Aplikacija za organizaciju i provedbu natjecanja

Maretić, Antonio

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ANTONIO MARETIĆ

APLIKACIJA ZA ORGANIZACIJU I PROVEDBU NATJECANJA

Diplomski rad

Pula, lipanj 2018.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ANTONIO MARETIĆ

APLIKACIJA ZA ORGANIZACIJU I PROVEDBU NATJECANJA

Diplomski rad

JMBAG: 30-INFO-D, redoviti student
Studijski smjer: Informatika

Predmet: Napredni algoritmi i strukture podataka
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, lipanj, 2018.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za magistra _____ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student _____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom_____

_____

_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis _____

Sadržaj

1. Uvod

Pojavom progresivnih web aplikacija (eng. progressive web application) razvila su se nova, brža i pouzdanija razvojna okruženja koja na efikasan, jednostavan i brz način omogućuju izradu takvih aplikacija. One imaju prednost nad ostalima iz razloga zato što se razvija samo jedan projekt (jedna aplikacija) kojem se može pristupiti preko web preglednika ili mobilnoga uređaja. Iz tog razloga je ova aplikacija napravljena kao progresivna web aplikacija. Kako bi se napravila progresivna aplikacija korištena je moderna tehnologija i moderni jezici koji omogućavaju i olakšavaju takav razvoj.

Opisat će se način na koji radi aplikacija, na koji način korisnik koristi aplikaciju te sve mogućnosti koje aplikacija pruža.

Aplikacija je podijeljena na dva dijela koja se sastoje od programiranja na strani klijenta (eng. frontend) i na strani poslužitelja (eng. backend). Za programiranje na strani klijenta koristio se Angular, moderno i sve više globalno popularno JavaScript razvojno okruženje (eng. framework) dok se na strani poslužitelja koristio Laravel, jedan od najpopularnijih PHP razvojnih okruženja.

Čitanjem ovog rada objašnjeno je na koji je način napravljen klijentski dio aplikacije (Angular) i njegovi dodatni paketi (HTML5, Bootstrap, RxJs...) koji olakšavaju izradu aplikacije te znatno ubrzavaju njezin razvoj i brzinu izvođenja. Isto tako, na poslužiteljskom dijelu aplikacije objasnit će se pohranjivanje podataka u bazu (PostgreSQL) gdje se također koriste dodatni paketi (JWT, DingoApi...) koji služe za jednostavno autoriziranje i komuniciranje klijentskog dijela sa serverom i bazom.

Kako se radi o modernoj aplikaciji korišten je i Socket.io, sustav jednostavne komunikacije klijenta i servera te primanja podataka u stvarnom vremenu bez nepotrebnog osvježavanja (eng. reload) stranice ili paljenja i gašenja aplikacije.

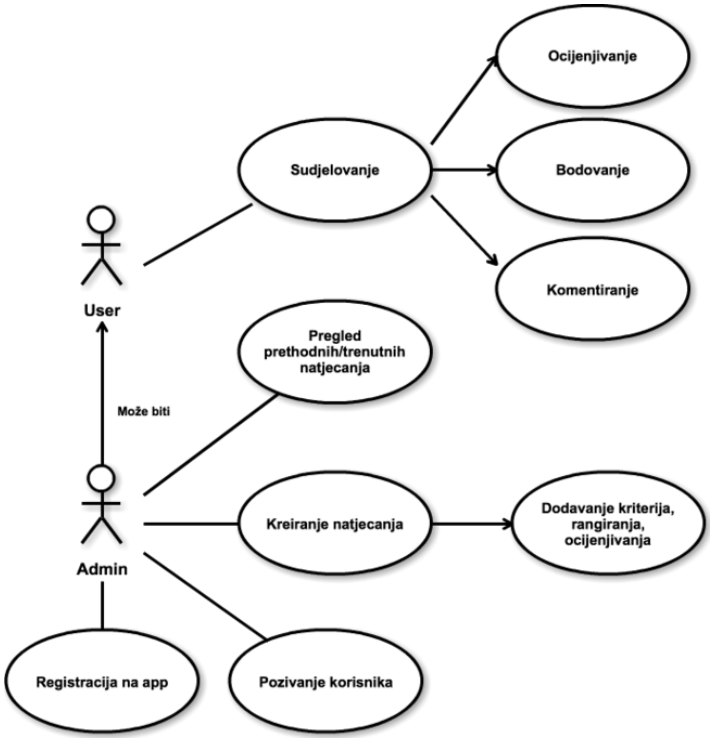
Struktura rada sadrži upute korištenja i mogućnosti aplikacije i način izrade aplikacije. Ona je izrađena tako da ima dijelove odvojene na poslužiteljski i klijentski dio, koji su objašnjeni i provedeni kroz određene primjere te dodatan dio aplikacije (“Socket.io”) koji je neovisan o programiranju na strani klijenta i poslužitelja.

2. O aplikaciji

Aplikacija je namijenjena svim korisnicima koji organiziraju različita natjecanja. Aplikaciju koriste organizatori natjecanja dok drugi korisnici (ocjenjivači) pristupaju samo jednom dijelu aplikacije, a to je dodjeljivanje rezultata. Ocjenjivače odabire administrator (glavni korisnik) aplikacije koji se registrirao i prijavio u sustav. Također, administrator upisuje imena sudionika natjecanja nad kojima će se vršiti ocjenjivanje. Prilikom kreiranja natjecanja korisnik mora:

- dodijeliti ime natjecanju
- odabrati tip natjecanja (bodovanje, rangiranje ili ocjenjivanje)
- dodijeliti minimalni i maksimalni broj bodova (nije potrebno dodjeljivanje ako je odabran način rangiranja)
- upisati adrese elektroničkih pošta (eng. email) korisnika koji će biti pozvani na sudjelovanje u natjecanju
- upisati imena sudionika natjecatelja (osobe koje će se ocjenjivati)

Slika 1 prikazuje mogućnosti koje ima administrator odnosno korisnik aplikacije.



Slika 1. Dijagram korištenja sustava (eng. use-case)

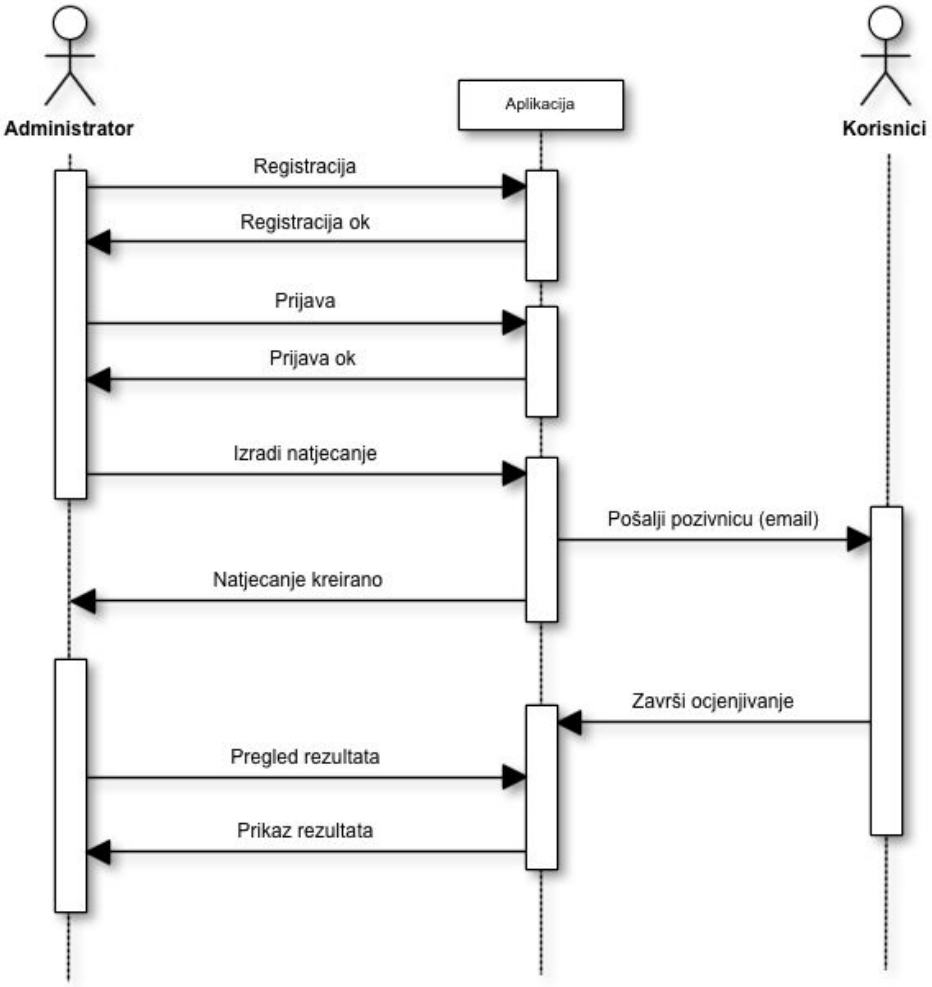
Izvor: izradio autor

Kreiranjem novog natjecanja, adrese elektroničkih pošta koje je korisnik upisao koriste se kao pozivnice za ocjenjivače. Svaki od ocjenjivača pristupa natjecanju preko poveznice koju dobije u svom pretincu elektroničke pošte. Administrator natjecanja ima mogućnost pratiti ocjenjivanje, tj. kako ocjenjivači dodjeljuju bodove i završavaju ocjenjivanje tako administrator na svom sučelju prati rezultate.

Dodatne mogućnosti koje administrator još ima su:

- zatvoriti natjecanje u bilo kojem trenutku bez obzira jesu li svi ocjenjivači završili svoje ocjenjivanje
- pregled svih aktivnih natjecanja i detalja određenog natjecanja
- pregled svih prethodno kreiranih natjecanja, njihove detalje i rezultate

Na sljedećoj slici 2 prikazuje se dijagram korištenja podataka (eng. use sequence diagram).

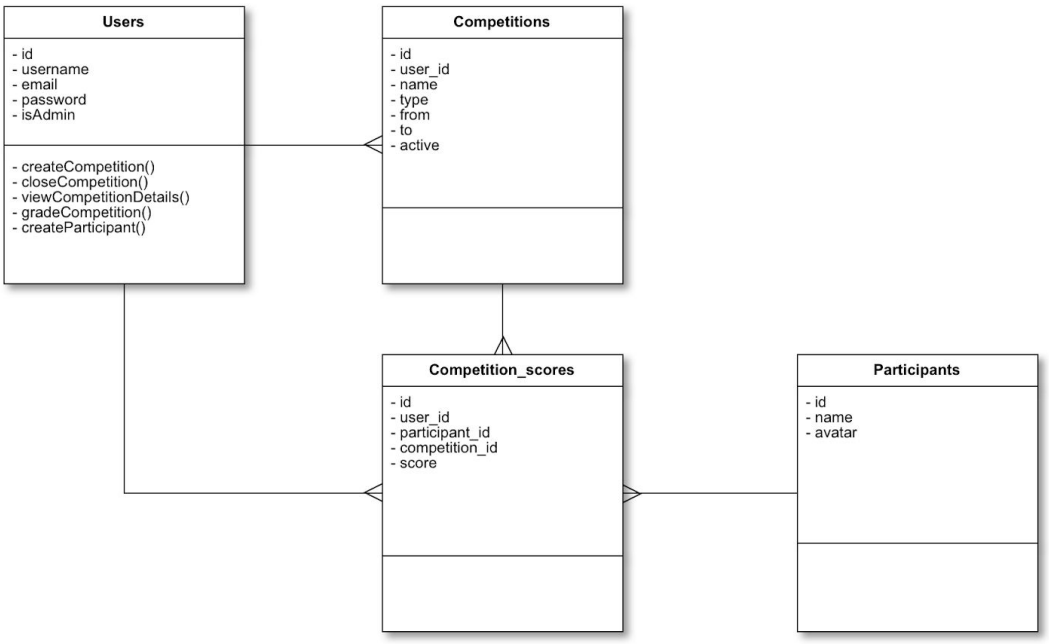


Slika 2. Dijagram korištenja podataka

Izvor: izradio autor

Dijagram korištenja toka podataka sa slike prikazuje način na koji korisnici koriste aplikaciju. Najprije se administrator registrira u sustav aplikacije te se prijavljuje u sustav. Nakon uspješne prijave administrator izrađuje novo natjecanje. Sustav aplikacije prilikom kreiranja natjecanja šalje pozivnice korisnicima za sudjelovanje. Kad korisnici daju svoje ocjene i završe ocjenjivanje, sustav aplikacije prikazuje administratoru rezultate za pojedinog korisnika. Kad svi korisnici daju svoje ocjene, natjecanje se zatvara.

Slika 3 prikazuje klasni dijagram (eng. class diagram) aplikacije.



Slika 3. Klasni dijagram aplikacije

Izvor: izradio autor

S klasnog dijagrama aplikacije prikazano je da korisnik kreira natjecanje. Svako novo kreirano natjecanje ima i “id” korisnika od onog korisnika koji je kreirao natjecanje. Također kreiraju se i “participants”, koji nisu vezani na korisnika, ali su vezani na “competition_scores” klasu. “Participants” klasi pripadaju svi sudionici natjecanja koji dobivaju svoju ocjenu. U “competition_scores” klasi nalaze se konačne ocjene i to tako da se povezuje koji korisnik (“user”) je za koje natjecanje (“competition”) dao određenu ocjenu (“score”) za određenog sudionika natjecanja (“participant”).

3. Programiranje na strani klijenta

Programiranje na strani klijenta (eng. frontend) odnosno klijentski dio aplikacije (ono što klijent vidi i koristi) je napravljen u Angular platformi. Također, uz Angular dio korištene su i druge tehnologije kao i različiti dodatni paketi koji olakšavaju razvoj aplikacije:

- HTML5
- Bootstrap 4
- Animate.css
- TypeScript
- RxJs
- NG2-dnd
- Socket.io-client

Svaki od ostalih paketa i tehnologija bit će objašnjeni detaljnije kasnije u tekstu i bit će naveden primjer kako bi se vidjelo na koji je način svaki od njih korišten za izradu aplikacije.

Klijentski dio aplikacije koji je napravljen u Angular sustavu nema mogućnost spremanja podataka u bazu, odnosno nema podršku da se poveže s bazom, ali taj problem je riješen tako da klijentski dio komunicira s poslužiteljskim prilikom dohvaćanja podataka iz baze ili spremanja podataka u bazu. Poslužiteljski dio aplikacije opisan i objašnjen je u daljnjem sadržaju ovog rada.

3.1. Angular

Angular je platforma otvorenog koda (eng. open-source) koja služi za izradu web-aplikacija temeljenog na TypeScriptu, koju su izradili jedan tim iz Google-a, zajednica JavaScript pojedinaca i drugih korporacija koje nisu vezane uz Google. Angular je nastao prepisivanjem prijašnjeg projekta AngularJS koji se smatra prethodnikom sadašnjeg Angulara (Murray i sur., 2017:4). Angular je ujedno i JavaScript razvojno okruženje koje na jednostavan način pruža niz značajnih i gotovih rješenja za implementaciju složenih zahtjeva modernih aplikacija kao što su povezivanje podataka (eng. data binding), navigiranje (eng. routing) i animacije.

Kod izrade ove aplikacije korištena je verzija 5.2 odnosno “Angular 5”. Glavna prednost i pogodnost ovog Angulara je ta što se u petoj verziji fokusiralo na progresivne web aplikacije (eng. PWA - progressive web app) tako da je ova aplikacija ustvari napravljena kao PWA.

Progresivne web aplikacije (PWA) su web aplikacije koje su normalne web stranice, ali se mogu pojaviti korisniku kao tradicionalne aplikacije ili izvorne (eng. native) mobilne aplikacije. Aplikacija pokušava kombinirati značajke koje nude najsuvremeniji preglednici s izgledom mobilnih aplikacija.

3.1.1. Primjer Angulara

Kao primjer uzet ćemo početnu stranicu, odnosno stranicu na kojoj se nalazi prijava u aplikaciju. Jedna web stranica ili dio web stranice se može u Angularu tretirati kao Angular komponenta (eng. Angular component). Ako se ista ili slična stranica nalazi na više mjesta tako se komponenta može ponovno iskoristiti (eng. reuse) kako se iste komponente ne bi ponavljale. Jedna komponenta sastoji se od:

- HTML predložak (eng. template)
- CSS (Cascading Style Sheets) ili SCSS (Sassy CSS) datoteka (eng. file)
- "spec" datoteka
- JS (JavaScript) ili TS (TypeScript) dio komponente

U HTML predlošku se nalazi izgled web stranice ili određeni dio/sadržaj stranice koji se u određenom trenutku prikazuje na zaslonu. Kako bi se odvojio izgled komponente od stiliziranja te kako bi preglednost bila ljepša, stil komponente se piše u CSS-u. CSS je stilski jezik koji se koristi za stiliziranje HTML dokumenta, sadržaja ili komponente (HTML komponente) (Murray i sur., 2017:16).

"Spec" datoteka služi za pisanje jediničnih (eng. unit) testova. Jedinični test je metoda testiranja softvera kojim se testiraju pojedinačni dijelovi koda, skupovi jednog ili više modula računalnih programa s pridruženim kontrolnim podacima, postupci korištenja modula i postupci rada modula kako bi se utvrdilo jesu li moduli prikladni za korištenje.

U JS ili TS datoteci se nalazi definicija komponente. Tamo se nalazi dekorater komponente u kojem se definira naziv komponente, HTML predložak i CSS datoteka da bi se u aplikaciji sve definirane stvari mogle povezati (Murray i sur., 2017:14). Ovisno o ECMAScriptu (5 ili 6) koristi se JS odnosno TS ekstenzija datoteke. ECMAScript je skriptni jezik (standard) koji se bazira za JavaScriptu. ECMAScript je standardiziran od strane "ECMA International" organizacije po specifikacijama ECMA-262 i ECMA-402 (Haverbecke, 2015:6).

Na slici 4 vidi se primjer “login” komponente koja će biti detaljnije objašnjena.

Slika 4. Datoteke jedne komponente

Izvor: izradio autor

Kako bi se prikazala komponenta “login” potrebno je u ruteru (eng. router) projekta povezati određenu rutu za određenu komponentu. Pa se sa sike 5 vidi vezivanje komponente “login” uz određenu rutu.

```
{
    path: "",
    pathMatch: "full",
    redirectTo: "login",
},
{
    path: "login",
    children: [
        {
            path: "",
            pathMatch: "full",
            component: LoginComponent
        },
        {
            path: "register",
            pathMatch: "full",
            component: RegisterComponent
        }
    ]
},
```

Slika 5. Primjer konfiguracije rutera

Izvor: izradio autor

U ovom slučaju kada korisnik dođe na početnu stranicu prikazuje se “login” komponenta, odnosno prikaz prozora za prijavu.

Kako se komponenta mora prikazati na web stranici, izgled komponente se nalazi u HTML predlošku i izgleda kao na slici 6.

```
<div class="flex-wrapper h-100 container-fluid bg-gradient">
    <div class="row justify-content-center align-middle">
        <div class="col-md-6 col-sm-8 col-xs-12 login-div px-4 animated fadeIn">
            <h1>Prijavi se</h1>
            <form (submit)="login()" class="mt-4" [formGroup]="loginForm">
                <div class="form-group">
                    <input type="email" class="form-control" placeholder="E-mail"
                           [formControl]="loginForm.get('email')">
                </div>
                <div class="form-group">
                    <input type="password" class="form-control" placeholder="Lozinka"
                           [formControl]="loginForm.get('password')">
                </div>
                <button type="submit" class="btn btn-primary btn-block btn-large">Prijava</button>
                <a routerLink="/login/register" class="mt-4 btn">Registracija</a>
            </form>
        </div>
    </div>
</div>
```

Slika 6. HTML predložak “login” komponente

Izvor: izradio autor

S prethodne slike vidi se da se u predlošku komponente nalaze standardni HTML tagovi (div, h1, form, input). Također, vidi se i atribut “class” koji određenu klasu povezuje s klasom u SCSS fajlu.

Kod “input” taga nalazi se atribut “[formControl]”, a on služi za vezivanje vrijednosti (eng. data bind) s TS fajlom.

U TS fajlu deklarirana je varijabla “loginForm” koja prima vrijednosti “email” i “password”

```
this.loginForm = fb.group( controlsConfig: {
    email: fb.control( formState: ""),
    password: fb.control( formState: "")
});
```

Slika 7. Izgled “loginForm” varijable

Izvor: izradio autor

Slika 7 prikazuje kako je “loginForm” varijabla kojoj se u konstruktoru komponente dodjeljuje uloga forme. Tako su “email” i “password” vrijednosti koje su na početku (kod prikazivanja komponente) prazne vrijednosti sve dok korisnik ne upiše potrebne podatke za prijavu.

Kad korisnik upiše potrebne podatke i klikne (odabere) “Prijava”, odabrani gumb poziva funkciju “login” kao što je prikazano na slici 6. Na toj je slici prikazano da se prilikom potvrde (“submit”) forme aktivira funkcija koja prijavljuje korisnika u sustav. Izgled funkcije nalazi se na sljedećoj slici 8.

```
login() {
    const body = this.loginForm.value;
    this.userApiService.login.POST(new NeoRequest( data: {body: body})).subscribe(
        next: (token: ITokenResponse) => {
            this.tokenService.setToken(token);
            this.router.navigateByUrl( url: "nav");
        },
        error: (error) => {
            this.showErrorAlert();
            switch (error.status) {
                case 401:
                    this.loginForm.setErrors( errors: {unauthorized: true});
                    break;
                default:
                    this.loginForm.setErrors( errors: {unknown: true});
                    break;
            }
        }
    );
}
```

Slika 8. Izgled “login” funkcije

Izvor: izradio autor

Kada se pozove funkcija “login”, koja iz servisa “userApiService” kreira upit na API (programsko sučelje aplikacije, eng. Application program interface) kome se šalju upisani podaci. Potreban servis za ostvarivanje komunikacije je generiran na poslužiteljskom dijelu te je povezan s određenom rutom (“/api/login”).

Ako je pozitivan odgovor sa strane poslužiteljskog dijela aplikacije, u odgovoru se dobiva JWT (Json Web Token) token koji se sprema u “local storage” (dio preglednika pogodan za spremanje podataka). Na taj se način osigurava da se može iz bilo kojeg dijela aplikacije ili komponente dohvatiti vrijednost tokena. Taj će se token kasnije koristiti kao autorizacija korisnika, prilikom slanja podataka na poslužiteljski dio aplikacije.

Ako se slučajno dogodi kakva greška na poslužiteljskom dijelu odgovor koji se vrati je negativan te se prikazuje greška na zaslonu korisnika.

Kada korisnik dobije svoj token aplikacija preusmjerava korisnika na autoriziran dio aplikacije gdje može krenuti s izradom natjecanja, a na koji inače ne može pristupiti ako prethodno nije dobio svoj token odnosno nije se prijavio preko forme za prijavu u aplikaciju.

3.2. Dodatne tehnologije i paketi vezani uz klijentski dio programiranja

HTML5 je najnovija verzija standarda koja definira HTML. HTML5 je uveo nove elemente i atribute koji omogućuju izgradnju moćnih web stranica i aplikacija. Od HTML5 tehnologije u aplikaciji, korištena je mogućnost spremanja podataka na klijentski dio aplikacije te efikasan i jednostavan rad s multimedijima (slika avatara). Na slici 9, primjer je korištenja novog “img” elementa kojeg HTML5 omogućava.

```
<img class="avatar" [src]="result.avatar" *ngIf="result.avatar">
```

Slika 9. Primjer korištenja “img” taga

Izvor: izradio autor

Bootstrap je najpopularnije svjetsko razvojno okruženje za izgradnju responzivnih, mobilno-orijentiranih aplikacija. U aplikaciji koristi se “Bootstrap 4” koja je ujedno i zadnja verzija tog razvojnog okruženja.

Bootstrap se koristi tako da se njegove klase dodaju određenim HTML komponentama te se automatizacijom komponente smještaju po stranici, isto kao što se i automatski povećavaju i smanjuju ovisno o veličini ekrana (eng. responsive). Na slici 10 prikazan je primjer elementa s Bootstrap klasama.

```
<div class="row">
    <div class="col-md-6 col-sm-8 col-xs-12">

    </div>
</div>
```

Slika 10. Primjer HTML elementa s Bootstrap klasama

Izvor: izradio autor

Na slici 10 prvi “div” element ima klasu “row”. Takva klasa daje elementu veličinu koja je jednaka veličini ekrana samo s dodatnim određenim marginama.

Sljedeći “div” element ima drugačije klase: “col-md-6”, “col-sm-8”, “col-xs-12”. Svaka od klasa daje tom elementu određenu širinu za pojedine veličine ekrana.
Svaka od klasa ima drukčije značenje:

- “col-md-6” - za rezolucije manje od 992px element ima 50% širine prvog diva
- “col-sm-8” - za rezolucije manje od 768px element ima 67% širine prvog diva
- “col-xs-12” - za rezolucije manje od 540px element ima 67% širine prvog diva

Bootstrap je sam po sebi jako veliko razvojno okruženje koje ima puno unaprijed definiranih i gotovih klasa te ima jako veliku primjenu kod izrade web stranica i web aplikacija.

Animate.css je skupina lijepih, zabavnih animacija koje se mogu koristiti u projektima. Ovaj paket se koristi kao zamjena za GIF (Graphics Interchange Format) animacije. Kako je bitno da aplikaciju budu što brže, tako se i razvojni programeri brinu da koriste što manje elemenata koji usporavaju web stranicu ili aplikaciju. Inače, slike-animacije zauzimaju puno memorije i dugo se učitavaju dok je Animate.css, CSS paket koji koristi CSS jezik za animacije, koji ne zauzima puno memorije. Na slici 11. vidi se jednostavno korištenje klasa koje dolaze uz taj paket.

```
<div class="animated fadeIn">

</div>
```

Slika 11. Primjer korištenja Animate.css paketa

Izvor: izradio autor

Kako čisti JavaScript ima svojih mana koristi se TypeScript iako to nije nužno. Glavni razlog radi kojeg se sve češće koristi TypeScript leži u tome da JavaScript kao takav nema tipove varijabli koji su učestali problem. Velik dio grešaka (eng. bugs) u aplikacijama proizlazi upravo radi toga što varijable u aplikacijama nemaju određene tipove podataka (npr. zbrajanje teksta i broja) (Murray i sur., 2017:66). Također, dodjeljivanjem tipa varijablama u modernim tekst uređivačima (eng. code/text editor) dobivamo automatski pristup funkcijama koje se mogu izvršiti nad određenom varijablom.

Kada se piše kod u TypeScrpit-u on se prevodi (eng. compile) u JavaScript tako da se na kraju poslužuje JavaScript kod. Sa slike 12 vidi se primjer deklaracije jedne varijable koristeći TypeScript.

```
private router: Router,
```

Slika 12. Deklariranje varijable koristeći TypeScript
izvor: izradio autor

Ta se varijabla koristi samo u komponenti u kojoj je deklarirana te tako druge komponente u aplikaciji nemaju nepotrebne varijable koje zauzimaju memoriju i usporavaju rad aplikacije. Na sljedećoj slici 13 je primjer korištenja iste te varijable sa slike 12.

```
this.router.navigateByUrl( url: "");
```

Slika 13. Korištenje funkcije nad varijablom “router”
izvor: izradio autor

Primjer sa slike 13 koristi gotovu Angular klasu “Router” preko koje se navigira (preusmjerava) korisnika kroz aplikaciju. U ovom slučaju korisnik će, kad se pozove ova funkcija, biti preusmjeren na početnu stranicu aplikacije.

Sljedeća stvar koja se koristi u Angular-u je RxJs (Reactive Extensions Library for JavaScript). RxJS je biblioteka (eng. library) za reaktivno programiranje. Koriste se observeri (eng. observables) iz razloga da bi se olakšao rad s asinkronim pozivima ili povratnim (eng. callback) funkcijama. Opservere se može zamisliti kao polje (niz) čiji elementi dolaze asinkrono tijekom nekog vremena (učitavanje komponente, obrade zahtjeva). Opserveri pomažu da upravljaju asinkronim podacima, poput podataka koji dolaze s poslužiteljskog dijela aplikacije. Slika 14 prikazuje primjer jednog od načina na koji se može koristiti RxJS.

```
toast: BehaviorSubject<IToast> = new BehaviorSubject<IToast>( _value: null);

constructor(private toastService: ToastService) {
    this.toastService.showing.subscribe(this.toast);
}
```

Slika 14. Primjer RxJS-a

Izvor: izradio autor

Slika 14 prikazuje korištenje skočnih (eng. popup) prozora koji prikazuju greške, upozorenja ili informacije (eng. toast).

Paket NG2-dnd koristi se kod rangiranja sudionika natjecanja. Ovaj paket omogućuje sortiranje komponente u listi komponenti, koristeći pritom metodu “vuci i ispusti” (eng. drag and drop). Na slici 15. je prikazana primjena tog paketa.

```
<ul class="list-group my-4 ul-sort" #sortForm dnd-sortable-container
    [sortableData]="compData.participants">
    <li class="list-group-item"
        *ngFor="let participant of compData.participants; let i = index"
        dnd-sortable [sortableIndex]="i">
        <div class="view">
            <img class="avatar" *ngIf="participant.avatar" [src]=participant.avatar />
            <span [class.hasAvatar]="participant.avatar">{{participant.name}}</span>
            <span class="participantId" hidden>{{participant.id}}</span>
        </div>
    </li>
</ul>
```

Slika 15. Primjer korištenja NG2-dnd paketa

Izvor: izradio autor

Kako bi se koristio taj paket potrebno je dodati “dnd-sortable-container” direktivu koja označava da je taj element ustvari komponenta koja se može sortirati. Kod elemenata koji se vuku i spuštaju dodaje se direktiva “dnd-sortable” te svaka od njih ima svoj određeni indeks (redni broj) po kojemu se prepoznaje na kojoj je poziciji. Indeks se komponenti dodjeljuje preko direktive “sortableIndex”.

3.3. Socket

Socket omogućuje dvosmjernu komunikaciju na temelju događaja u realnom vremenu. Radi na svakoj platformi, pregledniku ili uređaju, fokusirajući se na pouzdanost i brzinu.

Socket se koristi iz razloga da bi korisnici web aplikacije ili stranice dobili podatke u stvarnom vremenu bez osvježavanja (eng. reload) stranice na kojoj se nalazi. Za ovu aplikaciju korištena je mala skripta napravljena u JavaScriptu koja sadržava Socket. Na sljedećoj slici 16. prikazan je kod korištenja skripte.

```
var app = require('express')();
var server = require('http').Server(app);
var io = require('socket.io')(server);

server.listen(3000);

app.get('/', function(request, response) {
    response.send('Alive');
});

io.on('connection', function(socket) {
    socket.on('gradeComp', function (message) {
        io.emit('gradeComp', message);
    })
});
```

Slika 16. Izgled JS skripte sa Socket

Izvor: izradio autor

Ključne stvari za Socket koje su prikazane u primjeru su sljedeće. Server sluša promjene (akcije) aplikacije na priključku (eng. port) 3000. Svaka promjena koja dođe na taj priključak ova skripta registrira. Ako se dogodi promjena s imenom “gradeComp” skripta emitira (signalizira) da se dogodila određena promjena.

S klijentskog dijela aplikacije (Angular) potrebno je također poslati podatak o tome da su se podaci promijenili.

Kako bi se s Angular-a poslali podaci korišten je paket Socket.io-client. Pa se, kako je prikazano na slici 17, šalje upit na JavaScript skriptu.

```
this.socket.emit( event: "gradeComp");
```

Slika 17. Emitiranje (signaliziranje) sa Angular-a
Izvor: izradio autor

Također, postoji i takozvani slušač promjena (slika 18).

```
this.socket.on( event: "gradeComp", function () {
    this.getResult();
}.bind(this));
```

Slika 18. Emiter (primatelj) obavijesti u Angular-u
Izvor: izradio autor

Dakle, ako emiter signalizira da se dogodila promjena, odnosno da su se novi podaci uspješno zapisali na bazu, emiter obavještava da se dogodila promjena pod nazivom “gradeComp”. Kada emiter koji prima obavijest, primi da se dogodila promjena “gradeComp” izvršava se određena funkcija, u ovom slučaju ponovni upit na poslužiteljski dio aplikacije koji dobiva određene podatke kod kojih se dogodila promjena.

4. Programiranje na strani poslužitelja

Programiranje na strani poslužitelja (eng. backend) je dio aplikacije napravljen u Laravelu. Laravel je jedan od najpoznatijih i najkorištenijih razvojnih alata koje je napisan je u PHP jeziku. U Laravel projektu korišteni su drugi PHP paketi koji će kasnije biti detaljnije objašnjeni. Također, dodatne stvari koje su vezane uz poslužiteljski dio aplikacije objašnjene su kasnije u tekstu.

Korišteni paketi uz Laravel razvojno okruženje su:

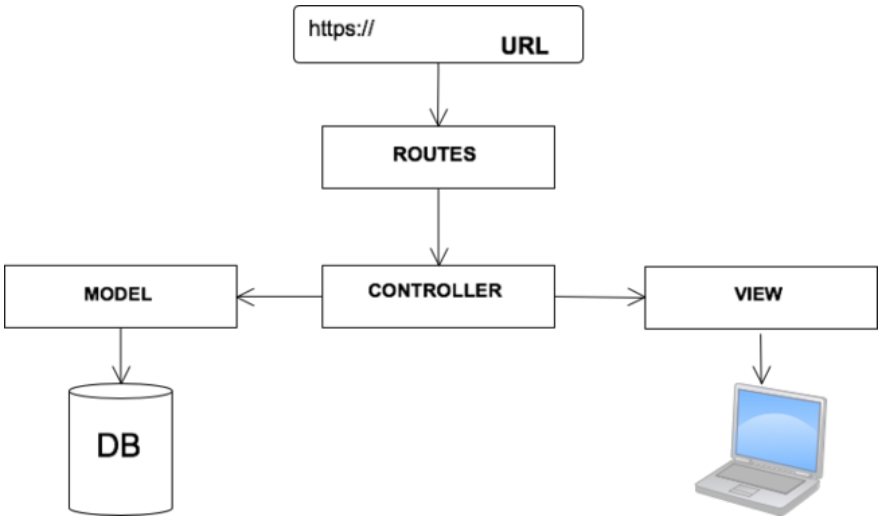
- JWT (Json Web Token)
- Dingo API
- Laravel-lang

Dodatne stvari vezane uz programiranje na strani poslužitelja:

- PostgreSQL
- Mailtrap
- Sendgrid

4.1. Laravel

Laravel je jedan od popularnih PHP razvojnih okruženja koji se koristi u svijetu za izradu tradicionalnih MVC (eng. model-view-controller) projekata. Izgled osnovnog MVC predloška nalazi se na slici 19.



Slika 19. Osnovni MVC izgled Laravela

Izvor: izradio autor

Kao što je prije napomenuto Laravel je korišten samo za poslužiteljski dio aplikacije tako da se “VIEW” dio Laravela ne koristi u projektu već je “VIEW” prebačen na Angular, klijentski dio aplikacije koji je opisan prethodno.

4.1.1. Primjer Laravela

Za primjer opisat će se i prikazati na koji način je napravljeno dohvaćanje detalja za određeno natjecanje koje korisnik odabere.

Sve rute na koje se vrši pozivanje nalaze se u mapi “routes”, u “api.php” datoteci. Pregled ruta izgleda kao na slici 20.

```
Route::group( [ 'middleware' => 'jwt.auth' ], function () {

    Route::group(['prefix' => 'competition'], function() {
        Route::post( uri: 'new',  action: 'CompetitionCtrl@store');
        Route::get( uri: 'active',  action: 'CompetitionCtrl@showActive');
        Route::get( uri: 'closed',  action: "CompetitionCtrl@showClosed");
        Route::get( uri: 'activeNumber',  action: 'CompetitionCtrl@activeCount');
        Route::get( uri: 'show/{id}',  action: 'CompetitionCtrl@show');
        Route::delete( uri: 'close/{id}',  action: 'CompetitionCtrl@close');
        Route::post( uri: 'grade',  action: "CompetitionCtrl@gradeCompetition");
        Route::post( uri: 'gradeRank',  action: "CompetitionCtrl@gradeRankCompetition");
        Route::get( uri: 'competitionReport/{id}',  action: 'CompetitionCtrl@getCompetitionReport');
    });

});
```

Slika 20. Primjer ruta u Laravelu

Izvor: izradio autor

Kao što je prikazano na slici vidimo prvi “Route::group” i “middleware” koji je postavljen na grupu ruta. “Middleware” u Laravelu nam služi kako bi određene rute bile zaštićene, odnosno da samo određeni korisnici mogu pristupiti pozivima na te rute. U ovom slučaju koristi se JWT autentifikacija koja će kasnije biti objašnjena.

Za dohvat detalja o određenom natjecanju koristi se ruta “show/{id}” gdje je “id”, identifikacijska oznaka koja je dodijeljena određenom natjecanju te preko koje će se dohvatiti određeni podatci. Za tu odabranu rutu koristi se funkcija “show” koja se nalazi u kontroleru “CompetitionCtrl” i na taj način poziv te funkcije izgleda kao na slici 20.

U “CompetitionCtrl” kontroleru funkcija “show” izgleda kao na slici 21. gdje se kod poziva funkcije prosljeđuje “id” natjecanja.

```
public function show($id)
{
    $competition = Competition::with(['user', 'competitions_score', 'participants', 'graders'])->where( column: 'id', $id)->first();
    $isAdminGrader = false;
    foreach ($competition->getRelation('graders') as $user) {
        if($user->id == $competition->getRelation('user')->id) {
            $isAdminGrader = true;
            if(self::checkIfUserHasMakeGrade($user->id, $competition->id)) {
                $isAdminGrader = false;
            }
            break;
        }
    }
    foreach ($competition->getRelation('graders') as $grader) {
        foreach ($competition->getRelation('competitions_score') as $graderFinish) {
            if(($grader->id == $graderFinish->user_id) && $graderFinish->score != null) {
                $grader = array_set( &array: $grader, key: 'isGrade', value: true);
                break;
            } else {
                $grader = array_set( &array: $grader, key: 'isGrade', value: false);
            }
        }
    }
    return [
        "status" => "success",
        "params" => $competition,
        "isAdminGrader" => $isAdminGrader
    ];
}
```

Slika 21. Izgled “show” funkcije

Izvor: izradio autor

Prvo što se napravi u funkciji je upit na bazu preko modela “Competition” kod kojeg se traži da dohvati podatke natjecanja s određenim “id-em”.

Dodatni parametri koji se prosljeđuju modelu su takozvani “relationships” odnosno veze tablice natjecanja s ostalim tablicama na bazi. Svaki model kreiran u Laravelu povezan je s određenim tablicama u bazi.

Isto tako svaki model ima i “fillable” polje u koje se upisuju imena stupaca iz tablice koji se mogu u određenu tablicu unositi ili uređivati. Primjer “fillable” polja može se vidjeti na slici 22.

```
protected $fillable = [
    'user_id',
    'name',
    'type',
    'from',
    'to'
];
```

Slika 22. “Fillable” polje modela “Competition”

Izvor: izradio autor

Na slici 23. vide se svi “relationships” modela “Competition” koji se pozivaju preko “with” funkcije kao što je prikazano na slici 21.

```php
function user() {
    return $this->belongsTo( related: User::class,  foreignKey: 'user_id',  ownerKey: 'id');
}

function competitions_score() {
    return $this->hasMany( related: CompetitionScore::class);
}

function participants(){
    return $this->hasManyThrough(
         related: Participant::class,
         through: CompetitionScore::class,
         firstKey: 'competition_id',
         secondKey: 'id',  localKey: 'id',  secondLocalKey: 'participant_id')
        ->distinct('participant_id');
}
function graders(){
    return $this->hasManyThrough(
         related: User::class,
         through: CompetitionScore::class,
         firstKey: 'competition_id',
         secondKey: 'id',  localKey: "id",  secondLocalKey: "user_id")
        ->distinct("user_id")->orderBy( column: "id");
}
```

Slika 23. “Relationships” modela “Competition”

Izvor: Izradio autor

Svaka funkcija predstavlja vezu na druge modele, odnosno na druge tablice. Ukratko objašnjeno, veze preko modela su iste kao i veze na bazi preko vanjskih ključeva.

Tako da za određeno natjecanje dobijemo sve potrebne detalje za odabrano natjecanje, a funkcije su:

- “users” - koji korisnik je kreirao natjecanje
- “participants” - tko sudjeluje na natjecanju, koga se ocjenjuje
- “graders” - korisnici koji vrše ocjenjivanje
- “competitions_score” - koji ocjenjivač je za kojeg polaznika natjecanja dao koju konačnu ocjenu

Nakon što se dohvate potrebni detaljni podaci za određeno natjecanje napravljena je provjera je li administrator (osoba koja je napravila natjecanje) također ocjenjivač. Provjera se radi tako da se dohvate svi ocjenjivači natjecanja, usporede njihovi “id-evi” sa “id-em” u “users” tablici te ako je postavljena pozitivna vrijednost “true” za određenog korisnika u “isAdmin” stupcu onda je administrator također pristupio ili ima pristup natjecanju.

“Foreach” petlja, koja je prikazana na slici 24. povezuje svakog ocjenjivača s pripadajućim rezultatom.

```
foreach ($competition->getRelation('graders') as $grader) {
    foreach ($competition->getRelation('competitions_score') as $graderFinish) {
        if(($grader->id == $graderFinish->user_id) && $graderFinish->score != null) {
            $grader = array_set( &array: $grader,  key: 'isGrade',  value: true);
            break;
        } else {
            $grader = array_set( &array: $grader,  key: 'isGrade',  value: false);
        }
    }
}
```

Slika 24. Prikaz povezivanja korisnika s određenom ocjenom

Izvor: izradio autor

Kao što je prikazano na slici 24. dohvaća se određeni “relation” koristeći funkciju “getRelation”. Prvi “relation” sadrži podatke koji su korisnici vezani uz natjecanje dok drugi dohvaća informacije o rezultatima za određene korisnike te se također uz njih stavlja informacija je li korisnik završio natjecanje ili ne.

Na kraju, kada su svi podatci dohvaćeni, podatci se vraćaju na klijentski dio aplikacije. Na slici 21 prikazana su tri parametra koja se vraćaju. To su:

- “status” - status koji govori jesu li podaci uspješno dohvaćeni
- “params” - parametri detalja o natjecanju
- “isAdminGrader” - parametar je li administrator ujedno i ocjenjivač

4.2. Dodatni paketi na poslužiteljskom dijelu aplikacije

Za bazu podataka koristila se PostgreSQL baza. PostgreSQL je objektno-relacijska baza otvorenog koda (eng. open-source). Takva baza podataka nudi sustav za upravljanje bazom podataka koji je sličan relacijskoj bazi podataka, ali s objektno orijentiranim modelom baze podataka: objekti, klase i nasljeđivanje su izravno podržani u shemama baze podataka i u upitima na bazu (Juba, Vannahme i Volkov 2015:31).

Kod kreiranja tablice na bazi “Laravel” ima rješenje koristeći “migrations”. “Migrations” omogućuju timu da jednostavno izmijeni i izradi shemu baze podataka za aplikaciju (Malatesta, 2015:35). Primjer migracije je na slici 25 gdje se vidi “migration” za kreiranje “users” tablice na bazi.

```
Schema::create('users', function (Blueprint $table) {
    $table->increments( column: 'id');
    $table->string( column: 'username')->unique()->nullable();
    $table->string( column: 'email')->unique();
    $table->string( column: 'password');
    $table->boolean( column: 'isAdmin')->default( value: false);
    $table->rememberToken();
    $table->timestamps();
});
```

Slika 25. Kreiranje “users” tablice koristeći “migration”

Izvor: izradio autor

Mailtrap je lažni SMTP (Simple Mail Transfer Protocol) server koji se koristi za testiranja i razvoj aplikacija koje koriste slanje elektroničke pošte. Sustav nudi besplatnu registraciju nakon čega se dobiva virtualni pretinac elektroničke pošte (eng. inbox) u kojem se nalaze “dostavljene” elektroničke pošte. U pretincu se vide podaci tko je pošiljatelj, tko primatelj i sadržaj elektroničke pošte. Jedini je nedostatak servisa što besplatna licenca ograničava korisnika na jedan virtualan pretinac koji može primiti samo 50 elektroničkih pošti što je mali broj ako se vrše testiranja na više korisnika.

Sendgrid je besplatan servis elektroničke pošte koji, kao i Mailtrap služi za slanje elektroničke pošte. Koristeći Sendgrid servis, elektroničke pošte se šalju krajnjim korisnicima. Takav servis se koristi samo u produkciji aplikacije, a ne u testiranju i razvoju.

Primjer slanja elektroničke pošte korisniku prikazano je na slici 26.

```
Mail::send( view: "email.invitation",
    [
        "competition" => $competition,
        "link" => $link,
        "user" => $userMail
    ],function ($mail) use ($userMail) {
    $mail->to($userMail);
    $mail->subject("Poziv za natjecanje!");
});
```

Slika 26. Primjer slanja elektroničke pošte korisniku

Izvor: izradio autor

Za slanje elektroničke pošte koristi se klasa “Mail” koja je integrirana u Laravel razvojno okruženje. Također, funkcija “send” (koja šalje) prima parametre “view”, “data”, “callback”. “View” parametar je HTML predložak, odnosno izgled elektroničke pošte sadržaja dok su “data” podaci (parametri) koji se prosljeđuju iz kontrolera u HTML predložak. Preko “callback” parametra izvršava se slanje elektroničke pošte gdje se određuje kome se šalje elektronička pošta (“to” funkcija) i naslov elektroničke pošte (“subject” funkcija).

JWT (JSON web token) je otvoreni standard (RFC 7519) koji definira kompaktan i samostalan način za siguran prijenos informacija između projekata koji komuniciraju preko JSON objekta. Te se informacije mogu potvrditi i pouzdane su jer su digitalno potpisane. JWT se u ovoj aplikaciji koristi za autentifikaciju korisnika. Korisnik koji se prijavi u aplikaciju, prilikom uspješne prijave dobiva svoj JWT koji se sprema u “local storage” preglednika. Također, JWT sadrži i podatke o korisniku koji je prijavljen. Kako klijentski dio aplikacije traži određene informacije iz baze podataka, klijentski dio za svaki upit šalje i JWT koji se koristi da se na poslužiteljskom dijelu aplikacije korisnik autentificira, a isto tako da se zna koji korisnik je poslao zahtjev. Na slici 27 vidi se izgled jednog od JWT-a koji se nalazi na klijentskom dijelu aplikacije.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vZGlwbG9tc2tpLW5hdGplY2FuamUubWU6ODA4MC9hdXRo
L2xvZ2luIiwiaWF0IjoxNTI3MTYwNDc1LCJleHAiOjE1MjcxNzg0NzUsIm5iZiI6MTUyNzE2MDQ3NSwianRpIjoiaUxRRVJXeHNYW
TIzUFpiaSIsInN1YiI6NCwicHJ2IjoiODdlMGFmMWVmOWZkMTU4MTJmZGVjOTcxNTNhMTRlMGIwNDc1NDZhYSIsInVzZXIiOnsiaW
QiOjQsInVzZXJuYW1lIjpudWxsLCJlbWFpbCI6InNhQHN1cGVyaXVzLmhyIiwiaXNBZG1pbiI6dHJ1ZSwiY3JlYXRlZF9hdCI6IjI
wMTgtMDUtMjQgMTE6MTQ6MjkiLCJ1cGRhdGVkX2F0IjoiMjAxOC0wNS0yNCAxMToxNDoyOSJ9fQ.h0_Vvl9xkOUqv0K7nDIc0WVAe
zdE-2F0PfHU1r_m0u0
```

Slika 27. Izgled JWT-a

Izvor: izradio autor

Za autentifikaciju na poslužiteljskom dijelu aplikacije postavljen je “middleware” pod nazivom “jwt.auth” koji služi za verifikaciju primljenog JWT-a. Na taj se način osigurava da su web rute zaštićene i sigurne kako ne bi bilo koji korisnik mogao pristupiti određenim podacima (primjer “middleware” nalazi se na slici 20.).

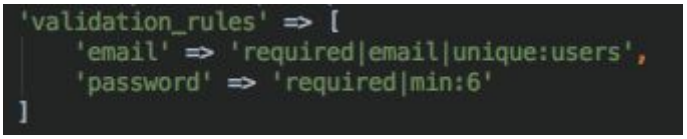
Kod dohvata podataka o tome koji je korisniku napravio upit, odnosno ako su potrebni detalji o korisniku, JWT je potrebno dekodirati. Primjer dekodiranja preko funkcija je prikazan na slici 21.

```
Auth::guard()->user();
```

Slika 28. Dekodiranje JWT-a u Laravelu

Izvor: izradio autor

DINGO Api paket je REST (REpresentational State Transfer) API (eng. Application Programming Interface) servis namijenjen da pruža razvojnom programeru niz alata koji olakšavaju brzu izgradnju vlastitog API-a. Glavni razlog koji se koristio ovaj paket je taj da se na klijentski dio aplikacije vraćaju odgovori u obliku JSON objekata zato što je jednostavnije pristupati i dohvatiti određene podatke. Isto tako se vrši i validacija, tj. provjera ispravnosti poslanih podataka. Primjerice, ako neko polje u tablici treba imati format elektroničke pošte (“email”), a poslan je običan tekst koji nije tog formata, preko ovog alata javi se greška na klijentskom dijelu aplikacije. Primjer na slici 22 pokazuje pravila kod upisanih podataka prilikom registracije u sustav aplikacije.



Slika 29. Primjer pravila validacije za određena polja

Izvor: izradio autor

Sa slike 22 se vide pravila za dva polja, “email” i “password”. Uzmemo li na primjer “email” polje, pravila koja vrijede su sljedeća:

- “required” - potrebno je upisati tekst u polje (polje ne smije biti prazno)
- “email” - upisano polje mora biti tipa “email” odnosno upisani tekst mora biti valjan format elektroničke pošte
- “unique:users” - upisana elektronička pošta mora biti jedinstvena u tablici “users” što znači da ne može postojati više istih u navedenoj tablici

Kako bi se vratio odgovarajući status korisniku je li nešto pogrešno potrebno je vratiti i poruku. Kako je aplikacija na hrvatskom jeziku onda se koristio paket Laravel-lang. Taj paket sadrži višejezičnost tako da se vrlo jednostavno mogu promijeniti poruke ako je potrebno prevesti aplikaciju na druge svjetske jezike.

5. Zaključak

Aplikacija za natjecanja izrađena je na moderan i suvremen način. Prednost takve aplikacije leži u tome da se ona može koristiti na modernim mobilnim uređajima ili web preglednika te je dostupna bilo gdje, na bilo kojem mjestu i u bilo kojem vremenu. Da bi se koristila na starijim uređajima potrebno ju je dodatno nadograditi i proširiti. Aplikacija se može također dodatno nadograditi i proširiti. Jedna od glavnih nadogradnji aplikacije bila bi da se ocjenjivači također mogu prijaviti u sustav, vidjeti natjecanja na kojima su sudjelovali te unaprijediti njihov korisnički račun (eng. account) na razinu administratora natjecanja. Isto tako, svaki od korisnika mogao bi imati detaljnije izvještaje o svim natjecanjima. Primjerice, uvid o učestalosti korisnika koje je administrator pozvao na natjecanja ili prikaz ukupnog broja bodova pojedinog natjecatelja za sva njegova natjecanja na kojima je sudjelovao.

Kroz izradu aplikacije, koristeći moderne tehnologije i globalno popularne alate za razvojno okruženje i pakete vidi se da je jednostavno napraviti vjerodostojnu, sigurnu i brzu aplikaciju koju se može pokrenuti na svim platformama, uređajima i web preglednicima. Brzim napretkom tehnologije napreduju i novi moderni preglednici, novi mobilni uređaji i računala također brzo razvili. Jezici i alati za razvojna okruženja i ostali paketi prate korak u tom razvoju. Za nove stvari da budu što sigurnije i brže, ljudi koji razvijaju alate za programere (developere) također moraju biti u toku s vremenom.

JavaScript jezik, jedan od najpopularnijih jezika smatra se najraširenijim, najbržim i najjednostavnijim jezikom za izradu progresivnih web aplikacija. Postoji i velik izbor alata za jednostavniji razvoj koristeći upravo taj jezik pa tako, u toj tehnologiji dominiraju razvojna okruženja kao što su: Angular (korišten je u ovom projektu), kojeg su razvili programeri iz tvrtke Google, React razvijen od strane Facebook programera i Vue alat kojeg je napravio Evan You, bivši zaposlenik Google-a. Teško se odlučiti za odabir određenog alata jer svaki od njih ima svoje prednosti i mane pa programer sam odlučuje o izboru s namjerom da najbolje iskoristi alat s kojim radi.

Literatura

- MURRAY, N., LERNER, A., COURY, F. i TABORDA, C. (2017), *ng-book The Complete Guide to Angular*, San Francisco, California, Fullstack.io
- HAVERBEKE, M. (2015) *Eloquent JavaScript a Modern introduction to programming 2nd edition*, San Francisco, California, No Starch Press
- MALATESTA, F. (2015), *Learning Laravel's Eloquent*, Livery Place, Ujedinjeno Kraljevstvo, Packt Publishing Ltd
- JUBA, S., VANNAHME, A. i VOLKOV, A. (2015), *Learning PostgreSQL*, Livery Place, Ujedinjeno Kraljevstvo, Packt Publishing Ltd

Web

- https://angular.io/ [Pristupljeno: 24.4.2018.]
- https://getbootstrap.com [Pristupljeno: 24.4.2018.]
- https://daneden.github.io/animate.css/ [Pristupljeno: 26.4.2018.]
- https://www.typescriptlang.org [Pristupljeno: 3.5.2018.]
- http://reactivex.io [Pristupljeno: 5.5.2018.]
- https://github.com/akserg/ng2-dnd [Pristupljeno: 15.5.2018.]
- https://socket.io [Pristupljeno: 17.5.2018.]
- https://laravel.com/docs/5.5 [Pristupljeno: 25.4.2018.]
- https://jwt.io [Pristupljeno: 30.4.2018.]
- https://github.com/dingo/api [Pristupljeno: 5.5.2018.]
- https://github.com/caouecs/Laravel-lang [Pristupljeno: 10.5.2018.]
- https://www.postgresql.org [Pristupljeno: 26.4.2018.]
- https://mailtrap.io [Pristupljeno: 7.5.2018.]
- https://sendgrid.com [Pristupljeno: 20.5.2018.]
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources [Pristupljeno: 25.5.2018.]
- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5 [Pristupljeno: 25.5.2018.]
- https://developer.telerik.com/topics/web-development/introduction-observables-angular-developers/ [Pristupljeno: 26.5.2018.]

Popis slika

Sažetak

Aplikacija natjecanja služi korisniku da kreira “virtualni” žiri koji će svoje bodove dodijeliti sudionicima natjecanja ili ih rangirati ovisno o njihovom postignuću. U bilo kojem trenutku korisnik može pratiti rezultate žirija u realnom vremenu. Također, korisnik može pregledati sva prethodna natjecanja koja je kreirao kao i informacije natjecanja te konačne rezultate koje su dali ocjenjivači za sudionike natjecanja.

Progresivne web aplikacije (PWA) sve su češće pa tako je i ova aplikacija izrađena kao takva. Koristeći se Angular, Laravel, RxJs, Bootstrap i drugim modernim jezicima i tehnologijama koje su dostupne svim programerima i developerima, služe za lakšu izradu aplikacija koje su sigurne, brze i pouzdane. Tako korisnik ima ugodno iskustvo koristeći aplikaciju na bilo kojem uređaju, platformi ili web pregledniku bez brige da mora ovisiti o računalu ili mobilnom uređaju kojeg posjeduje.

Summary

A competition application serves the user to create a "virtual" jury who will assign their points to the contestants or rank them depending on their achievement. A user can, at any time track the jury results in real time. Also, the user can review all the previous competitions he has created as well, the competition information and the final results that the graders have given to the contest participants.

Progressive web applications (PWA) are becoming more common, so this application (project) has been created as such. Using Angular, Laravel, RxJs, Bootstrap and other modern languages and technologies are available to programmers and developers for easy-to-use and easy-to-develop applications that are safe, fast and reliable. This way, the user has a pleasant experience using the application on any device, platform, or web browser without worrying to depend on the computer or mobile device he owns.