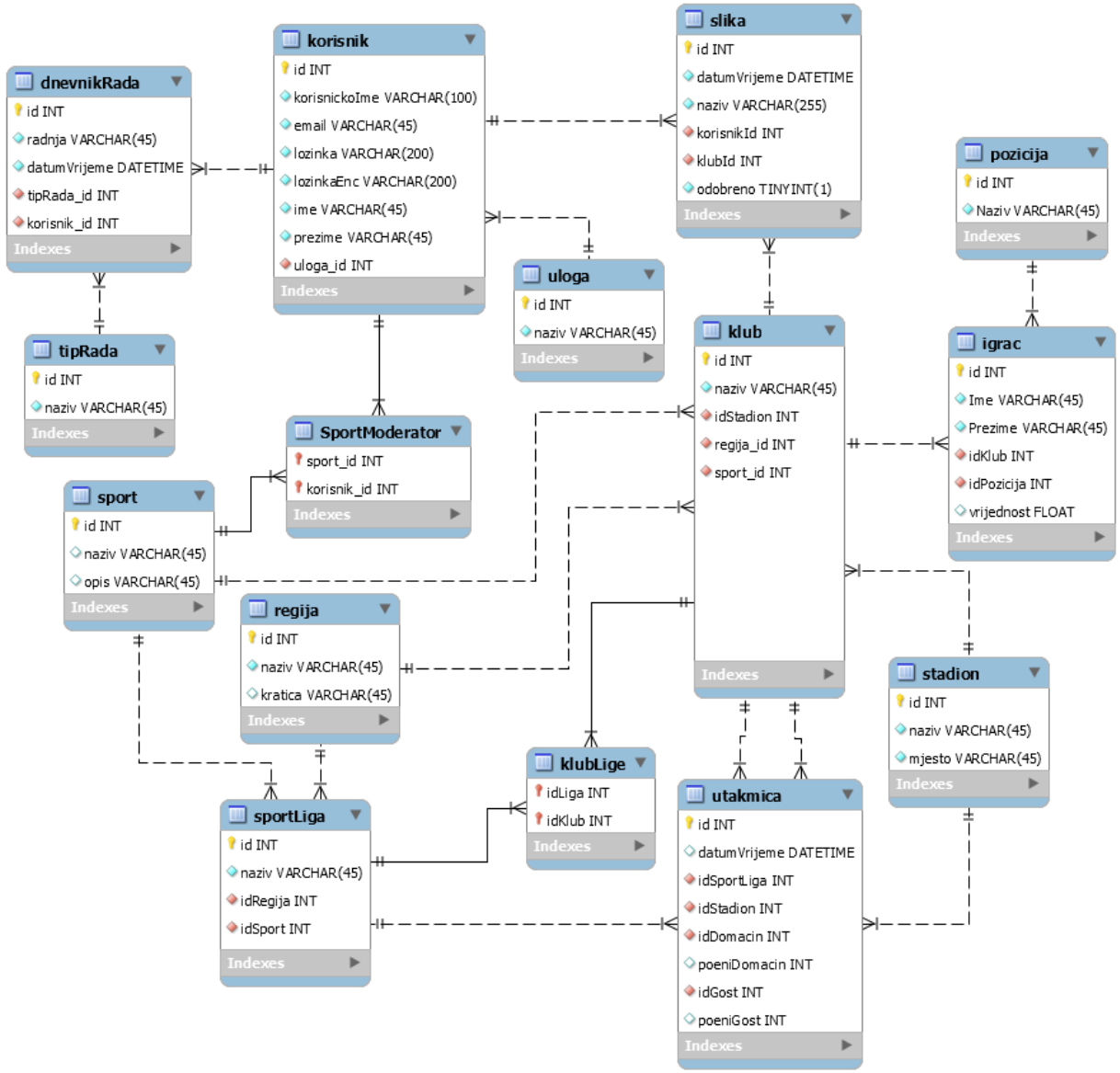
4.1. Opis web aplikacije i korištene tehnologije

Tema praktičnog dijela rada je web aplikacija “Rezlutati”. “Rezlutati”, kao što sam naziv sugerira, je web aplikacija koja korisnicima omogućuje praćenje sportskih događanja. Aplikacija je prije svega namijenjena korisnicima koji žele pratiti sportska događanja za pojedine sportove te pratiti statistiku liga i klubova. Aplikaciji mogu pristupiti svi korisnici odnosno za prikaz glavne funkcionalnosti nije potrebna registracija, no registrirani korisnici dobivaju pravo dodavanja fotografija u galeriju i pristup grafičkim statistikama za pojedine lige. U aplikacija također postoji uloga Moderatora koji ima kontrolu nad sportskim događanjima tj. dodavanje, ažuriranje i po potrebi brisanje događaja, dodavanje novih klubova, igrača,stadiona i pozicija te odobravanje fotografija koje su objavili korisnici. Na vrhu hijerarhije je uloga Administratora koji dodjeljuje moderatore željenim korisnicima, kreira sportove, regije i lige te ima pristup dnevniku aplikacije.

Za izradu dizajna web aplikacije korišten je CSS te u pojedinim dijelovima okvir Bootstrap. Aplikacije je naravno u potpunosti dinamična te je za programiranje na korisničkoj strani korišten Javascript i JQuery te AJAX za uspostavu dinamične i asinkrone komunikacije između korisničke i poslužiteljske strane aplikacije. Na poslužiteljskoj strani korišten je PHP sa mikro okvirima Fat Free i Lumen. Za izradu ERA modela i generiranje gotove SQL skripte korišten je MySQL Workbench. Sve zajedno, upakirano je u gotovo web poslužiteljsko rješenje XAMPP.

4.2. ERA model

Model baze podataka jedan je od prvih i najvažnijih koraka kod izrade složenih web aplikaciija. Baza podataka koristi se za pohranu i dohvaćanje svih podataka koje web aplikacija koristi, a vitalni su za pravilno funkcioniranje aplikacije. Prije same izrade web aplikacije, potrebno je izraditi kvalitetnu bazu podataka koja sadržava sve podatke odnosno atribute u tablicama tj. entitetima te relacije između kreiranih entiteta.



Slika 5: ERA model

Na slici 5 prikazan je model baze podataka za web aplikaciju “Rezlutati”. Model sadrži 15 entiteta od kojih su 13 jaka i 2 slaba entiteta. Za lakše razumijevanje cijele web aplikacije, pojašnjenje ERA modela bit će kombinirano sa ograničenjima koja su zadana programski.

Korisnik registracijom dobiva ulogu koja je po zadanoj vrijednosti korisnik. Administrator kreira sport, regiju i ligu. Kod kreiranja lige, ona dobiva dva vanjska ključa, sport koji se dodjeljuje toj ligi i regiju (npr. Hrvatska) u kojoj se ta liga održava. Na navedenoj slici, vidljiva je veza više-više (M:N) između entiteta korisnik i sport te je tako kreiran slabi entitet sportModerator u kojem je vidljivo koji korisnik ima uloga moderatora nad nekim sportom. Administrator svakom sportu dodjeljuje korisnika koji pritom postaje moderator tog sporta te svaki sport može imati više moderatora. Isto tako svaki korisnik može biti moderator više sportova. Moderator kreira klub,

igrača, poziciju, stadion i utakmicu. Kod kreiranja kluba, moderator dodjeljuje stadion, regiju i sport. Klubu može dodijeliti samo one sportove koje on moderira. Kod kreiranja igrača, moderator dodjeljuje igraču poziciju i samo onaj klub koji je u sportu kojega on moderira. Drugi slabi entitet klubLige, kreiran je vezom više-više između entiteta klub i liga. Moderator svakoj ligi može dodijeliti više klubova i obrnuto. Glavna funkcija aplikacije je sportsko događanje odnosno u ovom modelu baze podataka entitet utakmica. Kod kreiranja utakmice imamo četiri vanjska ključa: liga, stadion te domaćin i gost. Entitet utakmica ima dvije veze prema klubu, jedan za klub koji je domaćin i jedan za klub koji je gost. Kod odabira lige utakmice, moderator može dodijeliti samo one klubove koji su u toj ligi odnosno u entitetu klubLige.

Entitet dnevnikRada služi kako bi Administrator aplikacije mogao dobiti uvid u akcije koje rade korisnici i moderatori. Svakom redu dnevnikaRada pridružujemo vanjski ključ tipRada. Entitet slika koristi se kod prijenosa korisničkih slika. Korisnik sliku pridružuje klubu te moderator odobrava tu sliku kako bi bila javno vidljiva.

4.3. Korisničke uloge u aplikaciji

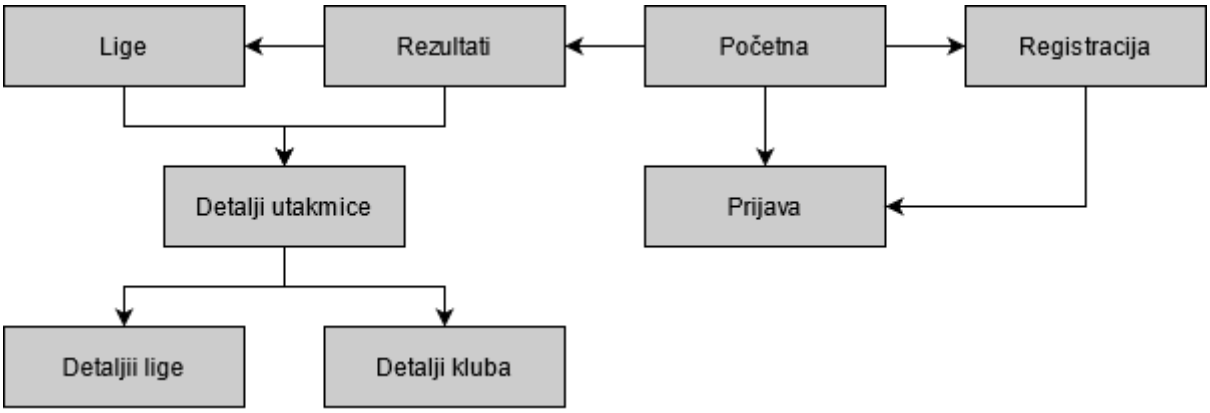
U aplikaciji postoje četiri korisničke uloge:

- Neregistrirani korisnik;
- Registrirani korisnik;
- Moderator;
- Administrator.

U nastavku su prikazani navigacijski dijagrami za svaku korisničku uloge, objašnjenje pojedinih dijelove stranica te slike dijelova aplikacije. Dijagrami će biti prikazani hijerhijskim redom odnosno svaka sljedeća uloga ima pristup svim prethodnim dijelovima aplikacije.

4.3.1. Neregistrirani korisnik

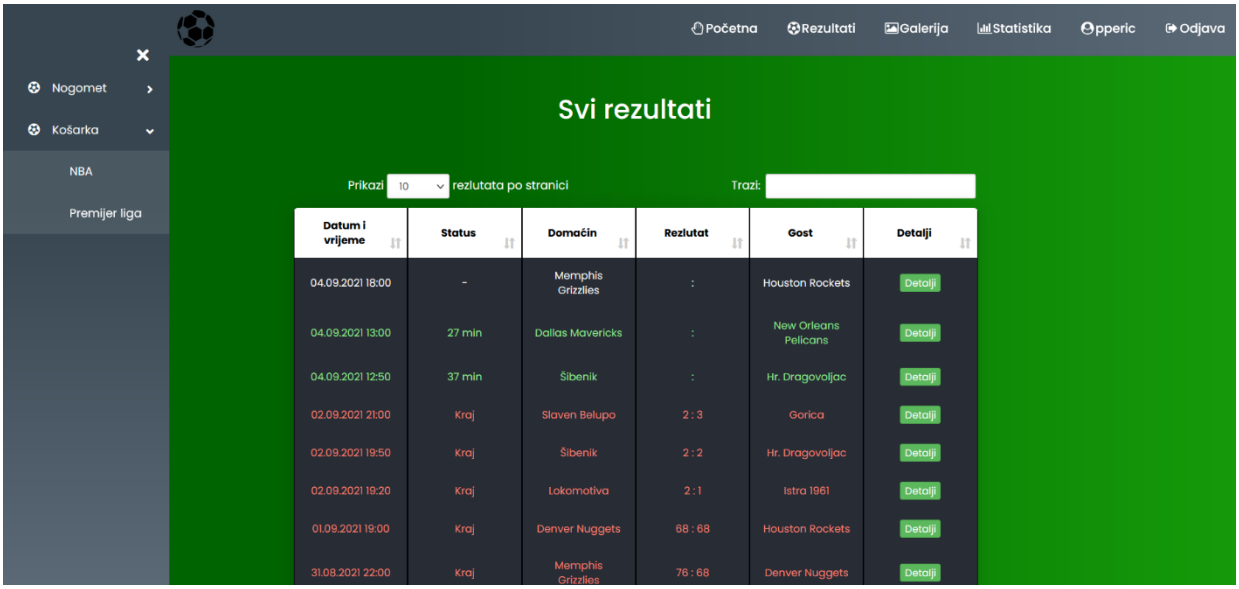
Neregistrirani korisnik je svaki korisnik koji pristupi web aplikaciji, a nema korisnički račun ili nije prijavljen na korisnički račun. Na slici 6 prikazan je navigacijski dijagram za neregistrirane odnosno neprijavljene korisnike.



Slika 6: Navigacijski dijagram za neregistriranog korisnika

Korisnika prilikom pristupanja web aplikaciji dočeka početna stranica gdje se korisnik može preusmjeriti na prijavu, registraciju ili rezultate. Kod registracije, korisnik mora upisati sve podatke koji se traže. Prilikom slanja registracijskog obrasca, provjerava se zauzetost e-mail-a i korisničkog imena, te podudaranje lozinka. Ako bilo koji od tih ulaza ne zadovoljava uvjete, korisnik mora ponovno unijeti tu novu vrijednost. Nakon registracije, korisnika se preusmjerava na prijavu gdje je potrebno upisati korisničko ime i lozinku. Prilikom slanja obrasca provjerava se da li korisničko ime postoji u bazi, te ako postoji lozinka se kriptira i uspoređuje sa zapisanom lozinkom pronađenog reda. Na slikama 7 prikazani su obrasci za registraciju i prijavu.

Slika 7: Obrazac za registraciju i prijavu



Slika 8: Stranica rezultati

Na slici 8 prikazan je izgled stranice rezultati. Prilikom preusmjeravanja na rezultate, korisnika dočeka glavna tablica u kojoj su prikazani svi sportski događaji. Na lijevoj strani prikazan je sklopivi izbornik bočne trake sa svim ponuđenim sportovima te za svaki sport ponuđene su sve lige koje pripadaju tom sportu. Klikom na neku od liga, na glavnoj tablici prikazuju se događaji samo u toj ligi. Tablica sadrži samo najbitnije stavke svakog događaja:

- Datum i vrijeme – početak događaja;
- Status – prikazuje da li je događaj završio (Kraj), je li događaj uopće počeo (-) te ako je počeo prikazuje minutu događaja u rangu od 0-90min;
- Domaćin – naziv kluba domaćina;
- Rezultat – stanje poena između domaćina i gosta;
- Gost – naziv kluba gosta;
- Detalji – dinamičko dugme koje preusmjerava na detalje odabrane utakmice.

Stanje događaja može se prepoznati i prema boji:

- Crvena – događaj završio;
- Zelena – događaj trenutno traje;
- Bijela – događaj nije započeo.

Tablica se također može pretraživati prema domaćinu ili gostu te se može odabrati broj podataka po stranici tablice. Kako bi korisnik došao do više informacija o ligi ili klubu, potrebno je za odabrani događaj pritisnuti dugme Detalji. Na slici 9 prikazana je stranica detalja utakmice.

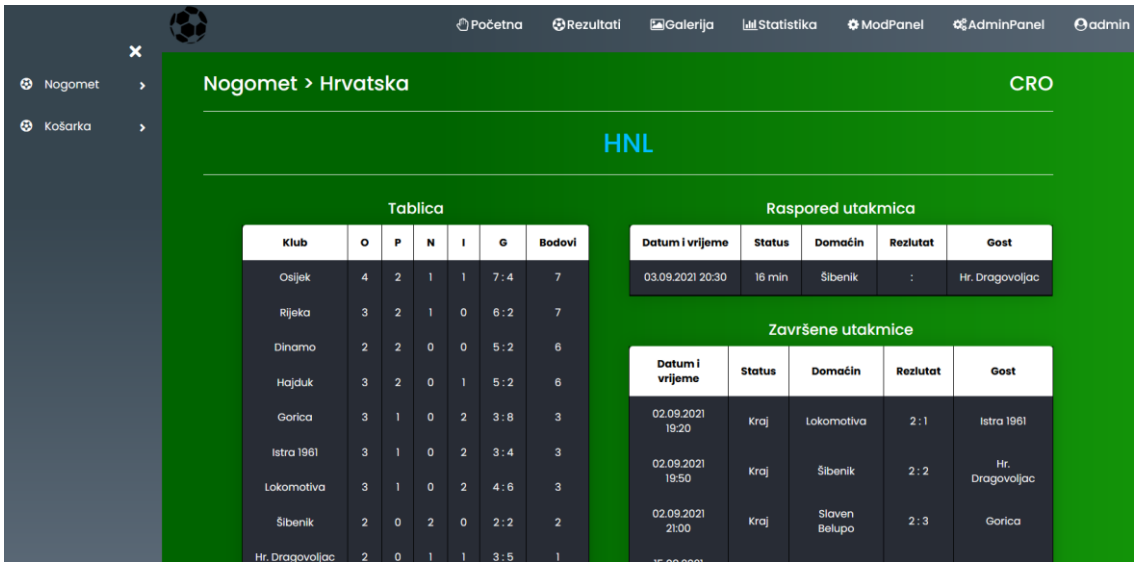
Slika 9: Stranica detalji događaja

Na stranici detalja događaja vidljivo je više informacija za odabrani sportski događaj. Na vrhu stranice vidljiv je naslov sa informacijama sporta, regije i lige. Taj naslov je sidreni element koji pritiskom korisnika preusmjerava na detalje navedene lige. Ispod naslova prezentirane su informacije o vremenu održavanja, mjestu te klubovima i rezultatu sportskog događaja. Također prikazane su i dvije tablice u kojima su ispisani članovi klubova koji sudjeluju u toj utakmici. Iznad tablica postoje još dva sidrena elementa koje korisnika preusmjeravaju na detalje kluba.

Kod detalja lige postoje tri tablice:

- Tablica bodovnog stanja;
- Raspored utakmica – prikazuje nadolazeće utakmice za ligu;
- Završene utakmice – prikazuje sve završene utakmice te lige.

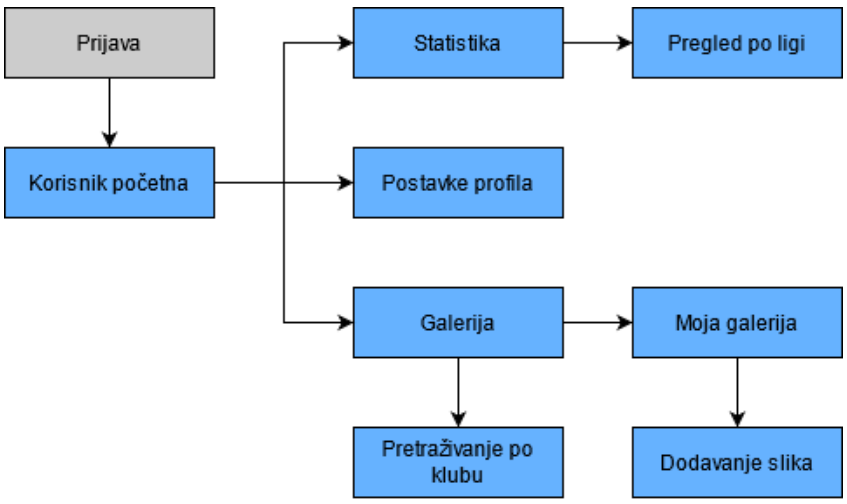
Tablica bodovnog stanja je statistička tablica u koju su, pomoću upita, izvučene sve važne informacije za tu ligu. Tablica u svakom redu sadrži naziv kluba te informacije o broju odigranih, pobijeđenih, neriješenih i izgubljenih susreta, postignute i primljene poene te broj bodova. Cijela tablica poredana je silazno prema broju bodova. Kod stranica detalja utakmice, lige i kluba korisniku je sa lijeve strane prisutan sklopivi izbornik te se u bilo kojem trenutku može preusmjeriti na rezultate druge lige.



Slika 10: Stranica detalji lige

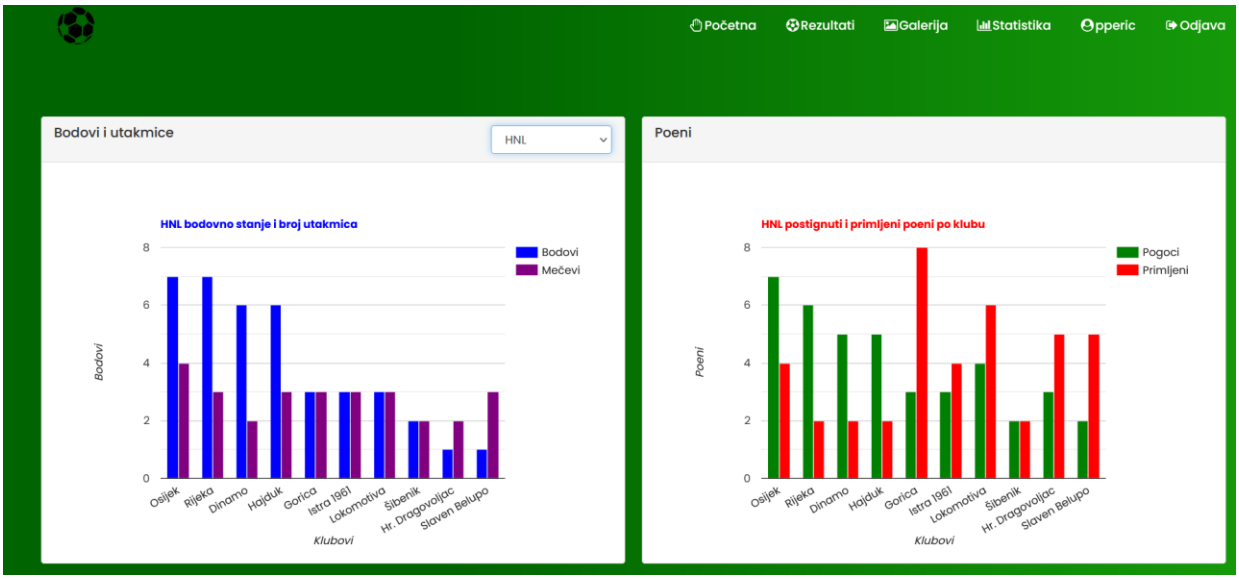
4.3.2. Registrirani korisnik

Registrirani korisnik je korisnik koji ima izrađen korisnički račun. Kako bi dobio pristup dodatnim dijelovima aplikacija, mora biti prijavljen. Za razliku od neprijavljenog korisnika, koji faktički nema nikakvu interakciju sa aplikacijom, registrirani korisnik dobiva pristup grafičkoj statistici liga te mogućnost pregledavanja i prenašanje novih slika u aplikaciju.



Slika 11: Navigacijski dijagram prijavljenog korisnika

Pristupanjem stranici statistike, korisnik odabire željenu ligu pomoću padajućeg izbornika. Nakon odabira željene lige, generiraju se dva grafa koja su izrađena pomoću dodatka Google Charts. Na prvom grafu prikazan je plavom bojom broj bodova i ljubičastom bojom broj utakmica za svaki klub. Na drugom grafu, zelenom i crvenom bojom prikazani su podaci o broju postignutih i primljenih poena.



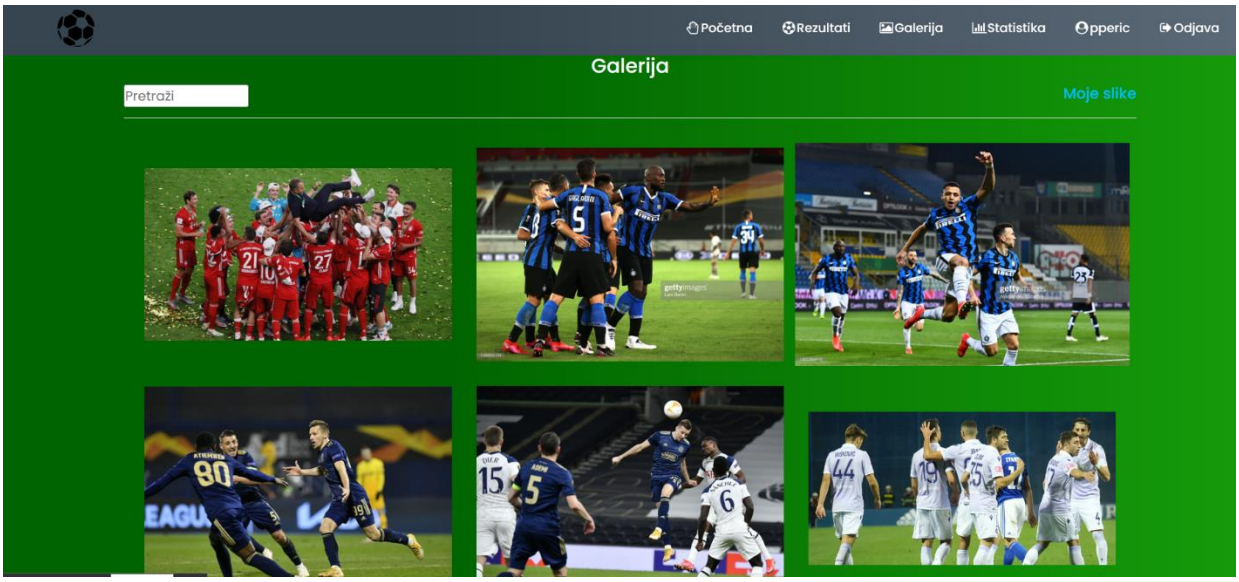
Slika 12: Stranica statistike

Preusmjeravanje na stranicu galerije, korisnik dobiva pristup svim prenesenim korisničkim fotografijama kojima su moderatori odobrili javni prikaz. Na vrhu stranice, implementiran je pretraživač u kojeg korisnik unosi naziv nekog kluba, te se pritom dinamički dohvaćaju pronađene fotografije. Pritiskom na neku fotografiju slika se prikazuje na zaslonu te se galerija može listati i zatvoriti. Za prikaz slika korišten je CSS i JavaScript dodatak Lightbox. Kako bi korisnik pristupio svojoj galeriji i dobio mogućnost prijenosa fotografija, na desnoj strani prikazan je sidreni element koji preusmjerava na korisničku galeriju.

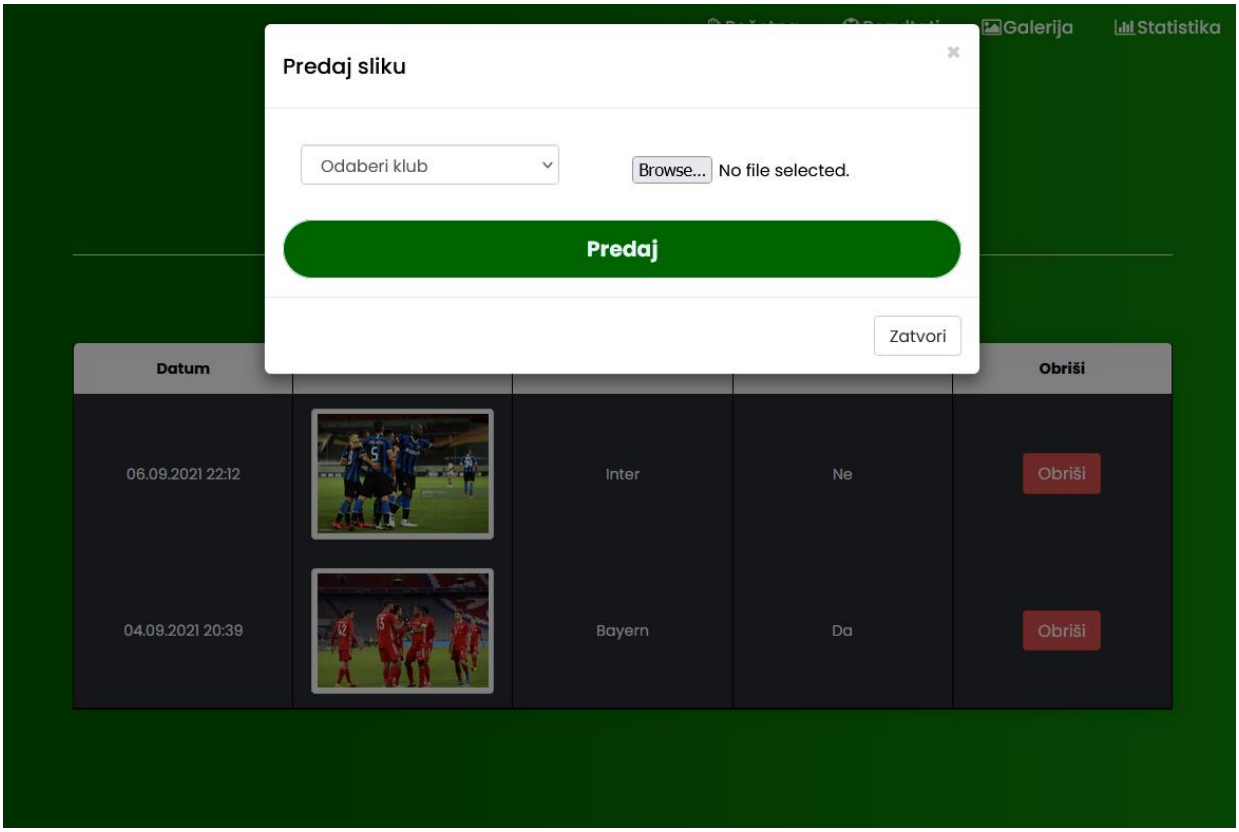
Na stranici korisničke galerija, prikazana je tablica sa sljedećim stupcima:

- Datum – datum i vrijeme prijenosa fotografija;
- Slika;
- Klub;
- Odobreno (Da/Ne);
- Dugme Obriši.

Naravno, da bi postojali podaci u toj tablici, korisnik prvo mora prenijeti neku fotografiju. Klikom na dugme Dodaj sliku. korisniku se prikazuje skočni prozor za dodavanje fotografije. Kako bi uspješno prenio sliku, potrebno je iz padajućeg izbornika odabrati klub i izabrati fotografiju koju namjerava javno prikazati. Nakon što je slika predana, potrebno je odobrenje moderatora kako bi se korisnička slika javno prikazala u galeriji.



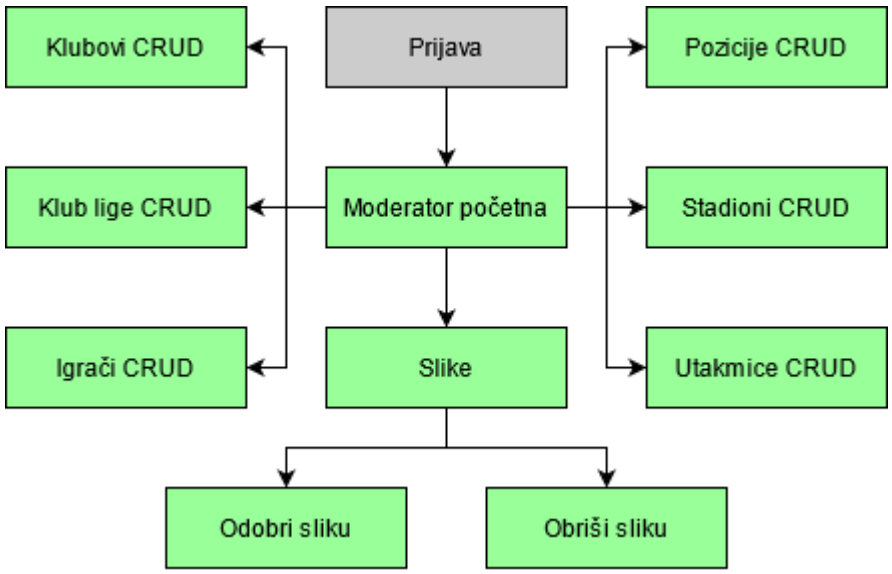
Slika 13: Stranica galerije



Slika 14: Tablica i obrazac za prijenos fotografija

4.3.3. Moderator

Moderator je u aplikaciji zadužen za kreiranje sadržaja odnosno sportskih događaja i odobravanje prenešenih fotografija. Prema navigacijskom dijagramu, vidljivo je da moderator ima dužnost kreiranja, ažuriranja i brisanja klubova, klub liga, igrača, pozicija, stadiona i utakmica te odobravanje slika. Pošto je princip kreiranja i izgled gore navedenih objekata manje-više identičan, prikazane će biti samo stranice za upravljanje utakmicama odnosno događajima i odobravanje fotografija.

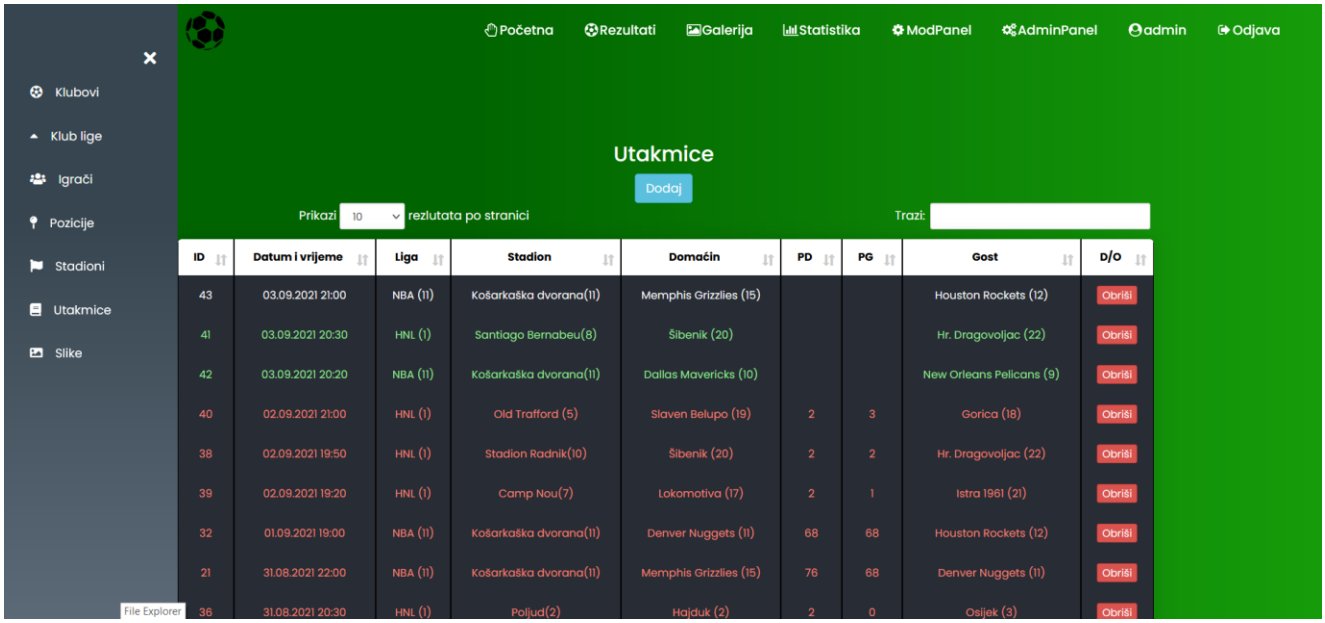


Slika 15: Navigacijski dijagram moderatora

Kad moderator pristupi svojoj upravljačkoj ploči, sa lijeve strane prikazuje se izbornik za upravljanje navedenim opcijama. Klikom na bilo koju od opcija, generira se tablični prikaz sa svojim atributima i podacima.

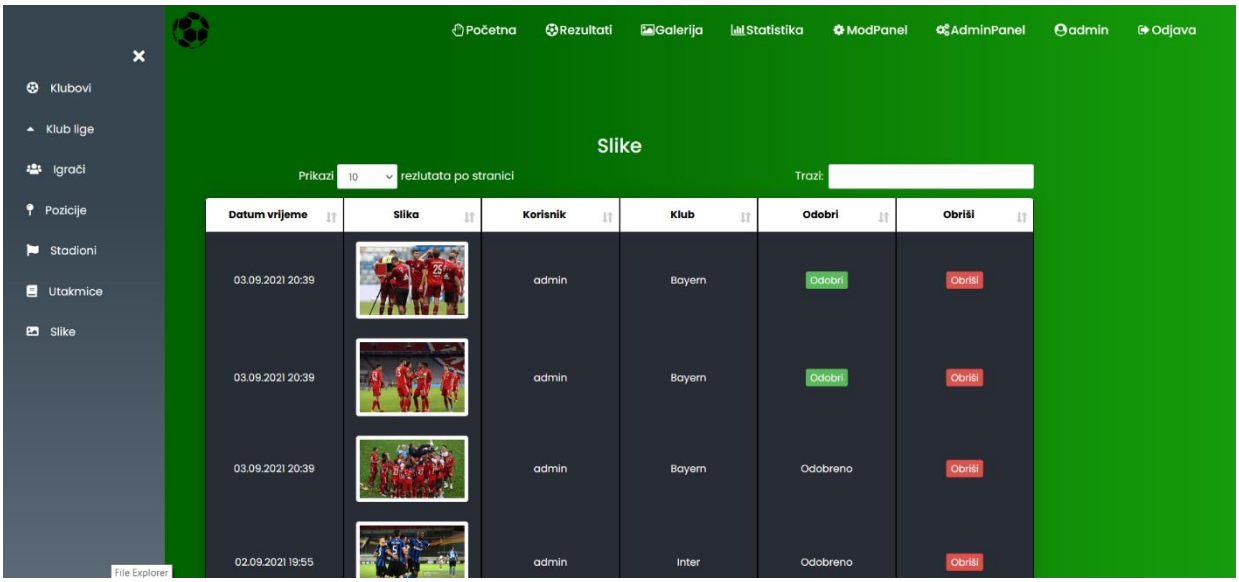
Kako bi kreirali novi događaj odnosno utakmicu potrebno je kliknuti na dugme Dodaj te se na vrhu tablice stvara novi redak. Primarni ključ nije potrebno unositi pošto je u izradi modela baze podataka odabrana opcija automatskog povećavanja. Pritiskom na polje za unos datuma, prikazuje se korisničko sučelje za odabir datum i vremena za koje je korišten dodatak JQuery DateTimePicker. Za stupce tablice Liga, Stadion, Domaćin i Gost kreiran je padajući izbornik pošto su ti stupci odnosno atributi prema modelu baze podataka vezani za druge entitete. Moderatoru su ponuđene samo one lige koje su u sportovima koje on moderira. Odabirom neke lige, otključava se odabir domaćina i gosta, no dostupni su samo oni klubovi koji spadaju u izabranu ligu. Nakon što moderator unese sve potrebne podatke, potrebno je pritisnuti gumb Dodaj kako bi se novo unešeni red spremio u bazu podataka. Svaki red ima svoj gumb za brisanje koji ako se pritisne, prvo prikazuje skočno upozorenje, te nakon prihvaćanja briše odabrani red iz baze podataka. Ažuriranje je provedeno na način da se klikne na željena ćelija

u tablici, promijeni vrijednost te se klikom izvan te ćelije ažurira vrijednost u bazi. Podaci se također mogu sortirati, pretraživati i straničiti.



Slika 16: Stranica za upravljanje utakmicama

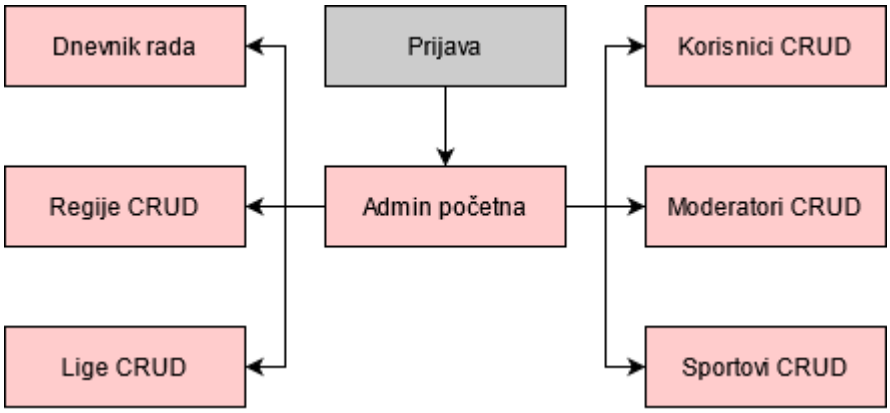
Kao što je prethodno spomenuto, prilikom prijenosa korisničkih slika, moderator mora odobriti sliku. U suprotnom, slika nije vidljiva u javnoj galeriji. Moderator pomoću svoje upravljačke ploče može odobriti ili izbrisati korisničke fotografije.



Slika 17: Stranica za upravljanje slikama

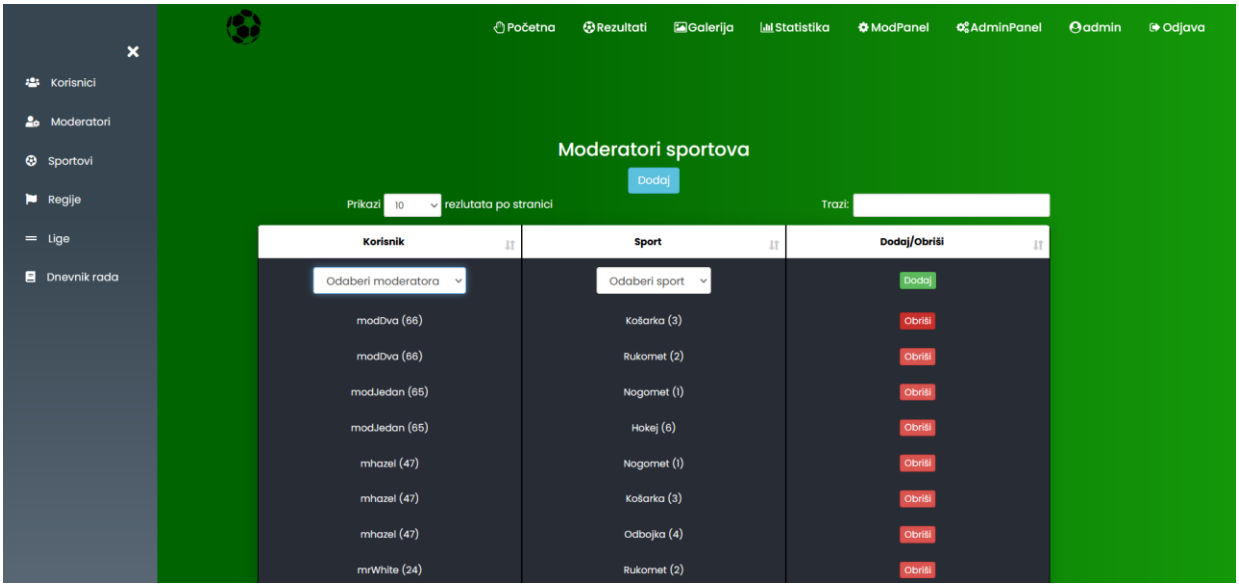
4.3.4. Administrator

Administrator je posljednja uloga koja ima pristup svim dijelovima aplikacije. Administrator može kreirati, brisati i dodavati regije, lige, korisnike, sportove i moderatore. Osim toga admin također ima pristup dnevniku rada aplikacije.



Slika 18: Navigacijski dijagram administratora

Prilikom usmjeravanja na upravljačku ploču administratora, isto kao i kod prethodne uloge, sa lijeve strane nalazi se izbornik prema kojemu se može navigirati na ponuđene dijelove. Kod dodavanja novih podataka u tablice, ako se traži unos vanjskog ključa, na svim tablicama imamo padajuće izbornike sa vrijednostima entiteta kojeg nasljeđuju. Sve tablice mogu se pretraživati i sortirati.



Slika 19: Stranica za upravljanje moderatorima

4.4. Korištenje mikro okvira

Za programiranje na poslužiteljskoj strani, korišteni su mikro okviri Fat Free i Lumen. Fat Free korišten je kao glavni mikro okvir za izradu većine web aplikacije, dok je Lumen korišten za registraciju korisnika.

4.4.1. Fat Free – Inicijalizacija

Kako bi se uspješno inicijalizirali Fat Free okvir, a samim time aplikaciju, potrebna je "index.php" datoteka u korijenu projekta. U index.php datoteci, kreira se instanca glavne klase F3 mikro okvira. Zatim se dohvaćaju i čitaju konfiguracijske datoteke "config.ini" i "routes.ini" koje služe za konfiguracija globalnih varijabli i svih ruta aplikacije. Na koncu pokrećemo mikro okvir sa funkcijom "run()".

```
<?php
require_once("vendor/autoload.php");
$f3 = Base::instance();
$f3->config('config.ini');
$f3->config('routes.ini');
$f3->run();
```

U konfiguracijskoj datoteki "config.ini", predefiniranim globalnim varijablama koje će se koristiti u aplikaciji, dodjeljujemo vrijednosti koje se mogu pronaći u dokumentaciji mikro okvira. U prvoj liniji datoteke potrebno je napisati naziv sekcije ("[globals]") kako bi mikro okvir znao o kakvoj se konfiguracijskoj datoteci radi.

Korištene su sljedeće globalne varijable:

- DEBUG – prikazuje više informacija prilikom neke greške.
- UI – direktorij u kojem se nalaze stranice odnosno pogledi.
- AUTOLOAD – direktorij u kojem se nalaze klase kontrolera kako bi ih mikro okvir automatski priključio u kod.
- MODELS – direkotrij u kojem se nalaze modeli.
- CACHE – potrebnu dodijeliti vrijednost "true" kako bi se mogla koristiti sesija mikro okvira.
- Inicijaliziranje globalnih varijabli za spajanje na bazu podataka.

```
[globals]
DEBUG = 3
UI = app/views/
AUTOLOAD = app/controllers/|Lumen/app/Http/Controllers/d
MODELS = app/models
CACHE = true
devdb = "mysql:host=127.0.0.1;port=3306;dbname=rezlutati"
devdbusername = "root"
devdbpassword = ""
```

Konfiguracijska datoteka “routes.ini” sadrži sve POST i GET zahtjeve za izrađenu aplikaciju. Ukupno je kreirano oko stotinjak ruta za cijelu aplikaciju. Te rute koriste se kod preusmjeravanja i AJAX poziva kako bi aplikacija bila što je više moguće dinamičnija. Kako bi mikro okvir znao da su u ovoj datoteci definirane rute, na vrhu je potrebno upisati naziv sekcije (“[routes]”). Svaka ruta ima naziv metode koja se koristi i rutu te se toj ruti dodjeljuje kontroler i metoda koja će se izvršiti prilikom zahtjevanja te rute.

```
[routes]
GET /=PocetnaController->render
GET /prijava=KorisnikController->render
POST /prijava/login=KorisnikController->prijava
GET /statistika = StatistikaController->render
POST /statistika = StatistikaController->dohvatiPodatkeZaGraf
GET /galerija = GalerijaController->render
POST /galerija/dohvati = GalerijaController->dohvatiSveSlike
GET /rezultati/detalji/@utakmica = RezultatiController->detaljiRender
GET /rezultati/@liga = RezultatiController->ligaRezultatiRender
GET /klub/@klubNaziv/@idKlub = KlubController->klubRender
GET /liga/@ligaNaziv/@idLiga = LigaController->ligaRender
```

4.4.2. Fat Free – Kontroleri

Kontroleri sadrže metode koje se koriste za sve poslužiteljske dijelove aplikacije. Svi kontoleri smješteni su u direktorij “/controllers”. Prilikom definiranje ruta, potrebno je za istu navesti kontroler i metodu koja se pozivam usmjeravanjem na tu rutu. Sve klase kontrolera nasljeđuju glavnu klasu “Controller.php”. U toj klasi inicijalizirane su dvije globalne varijable, jedna za pristupanje metodama mikro okvira i druga za spajanje i pristupanje metodama za rad sa bazom podataka.

```
class Controller{
      public $f3; public $db;
      function __construct(){
              $f3 = Base::instance();
              $this->f3 = $f3;
              $db = new DB\SQL(
                  $f3->get('devdb'),
                  $f3->get('devdbusername'),
                  $f3->get('devdbpassword'),
                  array(\PDO::ATTR_ERRMODE=> \PDO::ERRMODE_EXCEPTION)
              );
              $this->db=$db;
}}
```

U glavnoj klasi imamo i nekoliko metoda. Prva metoda je “beforeroute()”. Ovo je F3 metoda koja se samostalno poziva prije svakog preusmjeravanja. Pomoću ove metode provjeravana je uloga korisnika za pristup dijelovima aplikacije. Također ova metoda je veoma često nadjačana u kontrolerima koji nasljeđuju glavnu klasu, ovisno o tome koja je uloga potrebna za pristup nekom dijelu aplikacije.

```
    function beforeroute(){
        if($this->f3->get('SESSION.korisnik') === null) {
            $this->f3->reroute('/prijava');
            exit;
        }}
```

Sljedeća metoda je “dnevnikRada($upit)”. Ova metoda koristi se za bilježenje prijave i odjave te korisničkog rada sa bazom podataka. Kad god je nekom kontroleru potrebno zapisati podataka u dnevnik rada, pozove se ova metoda te se kao argument šalje upit ili akcija prijave i odjave.

```
    function dnevnikRada($upit){
        $datumVrijeme = date('Y-m-d H:i:s');
        $id = $this->f3->get('SESSION.id');
        $tipRada = 3;
        if($upit == "Prijava" || $upit == "Odjava" ){
            $tipRada = 1;
        }
        $upitt = 'INSERT INTO dnevnikrada() VALUES(?,?,?,?)';
        $this->db-
         >exec($upitt,array($upit,$datumVrijeme,$tipRada,$id));
    }
```

Kako bi bilo jasnije na koji način se prikazuju stranice pomoću kontrolera, prvo će biti objašnjeni pogledi i sustav predložaka pošto se koristi u cijeloj aplikaciji.

4.4.3. Fat Free – Pogledi i predlošci

Svi pogledi odnosno korisnička sučelja smješteni su u direktorij "/views". Pogledi koriste isti predložak koji je napravljen pomoću F3 sustava za predloške.

```
<include href="Predlozak/static_head.php"/>
<include href="Predlozak/navigacija.php"/>
<include href ="{{@stranica}}"/>
```

U "static_head.php" nalaze se sve ugrađene skripte i stilovi koji se koriste u aplikaciji. Datoteka "navigacija.php" sadrži glavnu navigaciju web aplikacija. Za navigaciju su korišteni uvjetni segmenti predložaka. Za one dijelove navigacije koji su vidljivi samo prijavljenim korisnicima odnosno moderatoru ili administratoru, dohvaća se uloga korisnika te se provjerava vrijednost te varijable. Ukoliko je uvjet zadovoljen prikazuje se taj dio navigacije. Nadalje je prikazan samo jedna dio navigacije u kojem se koriste uvjetni segmenti:

```
 <check if="{{@SESSION.uloga >= 1}}">
      <false>
           <a href="/prijava">Prijava</a>
       </false>
       <true>
            <a href="/galerija">Galerija</a>
           <a href="/statistika">Statistika</a>
            <check if="{{@SESSION.uloga >= 2 }}">
                <true>
                     <a href="/moderator">ModPanel</a>
                </true>
             </check>
```

Globalnu varijablu "stranica" koristimo kod svih kontolera. U nju se sprema lokacija pogleda koji će biti prikazan korisniku. Zatim je potrebno kreirati novu instancu predloška i prikazati taj predložak pomoću metode "render()";

```
    function render()
    {
        $this->f3->set('title',   'Moderator');   //title   je   naslov
stranice
        $template = new Template();
        $this->f3->set('stranica', 'Moderator/pocetna.php');
        echo $template->render('Predlozak/predlozak.php');}
```

Ovaj princip koristi se tokom cijele web aplikacije te je veoma jednostavan za razumijeti i primjeniti.

4.4.4. Fat Free – Ponavljajući segmenti predložaka

Ponavljajući segmenti veoma su korisna funkcionalnost kod sustava predložaka. Bez direktnog korištenja PHP-a u HTML dijelovima, mogu se dohvatiti podaci za prikaz korisniku. Ti isti podaci mogu se prikazati u tablicam, navigacijama ili nekim drugim dijelovima. Većina tablica u aplikaciji prikazana je pomoću AJAX-a. U cilju iskorištavanja funkcionalnosti predložaka, tablice u stranicama detalja utakmice, lige i kluba, popunjene su podacima pomoću ponavljajućih segmenata. Kao primjer prikazana će biti funkcija koja dohvaća momčad kod detalja utakmice:

```
    function DohvatiEkipuDomacina($id){
        $domacinEkipaArr = array();
        $upitDomacin="SELECT
igrac.Ime,igrac.id,igrac.Prezime,pozicija.Naziv
               FROM igrac, pozicija, utakmica, klub
               WHERE utakmica.id = ? AND utakmica.idDomacin = klub.id
               AND klub.id = igrac.idKlub AND igrac.idPozicija =
pozicija.id";
        $rezultat = $this->db->exec($upitDomacin, array($id));
        foreach ($rezultat as $red){
            $domacinEkipaArr[$red["id"]][] = $red["Ime"];
            $domacinEkipaArr[$red["id"]][] = $red["Prezime"];
            $domacinEkipaArr[$red["id"]][] = $red["Naziv"];
        }
        $this->f3->set('domacinEkipa', $domacinEkipaArr);}
```

U ovoj funkciji izrađen je upit koji dohvaća igrače i nazive njihovim pozicija za klub koji je domaćin odabrane utakmice. U cijeloj aplikaciji korišteni su parametrizirani upiti koji daju sigurnost bazi podataka. Nakon što je napisan upit, pomoću funkcije “exec()” izvršavamo taj upit te se rezlutat sprema u varijablu “$rezultat”. Nakon toga se iteriramo kroz sve redove rezlutata i spremamo željene podatke u dvodimenzionalni niz sa ključem “id”. Dobiveni niz spremamo u globalnu varijablu “domacinEkipa” pomoću F3 funkcije “set()”. Sad kada su dohvaćeni i spremljeni svi igrači nekog domaćina i njihove pozicije, potrebno je prikazati te podatke pomoću ponavljajućih segmenata:

```
 <repeat group="{{@domacinEkipa}}" key="{{@id}}" value="{{@igrac}}">
     <tr>
          <repeat group="{{@igrac}}" value="{{@igracInfo}}">
               <td>{{@igracInfo}}</td>
            </repeat></tr>
```

```
 </repeat>
```

U atribut “group” potrebno je staviti naziv globalne varijable u koju je spremljen niz sa podacima. Pomoću ključa “id” pristupamo pojedinim vrijednostima. Kreira se jedan red tablice, te se u drugom ugnježdenom segmentu, za svaki podatak o igraču kreira jedna ćelija u tablici i upisuje se taj podatak. Na kraju zatvaramo taj redak te se vraća u prvi ponavljajući segment po sljedeči niz podataka o igraču.

Pomoću istog principa, kreiranA je i dinamička navigacija u kojoj su ispisani svi sportovi te lige koje pripadaju tim sportovima.

```
    function DohvatiLigeISportove(){
        $rezlutatArr = array();
        $upitSport = "SELECT naziv FROM sport";
        $sportovi   =   $this->db->exec($upitSport);   //dohvacam   sve
sportove

        foreach ($sportovi as $sport){ //iteriram se kroz sportove
            $upitSportLige  =  "SELECT  sportliga.naziv  FROM  sportliga,
sport
            WHERE sportliga.idSport = sport.id AND sport.naziv = ?";
            $sportLige        =        $this->db->exec($upitSportLige,
$sport['naziv']);
            foreach ($sportLige as $liga){ //iteriram se kroz lige
                $rezlutatArr[$sport['naziv']][] = $liga['naziv'];
            }
        }
        $this->f3->set('sportLige', $rezlutatArr);
    } //dohvaca sportove i lige koje su u pojedinom sportu za sidebar
menu


<repeat group="{{@sportLige}}" key="{{@sport}}" value="{{@liga}}">
            <div class="item">
                <a class="sub-btn">{{@sport}}></a>
                <div class="sub-menu" id="prikazi">
                    <repeat group="{{@liga}}" value="{{@ligaIspis}}">
                        <a class="sub-item">{{@ligaIspis}}</a>
                    </repeat>
                </div>
            </div>
 </repeat>
```

4.4.5. Fat Free – Ostale funkcionalnosti

Osim navedenih, korištene su još neke funkcionalnosti. Korišten je veoma jednostavan F3 zapisnik podataka (eng. logger). Metoda za zapisnik “loggerFatFree($radnja)” kreirana je u glavnom kontroleru. Kada je potrebno zapisati neki podataka u zapisnik, pozove se ova metode te se radnja šalje kao argument. Prilikom poziva ove metode, kreira se objekt i datoteka na navedenoj lokaciji. Nakon toga sve što je potrebno je zapisati željenu radnju. Ovaj zapisnik koristi se za zapisivanje korisnika koji dodaju i brišu svoje fotografije u galeriju. Također zapisnik samostalno zapisuje datum i vrijeme te IP adresu.

```
function loggerFatFree($radnja){
    $logger = new Log('files/slike.log');
    $logger->write($radnja);
}
```

4.4.6. Lumen

Lumen je u aplikaciji korišten za registraciju novih korisnika. Prvo je potrebno izraditi novu “.env” datoteku u koju se zapisuju podaci za spajanje na bazu podataka.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rezlutati
DB_USERNAME=root
DB_PASSWORD=
```

U Lumenu definirane su samo dvije rute. Jedan ruta koristi se za dohvaćanje korisničkog sučelja, a druga se koristi za slanje obrasca registracija kada korisnik unese sve podatke.

```
$router->get('/registracija', 'RegistracijaController@render');
$router->post('registriraj', 'RegistracijaController@registriraj');
```

Sljedeće je kreiran kontroler u kojem imamo dvije metode prema prethodno definiranim rutama. Metoda “render()” koristi se za prikaz korisničkog sučelja. Također je iskorištena funkcionalnost priručne memorije u koju spremamo sve vrijednosti koje korisnik unese u registracijski obrazac. Ukoliko korisnik unesene neku krivu vrijednost, kreiramo poruku koju spremamo u priručnu memoriju za prikaz korisniku. Pomoću funkcije “get()” u pogledu dohvaćaju se podaci iz priručne memorije.

```
  public function render()
    {
        if(Cache::has('poruka') && !(Cache::has('refresh'))){
            Cache::set('refresh', 1);
        }
        else{
            Cache::flush();
        }
        return view('registracija');}
```

U metodi “registriraj(Request $req)”, dohvaćaju se sve vrijednosti koje je korisnik upisao tokom registracije pomoću funkcije “input()”. Te podaci se provjeravaju pošto korisničko ime i email moraju biti jedinstveni za svakog korisnika. Ako neki podataka ne zadovoljava uvjete, pomoću priručne memorije kreiramo poruku te se korisnik ponovno preusmjerava na stranicu registracije gdje se prikazuje poruke greške. Nakon što su svi podaci zadovoljeni, u bazu podataka sprema se novi korisnik i preusmjerava na stranicu prijave.

```
public function registriraj(Request $req)
{
      Cache::flush();
      $email = $req->input('email');
      $lozinka = $req->input('lozinka');
      $potvrdi_lozinku = $req->input('potvrdi_lozinku');
      $ime = $req->input('ime');
      $prezime= $req->input('prezime');
      $korisnickoIme = $req->input('korisnickoIme');
      $zauzetoKorisnickoIme = DB::select("SELECT * FROM korisnik WHERE
      korisnickoIme = ?", [$korisnickoIme]);
      if($zauzetoKorisnickoIme){
            Cache::put('poruka','Korisničko ime je zauzeto!');
            redirect()-
>to('http://localhost/Lumen/public/registracija');
            exit();
      }
      $lozinkaEnc = Hash::make($lozinka); //hashiranje lozinke
      DB::insert('insert into korisnik (korisnickoIme, email, lozinka,
      lozinkaEnc, ime, prezime, uloga_id)' .
      'values(?,?,?,?,?,?,?)',   [$korisnickoIme,   $email,   $lozinka,
      $lozinkaEnc, $ime, $prezime, 1]);
      Cache::flush();
      return redirect()->to('http://localhost/prijava');}
```

4.4.7. Rad aplikacije sa dva mikro okvira

Kako bi aplikacija uspješno radila sa dva mikro okvira, potrebno je bilo doći do kreativnog rješenja pošto nisu pronađeni nikakvi primjeri gdje se koristi više okvira u istom projektu. Problem je riješen tako da u glavnom projektu, ustvari imamo dva zasebna mikro okvira koji zasebno rade te imaju svoje “index.php” i “.htaccess” datoteke. U korijenu projektnog direktorija, nalaze se te dvije datoteke koje koristi Fat Free. U direktoriju “/Lumen” smještene su sve datoteke i direktoriji koji su potrebni Lumen-u da bi pravilno radio kao i pogledi, kontroleri te rute koje se koriste za registraciju. Preusmjeravanje između mikro okvira provedeno je pomoću apsolutnih putanja.

```
<a href="/prijava">Prijava</a>
<a href="http://localhost/Lumen/public/registracija">Registracija</a>
```

Prikazane su dvije putanje koje se koriste na početnoj stranici aplikacije. Putanja “/prijava” vodi korisnika do prijave koja je izvedena pomoću Fat Free mikro okvira. Druga putanja vodi korisnika do registracijskog obrasca. Kao što je prikazano, registracija se provodi pomoću Lumena. Da bi web server uspješno došao do samog Lumena koji onda prema ruti “/registracija” poziva potrebni kontroler i metode, potrebno je bilo koristiti apsolutne putanje.

5. Prednosti, nedostaci i problemi u radu sa mikro okvirima

Korištenje mikro okvira sa sobom nosi mnoge prednosti, ali i nedostatke. Nadalje će se opisati kakve sve prednosti i nedostatke donose mikro okviri i koji se problemi javljaju kombiniranje više mikro okvira u jedan projekt.

5.1. Prednosti korištenja mikro okvira

Korištenje mikro okvira u razvoju aplikacije daje čvrste argumente zašto se danas mnogo IT poduzeća oslanja na okvire. Iako ovaj rad ne koristi prave okvire nego mikro okvire, korelacija između njih je prethodno u radu prikazana.

Fat free dolazi u samo jednoj dolazi u samo jednom direktoriju. Kao najjači adut mikro okvira definitivno valja spomenuti sustav za preusmjeravanje (eng. Routing engine). Korištenjem autoloadera, pogleda i zasebno definiranih ruta, znatno olakšava održavanje i snalaženje u projektu. Odvajanje poslovne logike od korisničkog sučelja daje projektu bolju čitljivost i lakše implementiranje novih funkcionalnosti. Fat Free mikro okvira ne staje na put korisnik sa pravilima strukturiranja već ima potpunu slobodu da strukturira i izrađuje aplikaciju po svojoj volji.

Sustav predložaka sa kombinacijom globalnih varijabli znatno olakšava dohvaćanje dinamičkih dijelova web stranice. Umjesto pisanja PHP skripti unutar HTML oznaka, F3 nudi jednostavno dohvaćanje varijabli, uvjetovane segmente koji su vrlo korisni kod ispisa gdje neki zahtjev mora biti zadovoljen i naravno ponavljajući segmenti za ispis nizova sa podacima. Iako na kraju okvir sve to prevodi u PHP kod, korištenjem sustava predložaka kod je puno jednostavniji i pregledniji.

Lumen ima veoma jednostavan i prihvatljivi način dohvaćanja POST podataka koje korisnik šalje. Također korištenje i čišćenje predmemorije veoma je jednostavno i inutitivno uz korištenje gotovih metoda. Lumen dijeli jezgru sa Laravelom tako da je zajednica poprilično velika i mogu se brzo pronaći rješenja za probleme sa kojima se suočava.

Korištenje mikro okvira daje samom kodu drugačiji izgled, jednostavnost i čitljivost. Valja spomenuti da sa uspješnim savladavanjem mikro okvira, ulaz u svijet pravih okvira puno je jednostavniji i lakši za prebroditi.

5.2. Nedostaci korištenja mikro okvira

Mikro okviri sa sobom nose više prednosti nego nedostataka. Ovisno o odabiru mikro okvira, nedostaci mogu raznih vrsta. Najčešći nedostaci mikro okvira su loše dokumentacije, male zajednice te nezavršene funkcionalnosti, koje se ponekad onda moraju izbjegavati.

Fat Free mikro okvir nema mnogo nedostataka. Iako je dokumentacija veoma dobra i intuitivna, nedostatak primjera i tutorijala produljuje vrijeme koje je potrebno da se neka funkcionalnost okvira savlada i implementira. Također zajednica je poprilično mala, tako da traženje rješenja nekog problema na webu najčešće ne urodi plodom. Što se tiče funkcionalnosti, objektno relacijskom maperu fale neke ključne funkcionalnosti za dohvaćanje podataka iz baze te je malo primjera kako koristiti ovu funkcionalnost. Iz tog razloga upiti prema bazi, pisani su ručno bez korištenja mapera, odnosno podaci su se spremali u nizove te im je pristupano pomoću “foreach()” petlje.

Nedostatak lumen okvira je taj što instalacijom dobivamo mnogo direktorija koji zajedno sadrže oko 20MB. Sloboda strukturiranja projekta, Lumenu nije jača strana. Naime, Lumen dolazi sa predefiniranim direktorijima u kojim se nalaze kontroleri, pogledi i rute. Projekt se dakako može strukturirati po želji korisnika, no potrebno je tražiti i promijeniti linije koda u kojem se definira autoloader, lokacija pogleda, ruta i modela. U ovom slučaju vlastito strukturiranje dovelo je samo do većih komplikacija te je radi toga zadržana struktura koja dolazi sa Lumenom.

5.3. Problemi u radu sa više mikro okvira

Korištenje mikro okvira nagrađuje programera bržim razvojem, boljim performansama aplikacije i boljom strukturom i preglednošću cijelog projekta. Ako se želi koristiti dva ili više mikro okvira u jednom projektu, dolazi do poprilično velikih problema.

Uvjerljivo najveći problem pri pokušajima korištenja više mikro okvira u projektu, bio je nedostatak bilo kakvih primjera. Pretraživanjem popularnih web stranica kao što je Stack Overflow, nije postojao niti jedan primjer korištenja više mikro okvira u jednom projektu. Zašto je rijetkost koristiti više mikro okvira u jednom projektu? Mikro okviri dizajnirani su za što bolje performanse aplikacije, brzinu i jednostavnost razvijanja projekta. Kod korištenja više različitih mikro okvira u jednom projektu, njihove prednosti se na neki način poništavaju.

Kod instaliranja mikro okvira u projekt, taj mikro okvir postaje glavni sustav cijele aplikacije. Svi korisnički zahtjevi prema web serveru usmjeravaju se na sustav preusmjeravanja mikro okvira. Različiti mikro okviri, na prvi pogled izgledaju kao da koriste sličnu sintaksu, no jezgre mikro okvira nemaju isti sustav odnosno isti način obrade tih zahtjeva, te zbog toga dolazi do problema. Nije jednostavno konfigurirati .htaccess datoteku da različite zahtjeve usmjerava

na različite mikro okvire, te zbog nedostatka informacije i primjera takvog rada, ovaj princip rada nije niti implementiran.

Implementacija više mikro okvira u isti projekt i nema baš previše smisla. Osim što dolazi do očitih problema i kolizije između njih. Bolja solucija je odabir jednog mikro okvira te po potrebi priključivanje dodatnih alata i značajka ili korištenje pravog okvira punog spektra.

6. Zaključak

U današnje vrijeme, velika većina svih poslovanja temelji se na web aplikacijama. Korisnici su sve skloniji korištenju online poslovanja za sve aspekte privatnog i poslovnog života. Veliki razlozi tomu su ušteda vremena, novca i obavljanje raznih obaveza iz udobnosti doma. Jednake prednosti dobivaju i korporacije koje žele što je više moguće dijelova svojeg poslovanja prebaciti na internet. Uzet ćemo za primjer jednu od vodećih svjetskih tvrtki u potrošačkoj elektronici. Xiaomi je relativno mlada tvrtka koja je svojom strategijom online poslovanja bacila tehnološke divove na koljena.

Rapidan razvoj i velika potražnja online poslovanja, stavila je pritisak i na IT industriju. Kako bi zadovoljila potrebe tržišta, IT industrija se oslanja na korištenje gotovih temelja web aplikacija. Umjesto “ponovnog otkrivanja kotača” (eng. Reinventing the wheel) industrija se pouzda u okvire i mikro okvire, kako bi fokus realizacije nekog projekta bio na poslovnoj logici i zahtjevima klijenata. Slijedom toga, poslodavci danas nerijetko traže znanje i iskustvo u radu sa nekim okvirom ne samo u grani web programiranja već i u ostalim granama IT industrije.

Mikro okviri pružaju osnovne alate korisniku kako bi se razvoj web aplikacije pojednostavio. Potrebno je proći kroz nekoliko primjera da se dobije razumijevanje na koji način funkcioniraju kontroleri, pogledi i rute. No, razumijevanjem i korištenjem takvih okvira, projekt postaje pregledniji, jednostavniji za održavati i po lakši za nadogradnju. Korištenje baza podataka je veoma jednostavno. Uz samo nekoliko linija koda postići se može veoma mnogo na vrlo jednostavan način. Naime, ako se koristi više mikro okvira u istoj aplikacija, dolazi do raznih problema. Potrebno je konfigurirati poslužitelj tako da neke zahtjeve šalje jednom, a neke drugom mikro okviru. Za iskusne inženjere to možda i nije toliki problem pošto imaju iskustva i znanja za realizaciju takvih pothvata. Problem se javlja kad se neiskusan programer ohrabri na ovakve akcije. Teško je pronaći bilo kakve primjere ili informacije kako koristiti više okvira u istom projektu te je to ustvari i najveći problem, nedostatak praktične literature. No, uzevši u obzir sve što jedan mikro okvir ponudi korisniku, može se zaključiti da bi razvoj aplikacije bez korištenja mikro okvira vremenski trajao puno duže.

Neiskusnim i mladim programerima, prvi susret sa okvirima i mikro okvira budi razne emocije. Pothat korištenja okvira, od samog širokog spektra izbora, instalacije, učenja i na koncu primjene, definitivno nije jednostavan. No, korištenjem dokumentacije i primjera te ulaganjem truda, vremena i strpljenja, mikro okviri se mogu uspješno savladati i primijeniti u izradi vlastite web aplikacije.

Popis literature

[1] K. Tatroe, P. MacIntyre i R. Lerdorf, *Programiranje PHP*, 3. Izd., Zagreb: Dobar Plan, 2009.

[2] W3Techs, “Historical trends in usage statistics of server-side programming languages for websites”, 2021.

https://w3techs.com/technologies/history_overview/programming_language (pristupljeno 29.06.2021).

[3] PHP, “What can PHP do?”,https://www.php.net/manual/en/intro-whatcando.php (pristupljeno 29.06.2021).

[4] Kinsta, “The Most Popular PHP Frameworks to Use in 2021” https://kinsta.com/blog/php-frameworks/ (pristupljeno 30.06.2021).

[5] Blogspot, “Lets do PHP” https://letsdophp.blogspot.com/p/architecture-diagram-of-php-based-web.html (pristupljeno 30.06.2021).

[6] J. Samra, *Comparing Performance of Plain PHP and Four of Its Popular Frameworks* [Projektna disertacija]. Sveučilište Linnaeus, Švedska 2015 https://www.diva-portal.org/smash/get/diva2:846121/FULLTEXT01.pdf (pristupljeno 10.7.2021.)

[7] D. Shirey, “Top 10 PHP Security Vulnerabilities”, 2012. https://www.sitepoint.com/top-10-php-security-vulnerabilities/ (pristupljeno 10.7.2021)

[8] S. M. Srinivasan I R. S. Sangwan, “Web App Security: A Comparison and Categorization of Testing Frameworks”, *IEEE Software*, 2017, 34.1: 99-102. DOI: 10.1109/ms.2017.21

[9] R. E. Johnson “Frameworks = (components + patterns)”, *Communications of the ACM*, 1997, 40.10: 39-42. DOI: 10.1145/262793.262799

[10] N. Prokofyeva, V. Boltunova “Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems”, *Procedia Computer Science*, 2017, 104: 51-56. DOI: 10.1016/j.procs.2017.01.059

[11] Mindfire Solutions “Advantages and Disadvantages of PHP Frameworks”, 2018. https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-php-frameworks-c046d50754e5 (pristupljeno 17.7.2021.)

[12] M. Stauffer, *Laravel Up & Running A Framework for Building Modern PHP Apps*, 2. Izd, Sebastopol: O'Reilly, 2019.

[13] C. Rada, *Learning Phalcon PHP*, 1. Izd, Birmingham: Packt Publishing, 2015.

[14] M. Laaziri, K. Benmoussa, S. Khoulji, i M.L. Kerkeb "A Comparative study of PHP Frameworks performance", *Procedia Manufacturing*, 2019, 32: 864-871.

[15] O. Habib ,"PHP Microframework vs. Full Stack Framework", 2015. https://www.appdynamics.com/blog/engineering/php-microframework-vs-full-stack-framework/ (pristupljeno 02.08.2021.)

[16] C. Russel, *PHP Development Tool Essentials*, 1. Izd, Jacksonville: Apress, 2016.

[17] TechnoLush "Micro Vs Full-Stack Frameworks", 2019. https://www.technolush.com/blog/micro-vs-full-stack-frameworks (pristupljeno 05.08.2021.)

[18] Composer "Composer Documentation" https://getcomposer.org/doc/ (pristupljeno 10.08.2021.)

[19] XAMPP "XAMPP About" https://www.apachefriends.org/about.html (pristupljeno 10.08.2021.)

[20] Fat-Free Framework "Fat Free User Guide", https://fatfreeframework.com/3.7/user-guide (pristupljeno 11.08.2021.)

[21] Lumen "Lumen Docs", https://lumen.laravel.com/docs/8.x (pristupljeno 12.08.2021.)

[22] Flight "Learn Flight" https://flightphp.com/learn (pristupljeno 13.08.2021.)

[23] Limonade "Limonade README" https://github.com/sofadesign/limonade/blob/master/README.mkd (pristupljeno 16.08.2021.)

[24] PHPixie "PHPixie Documentation" https://phpixie.com/docs.html (pristupljeno 01.09.2021.)

[25] A. Amar "What is PHPixie?", DiscoverSDK, 2016. http://www.discoversdk.com/blog/what-is-phpixie (pristupljeno 01.09.2021.)

Popis slika

Slika 1: Upotreba programskih jezika na strani poslužitelja (Izvor: [2]) 3
Slika 2: Dijagram arhitekture web aplikacija temeljenih na PHP-u (Izvor: [5]) 4
Slika 3: Izgled MVC strukture (Izvor: [12]) 6
Slika 4: Struktura Lumen mikro okvira 15
Slika 5: ERA model 23
Slika 6: Navigacijski dijagram za neregistriranog korisnika 25
Slika 7: Obrazac za registraciju i prijavu 25
Slika 8: Stranica rezultati 26
Slika 9: Stranica detalji događaja 27
Slika 10: Stranica detalji lige 28
Slika 11: Navigacijski dijagram prijavljenog korisnika 28
Slika 12: Stranica statistike 29
Slika 13: Stranica galerije 30
Slika 14: Tablica i obrazac za prijenos fotografija 30
Slika 15: Navigacijski dijagram moderatora 31
Slika 16: Stranica za upravljanje utakmicama 32
Slika 17: Stranica za upravljanje slikama 32
Slika 18: Navigacijski dijagram administratora 33
Slika 19: Stranica za upravljanje moderatorima 33

Popis tablica

Tablica 1: Razlika okvira i mikro okvira [17] 8