Web aplikacija za prihvat podataka i nasumično nagrađivanje

Paradžiković, Iva

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija

Web aplikacija za prihvat podataka i nasumično nagrađivanje

Završni rad

Iva Paradžiković

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Iva Paradžiković
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. pristupnika, god.	4867, 29.07.2020.
JMBAG:	0165087788
Mentor:	izv. prof. dr. sc. Mirko Köhler
Sumentor:	
Sumentor iz tvrtke:	Marko Vukobratović
Naslov završnog rada:	Web aplikacija za prihvat podataka i nasumično nagrađivanje
Znanstvena grana završnog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rada:	Zauzeto za studenticu: Iva Paradžiković. Sumentor iz tvrtke: Marko Vukobratović. Potrebno je napraviti web aplikaciju za prikupljanje kontakt informacija posjetioca sajamskog štanda. Administrator kreira novi događaj i odabire podatke koje posjetioci trebaju ostaviti. Također određuje trajanje ankete i broj nagrada. Nakon što spremi odabir kreira se QR kod koji vodi na anketu. Po završetku događaja, slučajni odabirom se izabiru dobitnici nagrada. Svi sudionici koji su ispravno ispunili anketu sudjeluju u izvlačenju. Anketu je moguće ispuniti jednom s istim podacima. Potrebno je administratoru dati arhivu prethodnih

Datum prijedloga ocjene završnog rada od strane mentora:	17.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	30.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Iva Paradžiković
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. Pristupnika, godina upisa:	4867, 29.07.2020.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za prihvat podataka i nasumično nagrađivanje**

izrađen pod vodstvom mentora izv. prof. dr. sc. Mirko Köhler

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

U današnje vrijeme, velika je potražnja za radnicima i poslovnim suradnicima u svakoj industriji pa tako i u elektrotehničkoj industriji. Uspjeh određene tvrtke uvelike se zasniva na kvalitetnom timu zaposlenika, kao i na pomno odabranim tvrtkama s kojima se surađuje. Upravo iz tog razloga, poslodavci se nerijetko odluče za posjet manifestacijama poput sajmova tvrtki ili posjeta fakultetima koji omogućuju interakciju između tvrtki i potencijalnih zaposlenika, odnosno studenata zainteresiranih za studentsku praksu ili posao.

Cilj ovog završnog rada je izraditi web aplikaciju koja će poslodavcima olakšati vođenje evidencije o osobama koje su pokazale interes za njihovo poslovanje. Aplikacija će omogućavati kreiranje novog događaja u obliku forme, spremanje podataka zainteresiranih osoba u bazu podataka što će biti od velike koristi administratoru, kao i nasumično izvlačenje jedne ili više gore navedenih zainteresiranih osoba koje će biti nagrađene i motivirane za daljnju suradnju s tvrtkom.

U svrhu izrade web aplikacije koristit će se uređivač izvornog koda *Visual Studio Code*, kao i prezentacijski jezik HTML, stilski jezik CSS te programski jezik *JavaScript*.

U drugom poglavlju ovog završnog rada prikazana su već postojeća rješenja problema, odnosno već dostupne aplikacije koje su slične ovoj web aplikaciji. U trećem poglavlju pojašnjeni su zahtjevi koje bi aplikacija trebala zadovoljavati. Četvrto poglavlje detaljnije opisuje tehnologije izrade i alate, točnije programske jezike korištene za izradu. U sljedećem poglavlju prikazano je programsko rješenje, gdje je objašnjen cijeli postupak izrade koda za aplikaciju. Šesto poglavlje demonstrira način korištenja aplikacije i dizajn sučelja.

1.1. Zadatak završnog rada

Zadatak završnog rada je izrada web aplikacije za prikupljanje kontakt informacija posjetioca sajamskog štanda. Administrator kreira novi događaj i odabire podatke koje posjetioci trebaju ostaviti, određuje trajanje ankete i broj nagrada. Nakon što spremi odabir kreira se QR kod koji vodi na anketu. Po završetku događaja, slučajnim odabirom se odabiru dobitnici nagrada. Svi sudionici koji su ispravno ispunili anketu sudjeluju u izvlačenju. Anketu je moguće ispuniti jednom s istim podacima. Potrebno je administratoru dati arhivu prethodnih događaja, kao i pregled svih jedinstvenih kontakata.

2. PREGLED PODRUČJA TEME

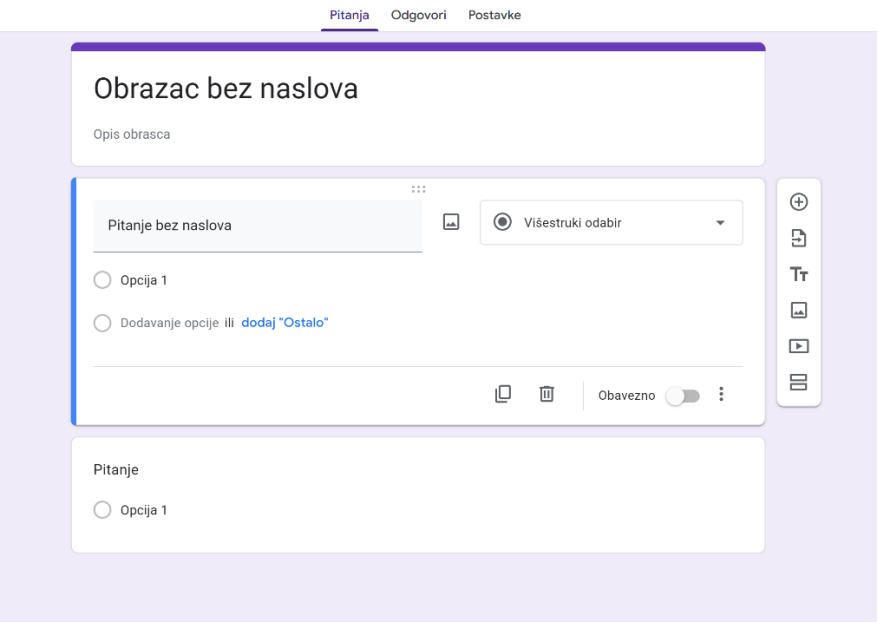
Prikupljanje podataka uz pomoć ankete jednostavan je način za organiziranu evidenciju potrebnih informacija. Stoga, na tržištu postoje razne web aplikacije koje služe upravo toj svrsi. Unatoč velikom broju već postojećih aplikacija, fokus svake aplikacije je samo na jednom problemu što dovodi do potrebe za korištenjem više aplikacija.

Web aplikacija *Scan&Connect* koja je napravljena i obrađena u ovom završnom radu spaja funkcionalnosti više aplikacija i omogućuje korisniku prikupljanje podataka kao i nasumični odabir osoba bez potrebe za korištenjem više aplikacija.

U nastavku su prikazane slične aplikacije koje su dostupne na tržištu i opisane su njihove funkcionalnosti.

2.1. Google Forms

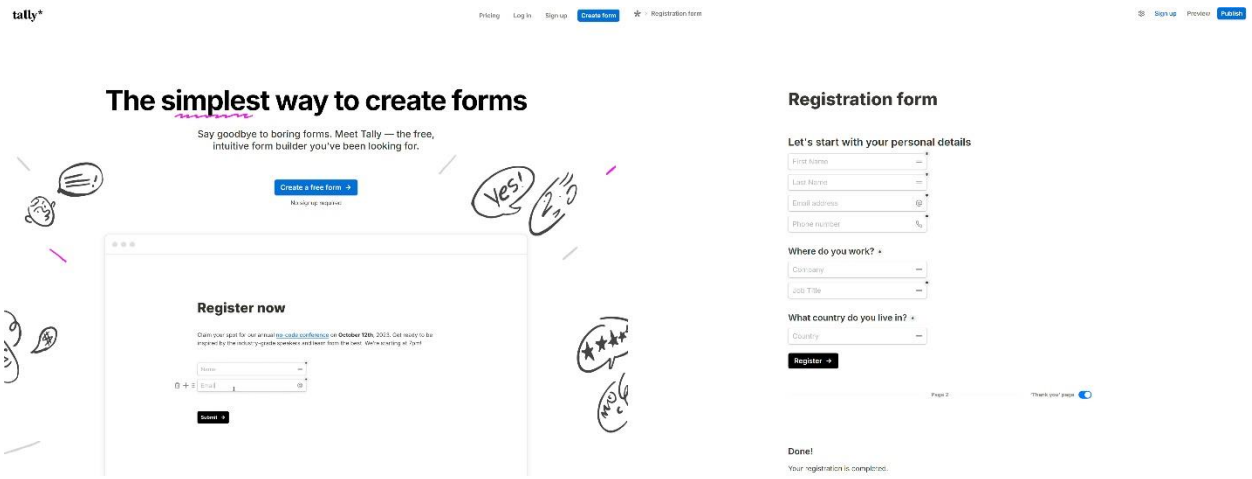
Google Forms [1] omogućuje jednostavnu izradu online obrazaca poput anketa i upitnika. Moguće je prilagoditi boje i fontove kao i dijeliti obrasce putem e-pošte ili putem društvenih medija. *Google Forms* automatski analizira podatke bez dodatnih koraka. Jedini preduvjet za korištenje je Google račun. Na slici 2.1. prikazano je kreiranje novog obrasca u aplikaciji.



Slika 2.1. Kreiranje novog obrasca u aplikaciji *Google Forms* [1]

2.2. Tally

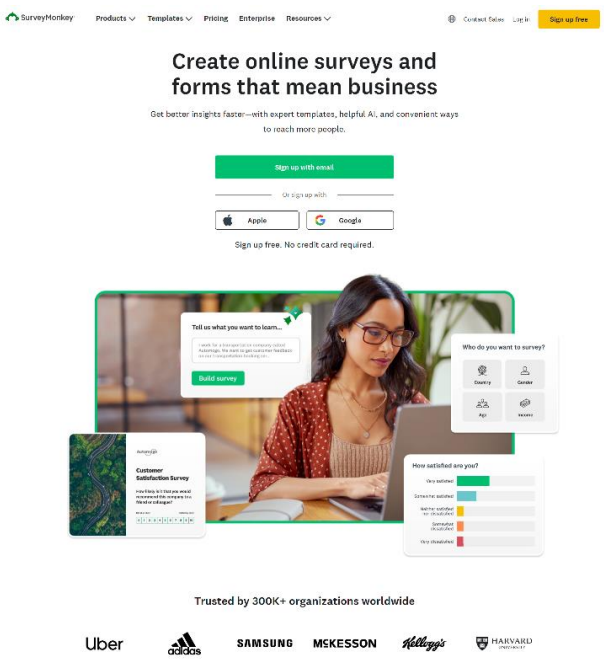
Tally [2] predstavlja alat za kreiranje online obrazaca. Fokus aplikacije je na jednostavnosti uporabe i na dizajnu samih obrazaca. *Tally* nudi mogućnost povezivanja s brojnim alatima za poslovanje i produktivnost kao što su *Notion, Airtable, Slack* i dr. Visoka razina prilagođavanja dizajna obrasca poput odabira boje, fonta, stila kao i dodavanje slika i video zapisa prednosti su aplikacije kao i razlog zbog kojeg korisnici odabiru *Tally*.



Slika 2.2. Korisničko sučelje aplikacije *Tally* [2]

2.3. SurveyMonkey

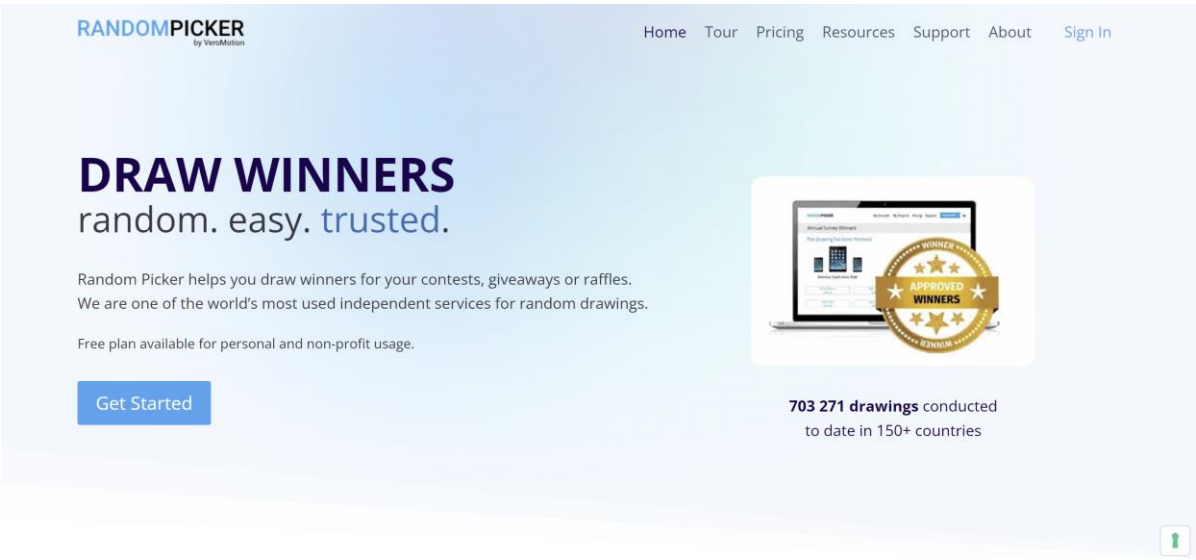
SurveyMonkey [3] još je jedan od alata za stvaranje online obrazaca i anketa. Za razliku od gore navedenih primjera, *SurveyMonkey* nudi naprednije opcije i detaljniju kontrolu nad anketama. Jedna od naprednijih opcija kojom se ova aplikacija izdvaja među konkurencijom je logika grananja koja omogućuje prilagodbu pitanja na temelju prethodnih odgovora. Također, postoji mogućnost ograničavanja pristupa anketa putem lozinke što čini ovaj alat veoma pouzdanim i sigurnim. U svojoj ponudi, *SurveyMonkey* ima mnogo predložaka po raznim kategorijama kako bi kreiranje obrazaca bilo još jednostavnije.



Slika 2.3. Početna stranica aplikacije *SurveyMonkey* [3]

2.4. RandomPicker

RandomPicker [4] omogućuje nasumičan odabir pobjednika iz prethodno unesenih popisa imena, brojeva itd. Koristan je za organizaciju nagradnih igara ili natjecanja. Alat koristi različite mehanizme izvlačenja ovisno o potrebi korisnika. Korisnici imaju mogućnost prilagodbe odabira prema različitim kriterijima kao i mogućnost odabira broja pobjednika. *RandomPicker* osigurava sigurnost i transparentnost pri izvlačenju kako bi postupak bio pošten. Sigurnost potvrđuju certifikatima koji se mogu podijeliti sa sudionicima u izvlačenju.



Slika 2.4. Početna stranica aplikacije *RandomPicker* [4]

3. FUNKCIONALNI ZAHTJEVI WEB APLIKACIJE

U nastavku će biti definirani funkcionalni i nefunkcionalni zahtjevi koje web aplikacija treba ispuniti.

3.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi definiraju što web aplikacija treba raditi, odnosno koje funkcionalnosti aplikacija mora imati kako bi ispunila svoju svrhu.

Funkcionalni zahtjevi web aplikacije:

- Autentifikacija korisnika: korisnici imaju mogućnost prijave pomoću e-mail-a i lozinke koji se predefiniraju u bazi podataka
- Kreiranje događaja: administrator može kreirati novi događaj i odabrati naziv, datume početka i završetka događaja, kao i ciljanu publiku
- Spremanje podataka u bazu podataka: kreiranjem novog događaja kao i popunjavanjem ankete, popunjeni podaci pohranjuju se u bazu podataka
- Pregled i filtriranje svih događaja: administrator ima uvid u sve događaje koje može filtrirati na temelju datuma završetka događaja (svi događaji, završeni događaji, događaji u tijeku, nadolazeći događaji)
- Prikaz detalja o pojedinom događaju: korisnici imaju mogućnost vidjeti detalje o događaju kao i unijeti podatke u kreiranu anketu
- Nasumični odabir pobjednika: administrator može nasumično odabrati pobjednika pomoću opcije *„Odaberi pobjednika“*

3.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi, za razliku od funkcionalnih, opisuju na koji način web aplikacija treba raditi. Fokus je na sigurnosti, održivosti, intuitivnosti i jednostavnosti.

- Sigurnost: podatci o korisnicima sigurno su pohranjeni koristeći *Firebase*
- Upotrebljivost: sučelje treba biti jednostavno, kao i navigacija kroz aplikaciju
- Kompatibilnost: aplikacija mora biti kompatibilna sa svim preglednicima i treba podržavati rad i na mobilnim uređajima

4. KORIŠTENE TEHNOLOGIJE IZRADE I RAZVOJNI ALATI

U ovome poglavlju bit će objašnjene i navedene tehnologije i alati koji su korišteni pri izradi web aplikacije u sklopu ovog završnog rada. Korišteni su prezentacijski jezik HTML, stilski jezik CSS, programski jezik *JavaScript*, uređivač izvornog koda *Visual Studio Code* kao i baza podataka *Firebase*.

4.1. HTML

HTML [5] skraćenica je za *Hyper Text Markup Language.* HTML označni je jezik pomoću kojeg se izrađuju web stranice. Definira strukturu sadržaja koristeći oznake (engl. *tags*) pomoću kojih se opisuju elementi na stranici. HTML nije programski jezik već služi za strukturiranje i prezentaciju informacija.

4.2. CSS

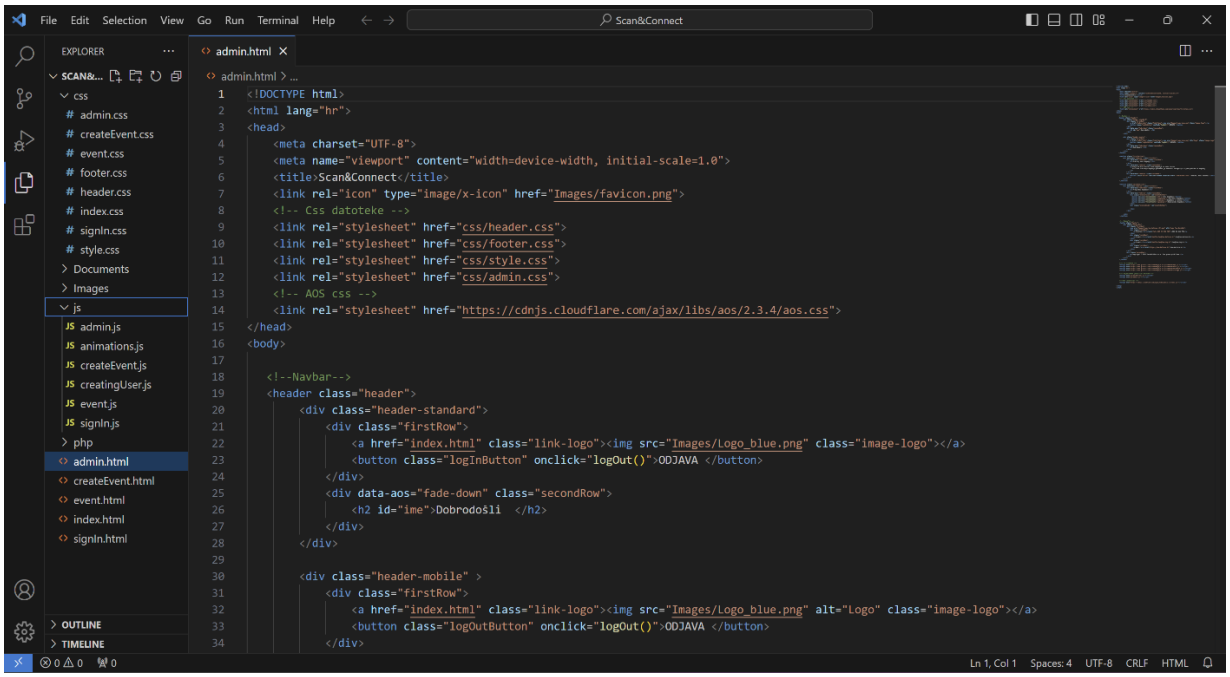
CSS [6] skraćenica je za *Cascading Style Sheets.* CSS stilski je jezik koji se koristi za definiranje HTML elemenata poput boje, fontova, tablica, margina i pozicije elemenata na stranici. Pojavom CSS-a postalo je moguće odvojiti strukturu (HTML) od dizajna, čime HTML kod postaje pregledniji i olakšava se održavanje web stranice.

4.3. JavaScript

JavaScript [7] je skriptni programski jezik za dodavanje dinamičkih funkcionalnosti i interaktivnosti web stranicama. Omogućava dinamičko ponašanje web stranica poput kreiranja animacija, provjere ispravnosti podataka unesenih u formu i sl. *JavaScript* se postavlja u HTML pomoću `<script>` oznaka, a moguće je i kreiranje zasebne vanjske datoteke koja mora imati *.js* ekstenziju.

4.4. Visual Studio Code

Visual Studio Code [8] je uređivač izvornog koda kojega je razvio *Microsoft*. Koristi se za pisanje i uređivanje koda te pruža podršku za razne programske jezike poput C++, *JavaScript,* C# i mnogih drugih. Podržan je na *Windows, Linux i MacOS* operativnim sustavima. *Visual Studio Code* poznat je po velikom broju dodataka, jednostavnom sučelju i mnogim značajkama kao što su automatsko dovršavanje koda, ugrađeni terminal, otkrivanje pogrešaka (engl. *debugging*), ugrađena podrška za Git itd. *VS Code* također nudi i mogućnost prilagodbe izgleda sučelja i razne teme. Na slici 4.1. prikazano je korisničko sučelje aplikacije.



Slika 4.1. Korisničko sučelje aplikacije *Visual Studio Code*

4.5. Firebase

Firebase [9] je *Backend-as-a-Service* (*BaaS*) usluga razvijena od strane Google-a. Vrlo je popularna platforma za razvoj i održavanje web i mobilnih aplikacija.

Glavne usluge koje nudi su:

- ***Firebase Realtime Database* (Baza podataka u realnom vremenu)**

Firebase Realtime Database je NoSQL baza podataka koja osigurava sinkronizaciju podataka u stvarnom vremenu. *Realtime Datebase* koristi JSON (engl. *JavaScript Object Notation*) za sinkronizaciju i pohranu podataka između korisnika i uređaja omogućujući pristup, pohranu i upravljanje podatcima u stvarnom vremenu na jednostavan način. JSON ima vrlo jednostavnu strukturu zahvaljujući kojoj je programerima olakšano pohranjivanje, obrada i prijenos podataka.

- ***Authentication* (Autentifikacija)**

Firebase Authentication olakšava autentifikaciju i nudi različite metode prijave poput prijave putem e-mail-a i lozinke i prijave putem *Google*, *Facebook*, *Twitter* ili *GitHub* računa.

5. PROGRAMSKO RJEŠENJE

Unutar ovoga poglavlja objašnjeno je programsko rješenje web aplikacije *Scan&Connect*. Detaljno je objašnjen rad s bazom podataka, kao i ostale funkcionalnosti i logika ove aplikacije, poput kreiranja događaja, prikaza događaja, nasumičnog odabira pobjednika i dr.

5.1. Baza podataka

Na slici 5.1. u nastavku prikazan je programski kod konfiguracije *Firebase* baze podataka. Konkretno *firebaseConfig* objekt sadrži sve ključne podatke potrebne za povezivanje aplikacije s *Firebase* projektom. Od navedenih atributa jedni od bitnijih su ključ koji je specifičan za svaki *Firebase* projekt i omogućava autentifikaciju aplikacije prema *Firebase* API-ju ,te *databaseURL* adresa koja povezuje aplikaciju s bazom podataka. Vidljivo je kako se baza nalazi na *scanconnect-ebede-default-rtdb.europe-west1.firebasedatabase.app*.

```
var firebaseConfig = {
    apiKey: "AIzaSyA5DJHbYfAQYPfwZLRcdcElI9y1-tmRgAE",
    authDomain: "scanconnect-ebede.firebaseapp.com",
    databaseURL: "https://scanconnect-ebede-default-rtdb.europe-west1.firebasedatabase.app",
    projectId: "scanconnect-ebede",
    storageBucket: "scanconnect-ebede.appspot.com",
    messagingSenderId: "972755645242",
    appId: "1:972755645242:web:42067b11d9fa4b4bc69603",
    measurementId: "G-1M4XDZHSGS"
};
```

Slika 5.1. Konfiguracija *Firebase* baze podataka

Na slici 5.2. prikazan je dio programskog koda u kojemu se koristi funkcija *initializeApp* na prethodno definiranom *firebaseConfig* objektu kako bi se *Firebase* inicijalizirao unutar aplikacije. To je ključno za pristupanje samoj bazi podataka, autentifikaciji i pohrani. Iduća linija koda *var database = firebase.database()* povezuje varijablu *database* s *Firebase* bazom podataka u stvarnom vremenu. To omogućava aplikaciji čitanje, te pisanje podataka u stvarnom vremenu koristeći predefinirane *Firebase* funkcije.

```
firebase.initializeApp(firebaseConfig);
var database = firebase.database();
```

Slika 5.2.Inicijalizacija *Firebase* baze podataka

Na slici 5.3. prikazana je struktura, odnosno organizacija podataka spremljenih u bazu. Budući da se radi o NoSQL strukturi, podaci su organizirani u obliku stabla. Dvije su ključne grane unutar ove baze, to su *events* i *registrirani korisnici*. *Events* predstavlja granu u kojoj se pohranjuju podaci o svim kreiranim događajima, dok se unutar grane *registrirani korisnici,* konkretno u ovom slučaju, nalaze podaci o administratoru koji događaje i kreira. Takav slučaj bio bi i za više upisanih administratora, odnosno korisnika koji utječu na stvaranje događaja.

Slika 5.3. Struktura podataka unutar *Firebase* baze podataka

5.2. Administrator

Na slici 5.4. prikazana je funkcija za kreiranje korisnika unutar *Firebase* baze podataka. Objekt *userData* sadrži osnovne informacije o korisniku: ime, e-mail i lozinku. Funkcija *database.ref('registriranikorisnici').push(userData)* upisuje korisničke podatke u *Firebase* bazu podataka pod granu *registrirani korisnici* objašnjenu u prethodnom potpoglavlju. Funkcija *createUser()* se poziva automatski nakon što se cijeli HTML dokument učita pomoću *document.addEventListener()*. To na koncu omogućava da se podaci ne mogu upisati prije nego što se stranica učita u cijelosti.

```
function createUser() {
    var userData = {
        ime: "admin",
        email: "admin@teo-belisce.hr",
        lozinka: "123456"
    };

    database.ref('registriranikorisnici').push(userData, function(error) {
        if (error) {
            console.error("Error saving data:", error);
        } else {
            console.log("Data successfully saved.");
        }
    });
}
document.addEventListener('DOMContentLoaded', function() {
    createUser();
});
```

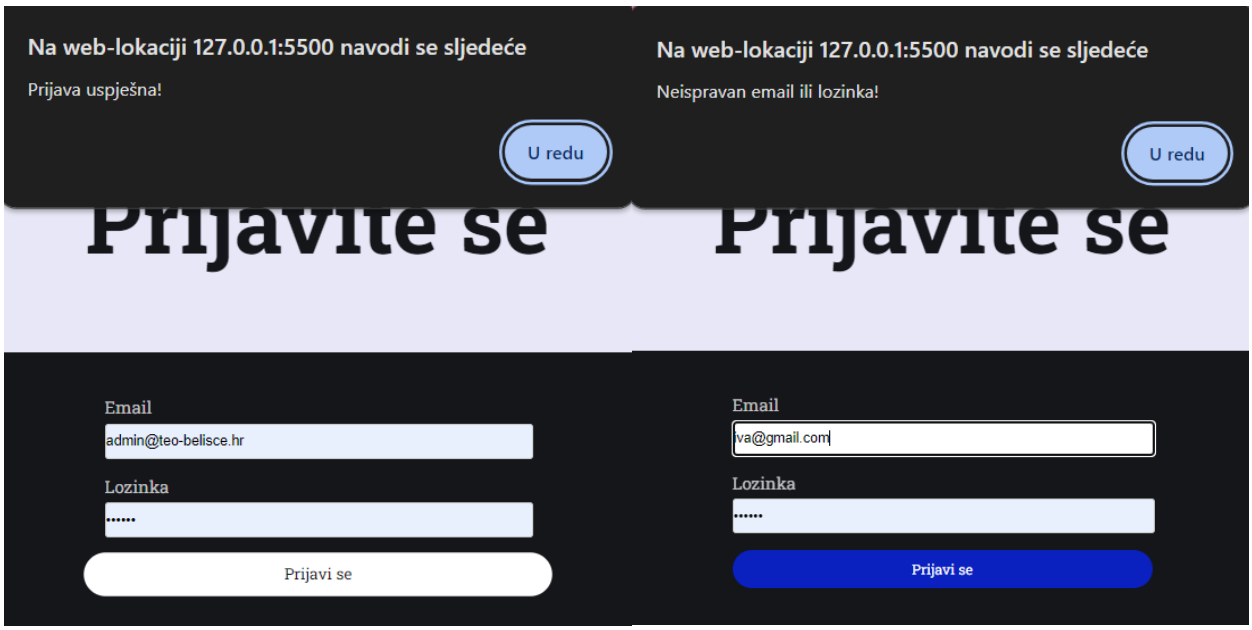
Slika 5.4. Funkcija za kreiranje korisnika

Iduća funkcija prikazana na slici 5.5. prikazuje provjeru korisničkih podataka. Funkcija koristi *database.ref('registriranikorisnici').once('value', function(snapshot)* za dohvaćanje svih korisničkih podataka iz *Firebase* baze podataka. Koristeći snapshot funkciju, prolazi se kroz sve korisnike pomoću *snapshot.forEach*, radi provjeravanja odgovara li uneseni email i lozinka nekom od upisanih korisnika. Ako je pronađen korisnik čiji su email i lozinka odgovarajući, postavljaju se varijable *userExists, userName, i userId* kako bi se prepoznao prijavljeni korisnik. Ako korisnik postoji, kreira se kolačić s podacima o prijavi (koristeći *document.cookie*), a korisnik se preusmjerava na administratorsku stranicu (*window.location.href = 'admin.html'*).

```
function checkCredentials(email, password) {
    database.ref('registriranikorisnici').once('value', function(snapshot) {

        let userExists = false;
        let userName = '';
        let userId = '';

        snapshot.forEach(function(childSnapshot) {
            const userData = childSnapshot.val();
            if (userData.email === email && userData.lozinka === password) {
                userExists = true;
                userName = userData.ime;
                userId = childSnapshot.key;
            }
        });

        if (userExists) {
            alert("Prijava uspješna!");
            let now = new Date();
            now.setTime(now.getTime() + (24 * 60 * 60 * 1000));
            let expires = "expires=" + now.toUTCString();

            document.cookie = "signedUser=true; path=/"; +expires;
            document.cookie = "userId=" + userId + "; path=/"; +expires;
            document.cookie = "userName=" + encodeURIComponent(userName) + "; path=/"; +expires;
            window.location.href = 'admin.html';
        } else {
            alert("Neispravan email ili lozinka!");
        }
    });
}
```

Slika 5.5. Funkcija za provjeru korisničkih podataka

Stvarni prikaz funkcije sa slike 5.5. prikazan je na slici 5.6. gdje su vidljiva oba konkretna primjera uspješne, odnosno neuspješne prijave nakon unosa korisničkih podataka. Prijava je uspješna za već predefiniranog, pronađenog korisnika, u ovom slučaju administratora.



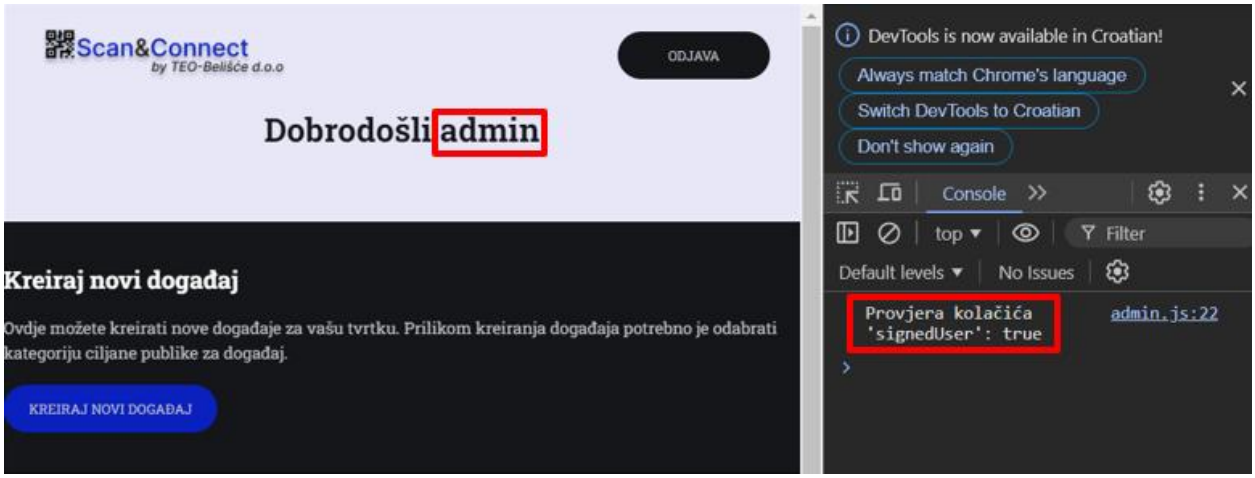
Slika 5.6. Provjera uspješnosti prijave

Kod na slici 5.7. osigurava da samo prijavljeni korisnici mogu pristupiti određenim dijelovima stranice, dok se neprijavljeni korisnici šalju na stranicu za prijavu. Funkcija *getCookie('signedUser')* dohvaća kolačić koji označava je li korisnik prijavljen. Ako kolačić ne postoji, korisnik se preusmjerava na stranicu za prijavu (*index.html*). Ako je korisnik prijavljen, dohvaća se ime korisnika pomoću kolačića *getCookie('userName')*. Ako je korisnik prijavljen, njegovo ime se prikazuje na stranici s porukom dobrodošlice *"Dobrodošli, [ime korisnika]"*.

```
document.addEventListener('DOMContentLoaded', function() {
    let loggedIn = getCookie("signedUser");
    let userName = getCookie("userName");

    console.log("Provjera kolačića 'signedUser':", loggedIn);

    if (!loggedIn) {
        window.location.href = 'index.html';
    } else {
        if (userName) {
            document.getElementById('ime').textContent = `Dobrodošli ${decodeURIComponent(userName)}`;
        } else {
            console.log("Ime korisnika nije pronađeno.");
        }
    }
    fetchEvents();
});
```

Slika 5.7. Provjera kolačića za prijavljene korisnike

Slika 5.8. prikazuje kako izgleda administratorska stranica nakon uspješne prijave. Na vrhu stranice prikazuje se poruka dobrodošlice, *"Dobrodošli, admin"*, što znači da su podaci o prijavi uspješno dohvaćeni iz kolačića o kojem je bilo riječi u prethodnom odlomku. Također, u konzoli je vidljiv ispis *"Provjera kolačića 'signedUser': true"*, što potvrđuje da je korisnik uspješno autentificiran što je i bio cilj postavljanja i upravljanja kolačićima.



Slika 5.8. Prikaz administratorske stranice nakon prijave

5.3. Događaj

5.3.1. Prikaz svih događaja

Funkcija pod nazivom *fetchEvents()* prikazana na programskom kodu na slici 5.9. koristi *Firebase* bazu podataka kako bi dohvatila sve događaje pohranjene u grani *events*. U funkciji se koristi *database.ref('events').once('value', ...)* za dohvaćanje svih događaja iz baze podataka u jednom trenutku. Nakon dohvaćanja događaja, funkcija *displayEvents(events, 'all')* zove se s parametrom *'all'*, što znači da će svi događaji biti prikazani bez filtera.

```
function fetchEvents() {
    database.ref('events').once('value', function(snapshot) {
        var events = snapshot.val();
        displayEvents(events, 'all');
    });
}
```

Slika 5.9. Funkcija za dohvaćanje svih događaja

5.3.2. Filtriranje postojećih događaja

Funkcija *filterEvents(filter)* koristi istu logiku kao *fetchEvents* s isječka programskog koda na slici 5.9., ali uz mogućnost dodavanja filtra za prikaz događaja. Ponovno se koristi *database.ref('events').once('value', ...)* za dohvaćanje svih događaja, no nakon toga se poziva funkcija *displayEvents(events, filter)* s dodatnim argumentom *filter* koji kontrolira koje događaje treba prikazati (npr. samo nadolazeće, završene ili trenutne događaje).

```
function filterEvents(filter) {
    database.ref('events').once('value', function(snapshot) {
        var events = snapshot.val();
        displayEvents(events, filter);
    });
}
```

Slika 5.10. Funkcija za filtriranje postojećih događaja

Slika 5.11. prikazuje petlju koja prolazi kroz sve dohvaćene događaje i primjenjuje filter za određivanje statusa svakog događaja. Korištena je petlja *for (var key in events)* koja prolazi kroz sve događaje po njihovim jedinstvenim ključevima u bazi. Ako je datum završetka događaja manji od trenutnog datuma *(event.zavrsava < now)*, status se postavlja na "Završen". Ako je događaj u tijeku, tj. datum početka je prošao, a događaj još traje *(event.pocinje <= now && event.zavrsava >= now)*, status se postavlja na "U tijeku". Ako događaj još nije počeo, status se postavlja na "Nadolazeći". Na kraju se koriste uvjeti za filtriranje događaja prema statusu (npr. *completed, ongoing, upcoming, all*), ovisno o korisničkom unosu.

```
for (var key in events) {
    if (events.hasOwnProperty(key)) {
        var event = events[key];
        var status = '';

        if (event.zavrsava < now) {
            status = 'Završen';
        } else if (event.pocinje <= now && event.zavrsava >= now) {
            status = 'U tijeku';
        } else {
            status = 'Nadolazeći';
        }

        if ((filter === 'completed' && status === 'Završen') ||
            (filter === 'ongoing' && status === 'U tijeku') ||
            (filter === 'upcoming' && status === 'Nadolazeći') ||
            (filter === 'all')) {
```

Slika 5.11. Isječak koda u kojem je napisana logika za filtriranje i status događaja

5.3.3. Prikaz detalja pojedinog događaja

Funkcija *fetchEventDetails(eventId)* koristi se za dohvaćanje i prikaz detalja pojedinog događaja pomoću njegovog ID-a. Koristi se *database.ref('events/' + eventId).once('value', ...)* kako bi se dohvatili podaci za događaj s određenim ID-em. Različiti elementi stranice se ažuriraju pomoću podataka o događaju (npr. naziv događaja, datum početka i završetka, ciljana skupina). Funkcija *showRelevantForm(event.ciljanaSkupina)* koristi se za prikazivanje odgovarajućih formi popunjavanja baziranih na ciljanoj skupini događaja. Generira se QR kod za događaj pomoću funkcije *generateQRCode(eventId)* o kojemu će više riječi biti u nastavku.

```
function fetchEventDetails(eventId) {
    database.ref('events/' + eventId).once('value', function(snapshot) {
        var event = snapshot.val();
        if (event) {
            document.getElementById('eventTitle').textContent = event.naziv;
            document.getElementById('eventTitleMobile').textContent = event.naziv;

            document.getElementById('startDate').textContent = formatDate(event.pocinje);
            document.getElementById('endDate').textContent = formatDate(event.zavrsava);
            document.getElementById('targetGroup').textContent = event.ciljanaSkupina;
            showRelevantForm(event.ciljanaSkupina);
            generateQRCode(eventId);

            checkEventStatus(event.zavrsava);
        } else {
            console.log("Događaj nije pronađen.");
        }
    });
}
```

Slika 5.12. Dohvaćanje i prikaz detalja pojedinog događaja

Funkcija *generateQRCode(eventId)* prikazana na slici 5.13. generira QR kod za određeni događaj na temelju njegovog ID-a. URL se kreira pomoću *window.location.origin + "/event.html?eventId=" + eventId*, što omogućava korisnicima da skeniraju QR kod i direktno pristupe stranici događaja. Funkcija *QRCode.toCanvas()* generira QR kod u *canvas* elementu na stranici. Ako se dogodi greška prilikom generiranja, ona se ispisuje se u konzoli.

```
function generateQRCode(eventId) {
    var qrCodeUrl = window.location.origin + "/event.html?eventId=" + eventId;
    var qrCodeCanvas = document.getElementById('qrcodeCanvas');

    QRCode.toCanvas(qrCodeCanvas, qrCodeUrl, function (error) {
        if (error) console.error(error);
        console.log('QR kod generiran!');
    });
}
```

Slika 5.13. Prikaz funkcije generiranja QR koda

Nadalje, potrebno je provjeriti i status događaja što se provjerava funkcijom na slici 5.14. Funkcija *checkEventStatus(endDate)* koristi se za provjeru je li događaj već završio. Ako je trenutni datum (*now*) veći od datuma završetka događaja (*eventEndDate*), prikazuje se poruka da je događaj završio i da obrazac više nije dostupan. Element na stranici ažurira se porukom *"Nažalost, događaj je završio i obrazac više nije dostupan."*

```
function checkEventStatus(endDate) {
    let now = new Date();
    let eventEndDate = new Date(endDate);

    if (now > eventEndDate) {
        document.querySelector('.eventForm').innerHTML = '<p>Nažalost, događaj je završio i obrazac više nije dostupan.</p>';
    }
}
```

Slika 5.14. Prikaz statusa događaja

5.3.4. Kreiranje novog događaja

Slika 5.15. prikazuje kod koji omogućuje korisnicima kreiranje novog događaja putem forme. Funkcija *event.preventDefault()* osigurava da se obrazac ne pošalje automatski prilikom klika na gumb, već se izvršava *JavaScript* kod. Uneseni podaci (naziv događaja, ciljana skupina, datum početka i završetka) dohvaćaju se iz pripadajućih ulaznih polja forme pomoću *document.getElementById*. Ukoliko su sva polja ispravno popunjena, kreira se objekt *newEvent*, koji sadrži sve potrebne podatke o događaju. Zatim se taj objekt upisuje u *Firebase* bazu podataka pomoću *database.ref('events').push(newEvent)*. Ako se dogodi greška prilikom kreiranja događaja, prikazuje se poruka u konzoli. Ako je događaj uspješno kreiran, korisniku se prikazuje poruka *„Događaj uspješno kreiran!“* i forma se resetira.

```
document.getElementById('createEventForm').addEventListener('submit', function(event) {
    event.preventDefault();

    var eventName = document.getElementById('eventName').value;
    var ciljanaSkupina = document.getElementById('dropdownMenu').value;
    var eventStart = document.getElementById('eventStart').value;
    var eventEnd = document.getElementById('eventEnd').value;

    if (eventName && ciljanaSkupina && eventStart && eventEnd) {
        var newEvent = {
            naziv: eventName,
            ciljanaSkupina: ciljanaSkupina,
            pocinje: eventStart,
            zavrsava: eventEnd
        };
        database.ref('events').push(newEvent, function(error) {
            if (error) {
                console.error("Error saving event:", error);
            } else {
                alert("Događaj uspješno kreiran!");
                document.getElementById('createEventForm').reset();
            }
        });
    } else {
        alert("Molimo popunite sva polja!");
    }
});
```

Slika 5.15. Kreiranje novog događaja pomoću forme

U nastavku je prikazana pripadajuća forma za kreiranje događaja u HTML strukturi. Slika 5.16. daje uvid u polja za unos koja je potrebno ispuniti prilikom popunjavanja forme. Unosi se naziv događaja, ciljana publika, te datum početka i završetka novog događaja. Također sadrži gumb za slanje forme koji poziva funkciju za kreiranje događaja nakon što su sva polja ispravno popunjena.

```
<form id="createEventForm">
    <div class="formRow">
        <div class="formColumn">
            <label for="eventName">Naziv događaja</label>
            <input type="text" id="eventName" name="eventName" required>
        </div>
        <div class="formColumn">
            <label for="dropdownMenu">Ciljana publika</label>
            <select id="dropdownMenu" name="options">
                <option value="studenti">Studenti</option>
                <option value="tvrtka">Tvrtka</option>
            </select>
        </div>
    </div>
    <div class="formRow">
        <div class="formColumn">
            <label for="eventStart" >Datum početka događaja</label>
            <input type="date" id="eventStart" name="eventStart" required>
        </div>
        <div class="formColumn">
            <label for="eventEnd" >Datum završetka događaja</label>
            <input type="date" id="eventEnd" name="eventEnd" required>
        </div>
    </div>
    <button class="createEventButton" type="submit">Kreiraj događaj</button>
</form>
```

Slika 5.16. HTML forma za kreiranje događaja

5.4. Odabir pobjednika događaja

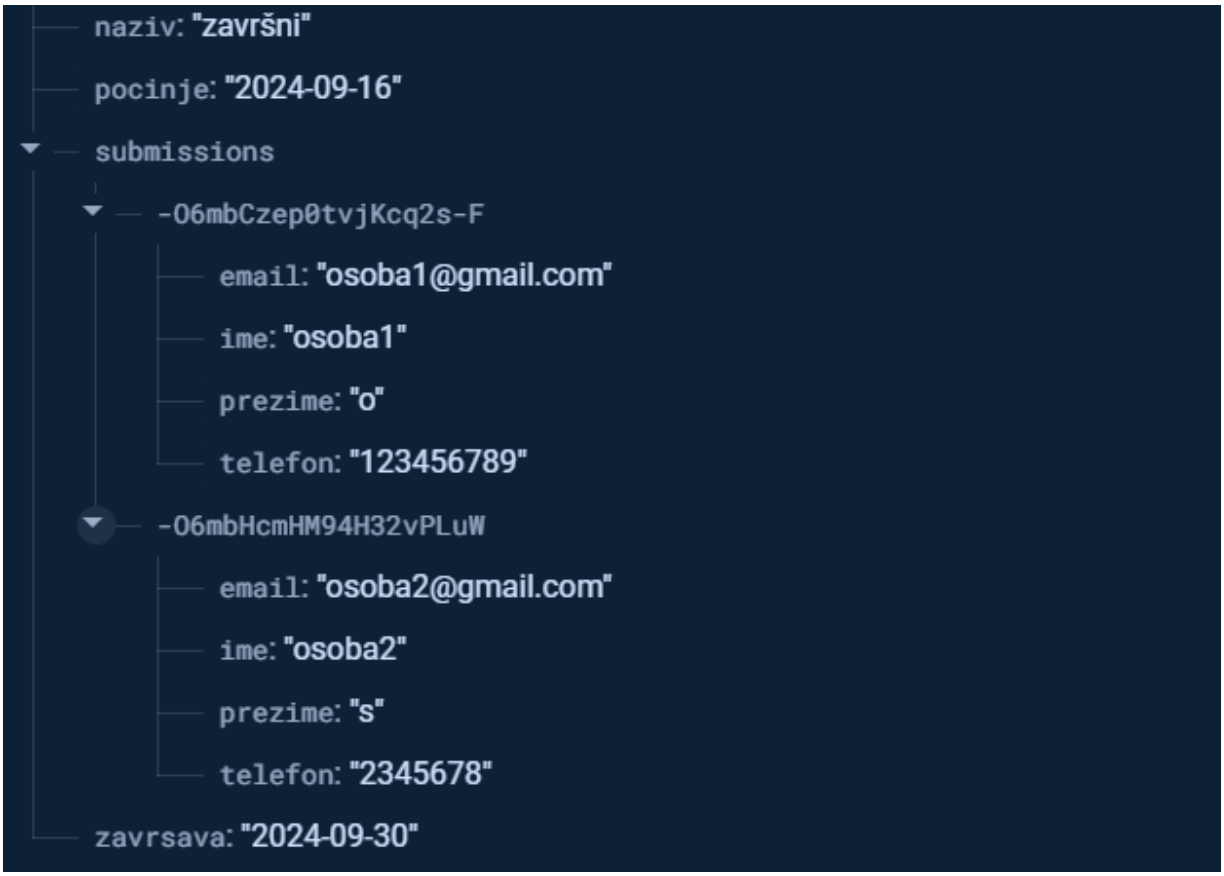
Slika 5.17. prikazana u nastavku prikazuje *JavaScript* funkciju *pickWinner(eventId)*, koja omogućuje slučajan odabir pobjednika među prijavljenim sudionicima za određeni događaj na temelju unosa igrača pohranjenih u *Firebase* bazi podataka. Prva stavka funkcije koristi izraz *database.ref('events/' + eventId + '/submissions').once('value', function(snapshot)* za dohvaćanje svih prijava povezanih s događajem čiji je ID *eventId*. Rezultat dohvaćanja prijava sprema se u varijablu *submissions*, koja predstavlja sve prijavljene sudionike za taj događaj. Ako varijabla *submissions* nema podataka (tj. ako nema prijavljenih sudionika), prikazuje se poruka *"Nema unešenih osoba"* u HTML elementu s ID-em *winnerName*, a funkcija se prekida. Ako postoje prijave, petlja *for (var key in submissions)* prolazi kroz sve prijave i pohranjuje svaku prijavu u niz *submissionsArray* za lakše rukovanje. Ako niz prijava (*submissionsArray*) sadrži barem jednu

prijavu, nasumično se odabire pobjednik. Ime nasumično odabranog pobjednika prikazuje se u HTML elementu s ID-em *winnerName*, uz poruku *„Pobjednik: [ime_pobjednika]“*. Ako nema prijava u sustavu, ponovno se prikazuje poruka „*Nema unešenih osoba*“.

```
function pickWinner(eventId) {
    database.ref('events/' + eventId + '/submissions').once('value', function(snapshot) {
        var submissions = snapshot.val();
        var submissionsArray = [];

        if (!submissions) {
            document.getElementById('winnerName').textContent = 'Nema unešenih osoba';
            return;
        }

        for (var key in submissions) {
            if (submissions.hasOwnProperty(key)) {
                submissionsArray.push(submissions[key]);
            }
        }

        if (submissionsArray.length > 0) {
            var randomIndex = Math.floor(Math.random() * submissionsArray.length);
            var winner = submissionsArray[randomIndex];
            var winnerName = winner.ime ? `${winner.ime} ${winner.prezime}` : winner.nazivTvrtke;
            document.getElementById('winnerName').textContent = `Pobjednik: ${winnerName}`;
        } else {
            document.getElementById('winnerName').textContent = 'Nema unešenih osoba';
        }
    });
}
```

Slika 5.17. Funkcija za izbor pobjednika

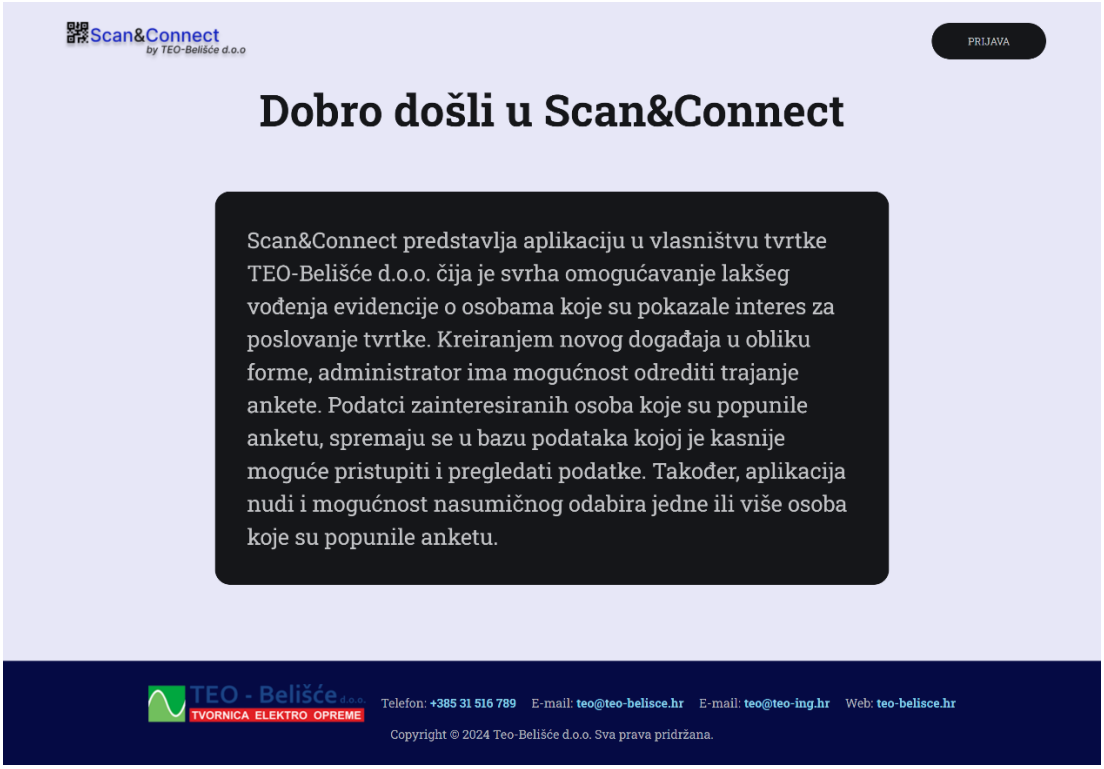
U nastavku prikazana slika 5.18. prikazuje strukturu podataka u *Firebase* bazi podataka, gdje su pohranjene prijave (*submissions*) za određeni događaj. Konkretan primjer odrađen je na događaju pod nazivom "*završni*" koji počinje 16. rujna 2024. godine, a pod njim se nalaze sve prijave sudionika. Unutar grane *submissions*, svaki sudionik ima svoj jedinstveni ID koji generira *Firebase* (konkretno u ovom slučaju -*O6mbCzep0tvjKcq2s-F*). Svaka prijava sadrži podatke da se omogući identificiranje sudionika, što je ključno za nasumično odabiranje pobjednika. Prijave se pohranjuju pod granu *submissions*, što omogućava lako upravljanje podacima.



Slika 5.18. Prikaz strukture događaja s naglaskom na prijave igrača

6. IZGLED I RAD APLIKACIJE

U nastavku će biti prikazan izgled aplikacije i detaljan opis načina korištenja web aplikacije. Kada korisnik pristupi aplikaciji putem web preglednika, prikazat će se početna stranica aplikacije koja je prikazana na slici 6.1.. Na vrhu stranice, u lijevom kutu nalazi se logo aplikacije, dok je u desnom kutu gumb za prijavu. U sredini se nalazi tekst koji objašnjava svrhu i funkcionalnost same aplikacije. U podnožju (eng. *footer*) nalazi se logo tvrtke *TEO-Belišće d.o.o.* [10] za koju je aplikacija *Scan&Connect* primarno napravljena, kao i podaci o tvrtki poput web stranice, e-mail adrese i broja telefona.

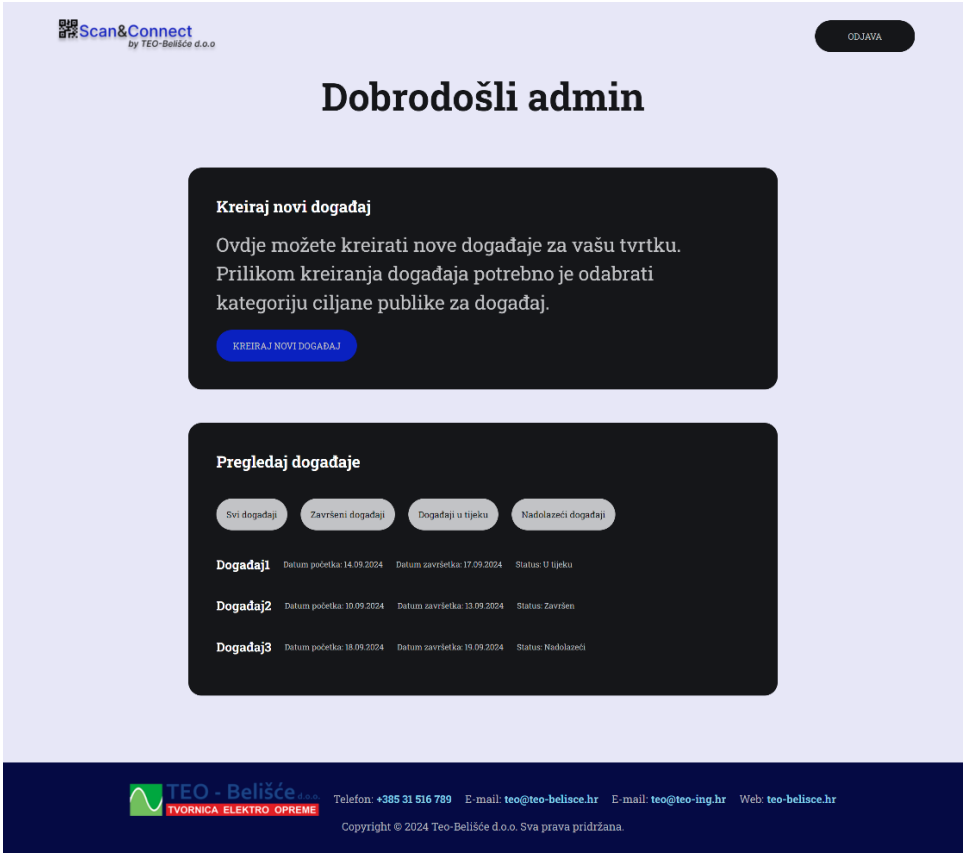


Slika 6.1. Početni zaslon web aplikacije

Pritiskom gumba „*PRIJAVA*“, korisniku se otvara zaslon za prijavu, gdje ima mogućnost unijeti predefinirane podatke: e-mail i lozinku. Zaslon za prijavu prikazan je na slici 6.2..

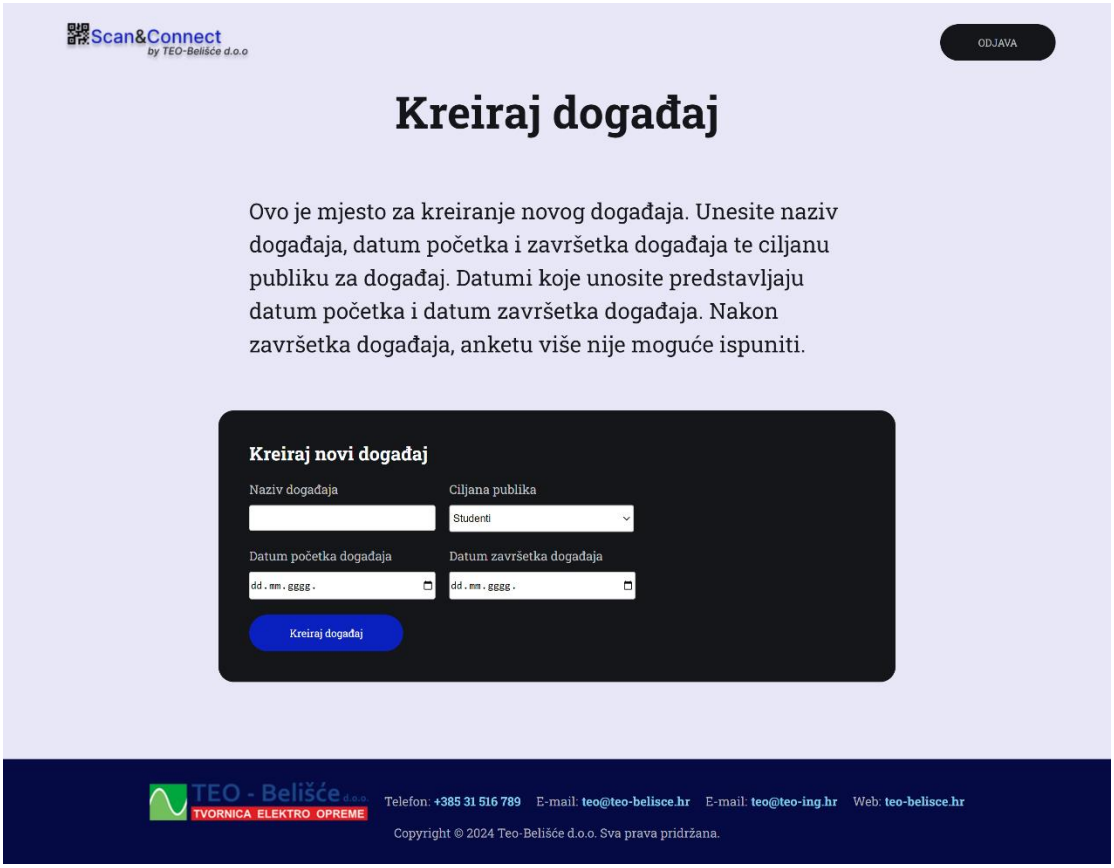
Slika 6.2 Zaslon za prijavu

Nakon unesenih podataka i pritiska na gumb „*Prijavi se*“, korisnik, odnosno administrator prelazi na zaslon gdje su prikazani svi događaji. Administrator ima mogućnost kreiranja novog događaja, kao i uvid u sve događaje. Omogućeno je i filtriranje događaja prema datumu završetka, koje je prikazano na zaslonu za upravljanje svim događajima (Slika 6.3.).



Slika 6.3. Zaslon za upravljanje svim događajima

Odabirom gumba „*KREIRAJ NOVI DOGAĐAJ*“, otvara se novi zaslon, zaslon za kreiranje događaja (Slika 6.4.). Administratoru je omogućeno kreiranje novog događaja uz odabir naziva događaja, ciljane publike pomoću padajućeg izbornika (engl. *dropdown menu*) kao i datuma početka i završetka događaja odabirom datuma na kalendaru.

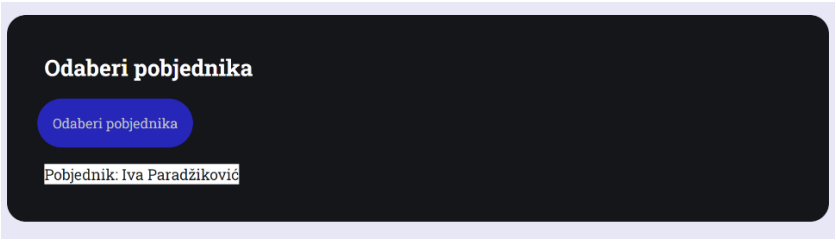


Slika 6.4. Zaslon za kreiranje novog događaja

Kreiranjem novog događaja, generira se QR kod te se otvara zaslon kreiranog događaja (Slika 6.5.). Pri vrhu, prikazuju se naziv događaja, odabrani datum početka događaja, datum završetka događaja kao i ciljana skupina, odnosno publika. Ispod toga nalazi se forma kojoj zainteresirani kandidati pristupaju skeniranjem QR koda. Kada kandidati ispune formu vlastitim podatcima, odabiru gumb „*Pošalji*“ i time ostvaraju pravo sudjelovanja u nagradnoj igri.

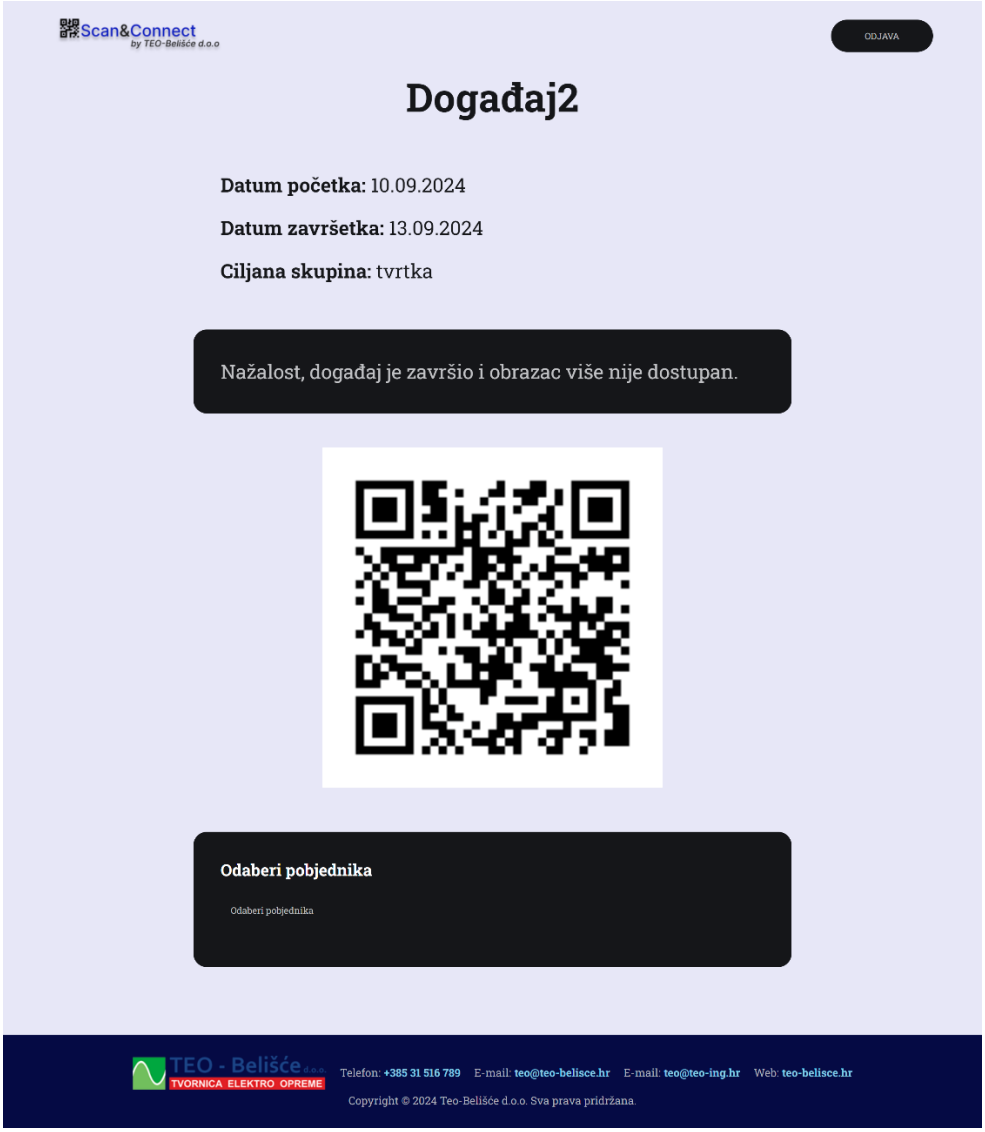
Slika 6.5. Zaslon kreiranog događaja u tijeku

Administrator pritiskom na gumb „*Odaberi pobjednika*“ ima mogućnost nasumično izabrati pobjednika, odnosno jednog ili više kandidata koji su ispunili formu i unijeti svoje vlastite podatke. Odabir pobjednika prikazan je na slici 6.6.



Slika 6.6. Odabir pobjednika

Ukoliko se pokuša pristupiti događaju koji je završio, odnosno događaju čiji je datum završetka prethodio datumu pristupa, prikazat će se zaslon završenog događaja (Slika 6.7.). Na tome će se zaslonu umjesto forme za ispunjavanje prikazati poruka „*Nažalost, događaj je završio i obrazac više nije dostupan.*“ i formu se neće moći ispuniti. Administratoru je na ovome zaslonu, kao i na prethodnim zaslonima, omogućena odjava iz aplikacije pritiskom na gumb „*ODJAVA*“



Slika 6.7. Zaslon završenog događaja

7. ZAKLJUČAK

U ovome završnome radu objašnjena je i razvijena web aplikacija *Scan&Connect* koja omogućuje prikupljanje informacija o posjetiteljima sajamskog štanda i nasumični odabir dobitnika nagrada. Cilj aplikacije bio je unaprijediti interakciju između tvrtki i potencijalnih zaposlenika lakšom evidencijom podataka. Implementacijom *Firebase Realtime Datebase* (baze podataka u stvarnom vremenu) omogućena je pohrana podataka, a pomoću HTML-a, CSS-a i *JavaScript*-a izrađeno je korisničko sučelje koje odgovara potrebama administratora.

Kreiranje događaja, pregled svih događaja, pregled događaja u tijeku, nasumični odabir pobjednika i generiranje QR koda za pristup anketi, funkcionalnosti su ove web aplikacije. Aplikacija ispunjava zadane funkcionalne i nefunkcionalne zahtjeve čime se postiže jednostavno upravljanje događajima, kao i nagrađivanje osoba.

Aplikacija *Scan&Connect* prikazuje važnost razvoja web aplikacija u poslovnome svijetu. Razvoj web aplikacija poput aplikacije *Scan&Connect* uvelike može olakšati komunikaciju i automatizaciju zadataka koji inače zahtijevaju ručno upravljanje. Moguća je nadogradnja aplikacije i prilagodba funkcionalnosti po potrebi.

LITERATURA

[1] Google Forms, https://www.google.com/forms/about/ [12.8.2024.]

[2] Tally, https://tally.so/ [12.8.2024.]

[3] SurveyMonkey, https://www.surveymonkey.com/ [12.8.2024.]

[4] RandomPicker, https://www.randompicker.com/[12.8.2024.]

[5]MDN Web Docs: HTML basics, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics [11.9.2024.]

[6]MDN Web Docs: CSS basics, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics [11.9.2024.]

[7]MDN Web Docs: JavaScript basics, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics [11.9.2024.]

[8] Visual Studio Code, Visual Studio Code - Code Editing. Redefined [11.9.2024.]

[9] Firebase, Firebase | Google's Mobile and Web App Development Platform [12.9.2024.]

[10] TEO-Belišće d.o.o., https://www.teo-belisce.hr/ [14.9.2024.]

SAŽETAK

U ovom završnom radu izrađena je web aplikacija *Scan&Connect* s ciljem olakšavanja prikupljanja podataka o posjetiteljima sajamskih štandova i nasumičnog odabira pobjednika. Glavni problem kojeg aplikacija rješava je potreba za jednostavnim načinom prikupljanja podataka o potencijalnim zaposlenicima i suradnicima tijekom poslovnih događaja. Aplikacija administratoru omogućuje kreiranje događaja, prikupljanje podataka putem ankete, pohranu podataka u bazu podataka i nasumični odabir dobitnika. Korištenje tehnologija poput HTML-a, CSS-a, *JavaScript*-a i *Firebase* baze podataka osigurana je funkcionalnost aplikacije.

Ključne riječi: anketa, baza podataka, događaj, nasumični odabir, web aplikacija

ABSTRACT

A web application for data collection and random rewarding

This final paper presents the development of web application Scan&Connect with the goal of facilitating data collection about visitors to trade fair stands and randomly selecting winners. The main problem that the application addresses is the need for a simple way to collect data about potential employees and collaborators during business events. The application allows the administrator to create events, collect data through a survey, store the data in a database, and randomly select winners. The use of technologies such as HTML, CSS, JavaScript, and the Firebase database ensures the functionality of the application.

Keywords: database, event, random selection, survey, web application