

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Departman za industrijsko inženjerstvo i menadžment
Predmet: Eksploatacija, održavanje i nadogradnja informacionih sistema



Idejno rešenje razvoja i implementacije veb aplikacije za podršku poslovanja prodaje karata za sportske događaje

— Projektni zadatak —

Student:
Isidora Japundža IT 34/2020

Predmetni profesor:
dr Teodora Vučković

Asistent:
Sara Kijanović

Novi Sad, jul 2024.

Sadržaj

1.	Opis realnog sistema	3
1.1	Cilj projekta.....	3
2.	Korišćene tehnologije.....	3
2.1	Kreiranje UML dijagrama	4
2.2	Kreiranje baze podataka	4
2.3	Kreiranje <i>backend</i> dela aplikacije.....	5
2.4	Kreiranje sigurnosnog dela aplikacije	5
2.5	Kreiranje <i>frontend</i> dela aplikacije.....	5
2.6	Korišćenje <i>Stripe</i> -a.....	6
2.7	Korišćenje <i>Webhook</i> -a	7
3	UML dijagrami	9
3.1	Dijagram slučajeva upotrebe	9
3.2	Dijagram klasa	16
3.3	Dijagram sekvenci	17
4.	Baza podataka	19
4.1	Data Definition Language – DDL	20
4.2	Data Modeling Language – DML	21
4.3	Triger	21
5.	Opis predloženog rešenja.....	24
6.	Zaključak	29

1. Opis realnog sistema

Prvi korak izrade projektnog zadatka podrazumeva jasno definisanje cilja projekta, njegov značaj, kao i opseg. Upravo ti koraci određeni su identifikacijom korisničkih zahteva. Jedan od osnovnih zadataka jeste svakako i detaljno istraživanje, analiza realnog sistema i poslovanja prodaje karata za sportske događaje kako bi se uočili eventualni nedostaci konkurenata i na adekvatan način projektovao informacioni sistem.

U okviru projektne specifikacije predstavljeni su svi koraci razvoja aplikacije, od osnovnih principa UML modelovanja, preko rada u Microsoft SQL Server-u i razvoja *backend* i *frontend* dela aplikacije.

Izabrani realni sistem jeste sistem za prodaju karata za sportske događaje. U okviru tog sistema potrebno je pružiti mogućnost korisnicima da steknu uvid u predstojeće sportske događaje. Sistem ima korisnički interfejs koji korisnicima omogućava da pretražuju, pregledaju i rezervišu karte za sportske događaje. Omogućava im pregled različitih informacija o sportskim događajima poput naziva samog događaja, datuma i vremena održavanja, lokacije, kao i drugih relevantnih detalja. Nakon što steknu uvid u trenutno aktuelne događaje korisnicima se pruža mogućnost kupovine karata preko sistema. Način plaćanja porudžbine koji je odabran za sistem jeste online plaćanje. Druga vrsta korisnika u sistemu je administrator. Administrator ima široku ulogu u sistemu. Ima uvid u evidenciju prijavljenih korisnika, karata, kao i porudžbina. Takođe pruža mu se mogućnost dodavanja, ažuriranja i brisanja podataka o navedenim entitetima. Korisnicima se omogućava kreiranje korisničkih naloga kako bi imali pristup kupovini karata. Sistem treba da obezbedi sigurnu obradu plaćanja i zaštitu ličnih podataka korisnika.

1.1 Cilj projekta

Cilj projektnog zadatka je razvijanje veb aplikacije koja podržava rad odnosno poslovanje sistema prodaje karata za sportske događaje. Dakle, to omogućava ispunjenje svih funkcionalnosti obuhvaćenih u specifikaciji zahteva. Takođe, potrebno je obuhvatiti što veći broj mogućih situacija, kako bi sistem funkcionisao bez nepredviđenih situacija i nastanka bezizlaznih problema.

2. Korišćene tehnologije

U nastavku će biti prikazane i detaljno objašnjene tehnologije i alati koji su korišćeni prilikom izrade projekta.

2.1 Kreiranje UML dijagrama

Na samom početku potrebno je prikazati izvođenje određenih procesa upotrebom UML dijagrama. Za potrebe projektovanja informacionog sistema korišćeni su sledeći dijagrami:

- Dijagram slučajeva upotrebe
- Dijagram klasa
- Dijagram sekvenci

Potrebno je koristiti veći broj dijagrama kako bi se realni sistem sagledao iz više različitih perspektiva i time stekao bolji pogled na njega, kao i da bi se umanjila mogućnost za grešku.

Za kreiranje UML dijagrama korišćen je GenMyModel.

GenMyModel je besplatna online platforma za kreiranje UML dijagrama za programere i softverske arhitekta. Prednost je mogućnost korišćenja sa bilo kog kompjutera, internet pretraživača i bilo kog operativnog sistema.

Unified Modeling Language (UML) je formalni jezik, namenjen za formalno specificiranje sistema (mogućih stanja i ponašanja). Podrazumeva modelovanje, vizuelizaciju, analizu i dokumentovanje sistema. Njegova velika prednost jeste nezavisnost od programskih jezika i zasnovanost na principima objektno orijentacije.

2.2 Kreiranje baze podataka

Detaljnom analizom svih činilaca realnog sistema kao i pažljivim prikupljanjem informacija potrebno je omogućiti kreiranje baze podataka koja olakšava poslovanje svim korisnicima sistema. Baza podataka korisnicima pruža različite manipulacije i operacije nad podacima i interakcije korisnika preko tog sadržaja.

Konkretna baza podataka omogućava beleženje i organizaciju podataka o administratorima, korisnicima, kartama, kao i porudžbinama.

Za kreiranje baze podataka koristi se Microsoft SQL Server Management Studio. Sql Server predstavlja sistem za upravljanje relacionim bazama podataka (RDBMS) koji je kreirao Microsoft. Omogućava skladištenje i upravljanje podacima.

Predstavlja model koji se koristi za organizaciju podataka u obliku tabela i definisanje veza između tih tabela. Osnovna ideja relacionog modela jeste da podaci budu organizovani u relacijama, gde svaka relacija predstavlja tabelu sa redovima i kolonama (atributima).

2.3 Kreiranje *backend* dela aplikacije

Nakon toga potrebno je preći na izradu *backend* dela aplikacije. Izabrana tehnologija jeste ASP .NET Core Web API koja predstavlja popularan izbor za robustan i skalabilan *backend* razvoj. Omogućava izradu API-ja, rad sa bazama podataka i integraciju sa drugim sistemima. Pruža visoke performanse i sigurnost.

Microsoft je razvio .NET kao okruženje za razvoj softvera u okviru kog je omogućeno korišćenje velikog broja programskih jezika od kojih su najpopularniji *C#*, *C++* i *VisualBasic*. ASP.NET Core predstavlja *cross-platform*, *open-source framework* za izradu modernih i cloud aplikacija. *Web Api*, koji će biti korišćen u okviru pokaznog primera, omogućava kreiranje RESTful HTTP servisa.

ASP.NET Core pruža mogućnosti kao što su dependency injection, modul za autentifikaciju i autorizaciju, te middleware komponente koje olakšavaju upravljanje HTTP zahtevima. Sve ovo doprinosi bržem i efikasnijem razvoju aplikacija.

Swagger pomaže korisnicima pri dokumentovanju i testiranju RESTful veb servisa. U okviru pokaznog primera Swagger dokumentacija će se automatski generisati na osnovu napisanog koda. Drugi način koji postoji omogućava prvo kreiranje Swagger dokumentacije, a zatim kreiranje servisa na osnovu te dokumentacije. Pored toga, Swagger omogućava interaktivno ispitivanje API endpoint-a, što dodatno olakšava razvoj i testiranje.

2.4 Kreiranje sigurnosnog dela aplikacije

Kada je reč o sigurnosnim tehnologijama koje su korišćene u okviru aplikacije za autorizaciju i autentifikaciju korišćen je OAuth, JWT (JSON Web Tokens) predstavlja standarde za autorizaciju i autentifikaciju korisnika. Omogućavaju sigurnu prijavu i zaštitu korisničkih podataka.

OAuth omogućava trećim stranama ograničen pristup korisničkim podacima bez deljenja lozinki, dok JWT omogućava prenos podataka između strana kao JSON objekti na siguran način. JWT se često koristi za održavanje sesija korisnika jer može uključivati informacije o korisniku i ima ugrađenu sigurnost pomoću potpisa koji omogućava proveru integriteta tokena.

2.5 Kreiranje *frontend* dela aplikacije

Na samom kraju važno je obraditi i *frontend* deo aplikacije. U okviru ovog dela korišćen je Angular koji predstavlja moderni JavaScript framework za izradu dinamičnih korisničkih interfejsa. Olakšava razvoj kompleksnih aplikacija omogućavajući bržu i efikasniju izradu funkcionalnosti. HTML, CSS i JavaScript predstavljaju tehnologije za

izradu veb stranica koje omogućavaju kreirane strukture, stilizaciju i interaktivnost. HTML se koristi za strukturu sadržaja, CSS za stilizaciju, dok JavaScript dodaje interaktivnost i dinamiku.

TypeScript je programski jezik otvorenog koda koji razvija i održava Microsoft. Predstavlja proširenje JavaScript-a, a dodaje jeziku opcionu statičku tipizaciju i objektnu orijentisanost. To omogućava lakše održavanje koda, bolju detekciju grešaka pre vremena izvršavanja i povećava produktivnost programera.

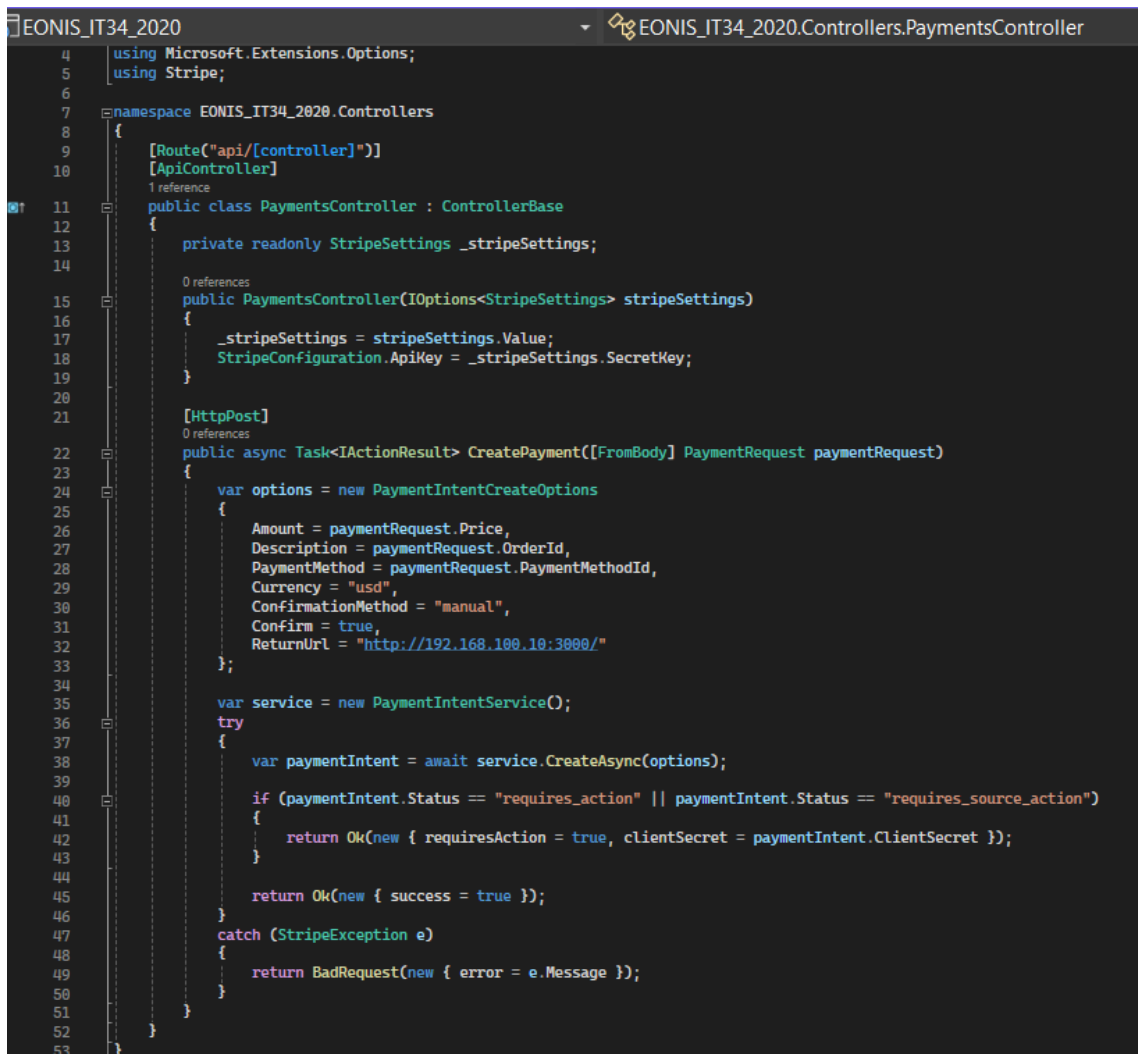
Node.js je višeplatformsko JavaScript radno okruženje otvorenog kôda za izvršavanje JavaScript-a na serverskoj strani, što omogućava kreiranje dinamičkih web stranica. Node.js koristi event-driven, non-blocking I/O model, što ga čini efikasnim i skalabilnim za real-time aplikacije.

2.6 Korišćenje *Stripe*-a

Kada je reč o tehnologijama za online plaćanje korišćen je Stripe koji predstavlja platni procesor koji omogućava sigurne online transakcije. Stripe omogućava integraciju sa različitim platnim metodama, automatizaciju procesa plaćanja i visok nivo sigurnosti zahvaljujući PCI DSS sertifikaciji. Stripe takođe nudi API za prilagođavanje plaćanja specifičnim potrebama korisnika, što olakšava integraciju u različite aplikacije.

Pored toga, Stripe nudi detaljne analitike i izvještaje o transakcijama, što pomaže poslovima u praćenju i optimizaciji finansijskih procesa.

Na sledećoj slici prikazan je izgled PaymentsController-a.



```

4      using Microsoft.Extensions.Options;
5      using Stripe;
6
7      namespace EONIS_IT34_2020.Controllers
8      {
9          [Route("api/[controller]")]
10         [ApiController]
11         public class PaymentsController : ControllerBase
12         {
13             private readonly StripeSettings _stripeSettings;
14
15             public PaymentsController(IOptions<StripeSettings> stripeSettings)
16             {
17                 _stripeSettings = stripeSettings.Value;
18                 StripeConfiguration.ApiKey = _stripeSettings.SecretKey;
19             }
20
21             [HttpPost]
22             public async Task<IActionResult> CreatePayment([FromBody] PaymentRequest paymentRequest)
23             {
24                 var options = new PaymentIntentCreateOptions
25                 {
26                     Amount = paymentRequest.Price,
27                     Description = paymentRequest.OrderId,
28                     PaymentMethod = paymentRequest.PaymentMethodId,
29                     Currency = "usd",
30                     ConfirmationMethod = "manual",
31                     Confirm = true,
32                     ReturnUrl = "http://192.168.100.10:3000/"
33                 };
34
35                 var service = new PaymentIntentService();
36                 try
37                 {
38                     var paymentIntent = await service.CreateAsync(options);
39
40                     if (paymentIntent.Status == "requires_action" || paymentIntent.Status == "requires_source_action")
41                     {
42                         return Ok(new { requiresAction = true, clientSecret = paymentIntent.ClientSecret });
43                     }
44
45                     return Ok(new { success = true });
46                 }
47                 catch (StripeException e)
48                 {
49                     return BadRequest(new { error = e.Message });
50                 }
51             }
52         }
53     }

```

Slika 1. Payments Controller

2.7 Korišćenje Webhook-a

Webhook je tehnologija koja omogućava automatizovanu komunikaciju između različitih sistema putem HTTP zahteva. Kada se dogodi određeni događaj u aplikaciji, Webhook šalje real-time obaveštenje u vidu HTTP POST zahteva na unapred definisanu URL adresu. Ovo omogućava momentalnu reakciju i integraciju sa drugim sistemima bez potrebe za periodičnim proverama.

Webhook se često koristi za ažuriranje podataka, sinhronizaciju sistema, obaveštavanje korisnika o promenama ili pokretanje specifičnih akcija kao odgovor na događaje. Zahvaljujući jednostavnosti implementacije i mogućnosti prilagođavanja, Webhook je postao standardni alat u modernom razvoju softvera.

Pored toga, Webhook omogućava povećanu efikasnost i brzinu obrade podataka jer eliminiše potrebu za ručnim proverama i omogućava sistemima da odmah reaguju na promene. Ova tehnologija je ključna za stvaranje responzivnih i interaktivnih aplikacija koje zahtevaju real-

time obradu događaja.

Na sledećim slikama prikazan je izgled WebhookController-a.

```
namespace EONIS_IT34_2020.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class WebhooksController : ControllerBase
    {
        private readonly StripeSettings _stripeSettings;
        private readonly IPorudzbinaRepository porudzbinaRepository;

        public WebhooksController(IOptions<StripeSettings> stripeSettings, IPorudzbinaRepository porudzbinaRepository)
        {
            _stripeSettings = stripeSettings.Value;
            this.porudzbinaRepository = porudzbinaRepository;
        }

        [HttpPost]
        public async Task<IActionResult> Stripe()
        {
            var json = await new StreamReader(HttpContext.Request.Body).ReadToEndAsync();
            try
            {
                var stripeEvent = EventUtility.ConstructEvent(
                    json,
                    Request.Headers["Stripe-Signature"],
                    _stripeSettings.WebhookSecret
                );

                if (stripeEvent.Type == Events.PaymentIntentSucceeded)
                {
                    var paymentIntent = stripeEvent.Data.Object as PaymentIntent;
                    // Handle successful payment intent
                    System.Console.WriteLine($"PaymentIntent was successful: {paymentIntent.Id}");

                    var orderId = paymentIntent.Description;
                    Guid.TryParse(orderId, out Guid guidOrderId);
                    if (guidOrderId != Guid.Empty)
                    {
                        Porudzbina porudzbina = porudzbinaRepository.GetExactPorudzbina(guidOrderId);
                        porudzbina.StatusPorudzbine = "Završena";
                        porudzbina.PotvrdaPlacanja = "Placeno";
                        porudzbinaRepository.UpdatePorudzbina(porudzbina);
                    }
                }
            }
        }
    }
}
```

Slika 2. Webhook Controller [1]


```

EONIS_IT34_2020
EONIS_IT34_2020.Controllers.WebhooksController

25 public async Task<IActionResult> Stripe()
26 {
27     var json = await new StreamReader(HttpContext.Request.Body).ReadToEndAsync();
28     try
29     {
30         var stripeEvent = EventUtility.ConstructEvent(
31             json,
32             Request.Headers["Stripe-Signature"],
33             _stripeSettings.WebhookSecret
34         );
35
36         if (stripeEvent.Type == Events.PaymentIntentSucceeded)
37         {
38             var paymentIntent = stripeEvent.Data.Object as PaymentIntent;
39             // Handle successful payment intent
40             System.Console.WriteLine($"PaymentIntent was successful: {paymentIntent.Id}");
41
42             var orderId = paymentIntent.Description;
43             Guid.TryParse(orderId, out Guid guidOrderId);
44             if (guidOrderId != Guid.Empty)
45             {
46                 Porudzbina porudzbina = porudzbinaRepository.GetExactPorudzbina(guidOrderId);
47                 porudzbina.StatusPorudzbine = "Završena";
48                 porudzbina.PotvrdaPlacanja = "Placeno";
49                 porudzbinaRepository.UpdatePorudzbina(porudzbina);
50             }
51         }
52         else if (stripeEvent.Type == Events.PaymentIntentPaymentFailed)
53         {
54             var paymentIntent = stripeEvent.Data.Object as PaymentIntent;
55             // Handle failed payment intent
56             System.Console.WriteLine($"PaymentIntent failed: {paymentIntent.Id}");
57
58             var orderId = paymentIntent.Description;
59             Guid.TryParse(orderId, out Guid guidOrderId);
60             if (guidOrderId != Guid.Empty)
61             {
62                 Porudzbina porudzbina = porudzbinaRepository.GetExactPorudzbina(guidOrderId);
63                 porudzbina.StatusPorudzbine = "Otkazana";
64                 porudzbinaRepository.UpdatePorudzbina(porudzbina);
65             }
66         }
67
68         return Ok();
69     }
70     catch (StripeException e)
71     {
72         return BadRequest(new { error = e.Message });
73     }
74     catch (Exception ex)
75     {
76         return StatusCode(500, new { error = ex.Message });
77     }
78 }

```

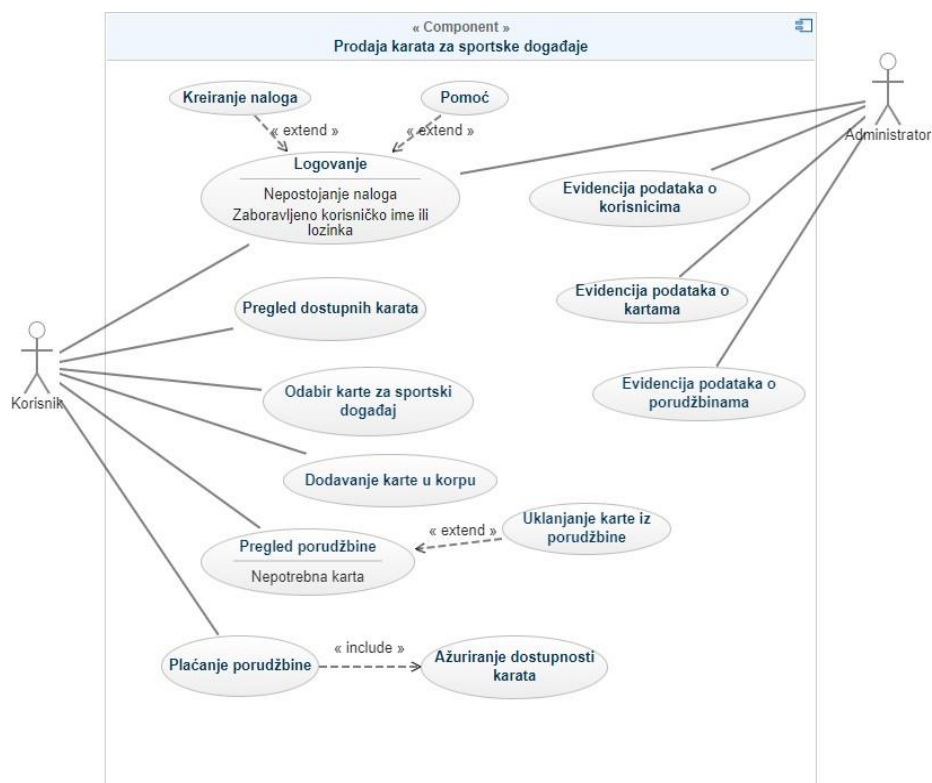
Slika 3. Webhook Controller [2]

3 UML dijagrami

Na samom početku pristupamo inicijalnom modelovanju sistema odnosno kreiranju dijagrama slučajeve upotrebe na osnovu datog opisa za sistem vulkanizerske radnje.

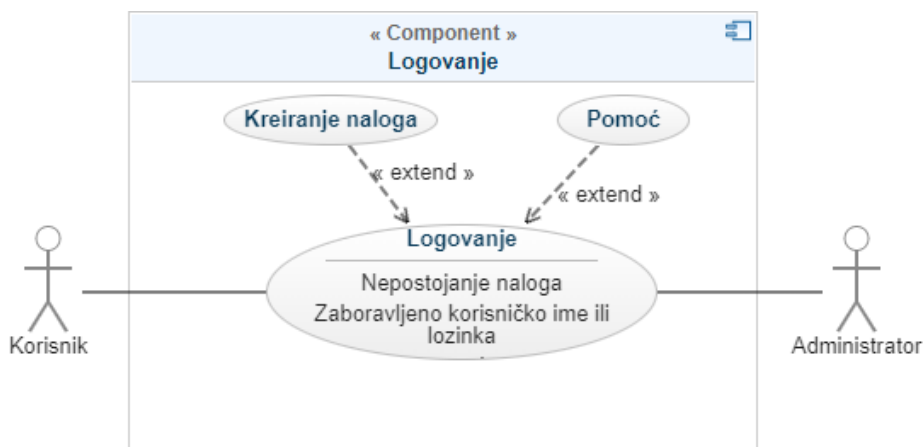
3.1 Dijagram slučajeve upotrebe

Slika 4 Use Case Dijagram prikazuje dijagram slučajeve upotrebe koji se odnosi na poslovanje sistema prodaje karata za sportske događaje. Takođe, na slikama 5 – 10 prikazani su slučajevi upotrebe grupisani po aktivnostima.



Slika 4. Use Case Dijagram

U tabelama 1–9 dati su tekstualni opisi slučajeva upotrebe prikazanih redom na slikama 5 – 10.

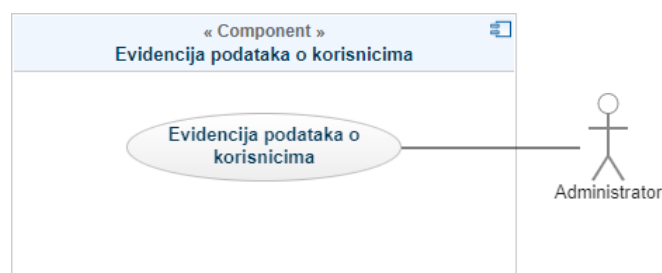


Slika 5. Logovanje – dijagram slučajeva upotrebe

Tabela 1. Logovanje

Logovanje	
Kratak opis slučaja upotrebe	Mogućnost logovanja u informacioni sistem.
Učesnici	Korisnik, administrator

Uslovi koji moraju biti zadovoljeni pre izvršenja	Postojanje informacionog sistema i sistema koji omogućava kreiranje naloga.
Opis	<p>Administrator logovanjem pristupa svom nalogu na informacionom sistemu, što mu omogućava korišćenje IS u različite svrhe kao zapis brojnih evidencija, unos i čitanje različitih podataka.</p> <p>Korisnik logovanjem pristupa svom nalogu što mu omogućava izvršavanje porudžbine odnosno kupovinu karte za sportski događaj.</p> <p>[Izuzeci: Nepostojanje naloga ili zaboravljeni podaci]</p>
Izuzeci	[Izuzetak: Nepostojanje naloga ili zaboravljeni podaci] Ukoliko određeni administrator/korisnik nema svoj nalog ili je zaboravio podatke, potrebno je da u zavisnosti od problema kreira novi nalog ili na određeni način pristupi izgubljenim podacima.
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Postojanje profila administratora/korisnika koji je potreban za dalju saradnju. Omogućen pristup informacionom sistemu.



Slika 6. Evidencija podataka o korisnicima – dijagram slučajeva upotrebe

Tabela 2. Evidencija podataka o korisnicima

Evidencija podataka o korisnicima	
Kratak opis slučaja upotrebe	Administrator unosi podatke o korisnicima.
Učesnici	Administrator
Uslovi koji moraju biti zadovoljeni pre izvršenja	Administrator mora imati podatke o svim korisnicima u sistemu.

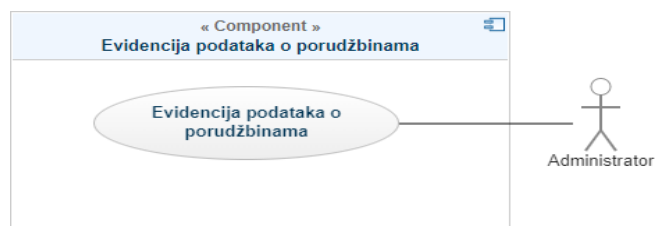
Opis	Administrator unosi podatke o svim korisnicima u bazu podataka i ažurira ih po potrebi. [<i>Izuzeci:</i> /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Podaci o korisnicima su evidentirani.



Slika 7. Evidencija podataka o kartama – dijagram slučajeva upotrebe

Tabela 3. Evidencija podataka o kartama

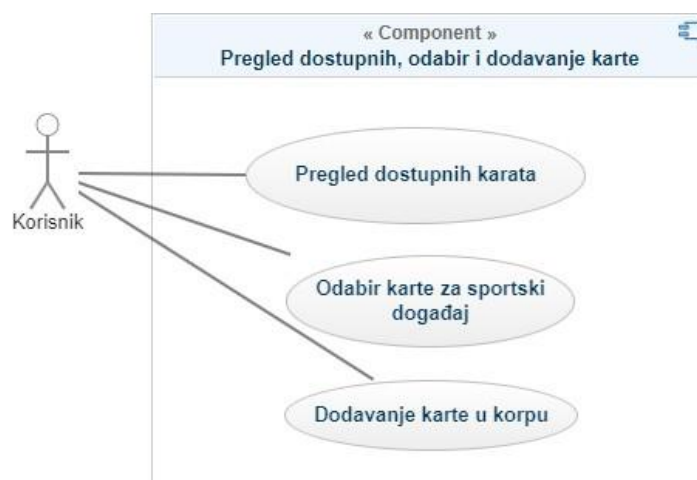
Evidencija podataka o kartama	
Kratak opis slučaja upotrebe	Administrator unosi podatke o kartama za sportske događaje.
Učesnici	Administrator
Uslovi koji moraju biti zadovoljeni pre izvršenja	Administrator ima podatke o svim kartama za sportske događaje.
Opis	Administrator unosi u bazu podataka podatke o svim kartama za sportske događaje i po potrebi ih ažurira. [<i>Izuzeci:</i> /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Podaci o kartama za sportske događaje su evidentirani.



Slika 8. Evidencija podataka o porudžbinama – dijagram slučajeva upotrebe

Tabela 4. Evidencija podataka o porudžbinama

Evidencija podataka o porudžbinama	
Kratak opis slučaja upotrebe	Administrator unosi podatke o porudžbinama.
Učesnici	Administrator
Uslovi koji moraju biti zadovoljeni pre izvršenja	Administrator ima podatke o svim porudžbinama.
Opis	Administrator unosi u bazu podataka podatke o svim porudžbinama i po potrebi ih ažurira. [Izuzeci: /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Podaci o porudžbinama su evidentirani.



Slika 9. Pregled dostupnih, odabir i dodavanje karte – dijagram slučajeva upotrebe

Tabela 5. Pregled dostupnih karata

Pregled dostupnih karata	
Kratak opis slučaja upotrebe	Korisnik ima uvid u trenutno postojeće karte za predstojeće sportske događaje.
Učesnici	Korisnik
Uslovi koji moraju biti zadovoljeni pre izvršenja	Postojanje odgovarajuće evidencije o kartama za sportske događaje.
Opis	Korisnik na osnovu postojeće evidencije o kartama može da vidi predstojeće sportske događaje. [Izuzeci: /]
Izuzeci	/

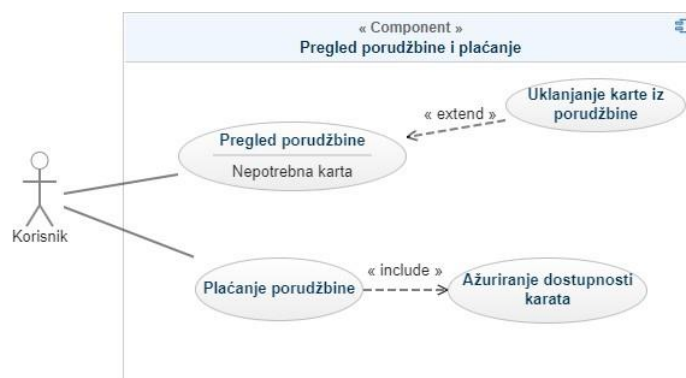
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Korisnik je stekao uvid u trenutno stanje predstojećih sportskih događaja.
--	--

Tabela 6. Odabir karte za sportski događaj

Odabir karte za sportski događaj	
Kratak opis slučaja upotrebe	Korisnik bira kartu/karte za predstojeći sportski događaj.
Učesnici	Korisnik
Uslovi koji moraju biti zadovoljeni pre izvršenja	Korisnik je stekao uvid o aktuelnim sportskim događajima za koje je moguće kupiti kartu.
Opis	Korisnik bira kartu za sportski događaj. [Izuzeci: /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Korisnik je izabrao kartu koja će dalje predstavljati stavku porudžbine

Tabela 7. Dodavanje karte u korpu

Dodavanje karte u korpu	
Kratak opis slučaja upotrebe	Dodavanje karte u korpu odnosno kreiranje porudžbine.
Učesnici	Korisnik
Uslovi koji moraju biti zadovoljeni pre izvršenja	Korisnik je izvršio pregled aktuelnih karata i izabrao kartu za predstojeći sportski događaj.
Opis	Nakon pregleda i odabira karte za sportski događaj korisnik je dodao kartu u korpu. [Izuzeci: /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Karta je dodata u korpu, tj. kreirana je porudžbina.



Slika 10. Pregled porudžbine i plaćanje – dijagram slučajeva upotrebe

Tabela 8. Pregled porudžbine

Pregled porudžbine	
Kratak opis slučaja upotrebe	Korisnik pregleda porudžbinu.
Učesnici	Korisnik
Uslovi koji moraju biti zadovoljeni pre izvršenja	Korisnik je dodao kartu u korpu.
Opis	Nakon dodavanja karte u korpu korisnik stiče uvid u svoju porudžbinu. [Izuzeci: Korisnik smatra da je karta nepotrebna]
Izuzeci	[Izuzetak: Korisnik smatra da je karta nepotrebna] Ukoliko korisnik smatra da je karta nepotrebna moguće je da izbaci kartu iz porudžbine.
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Porudžbina je kreirana.

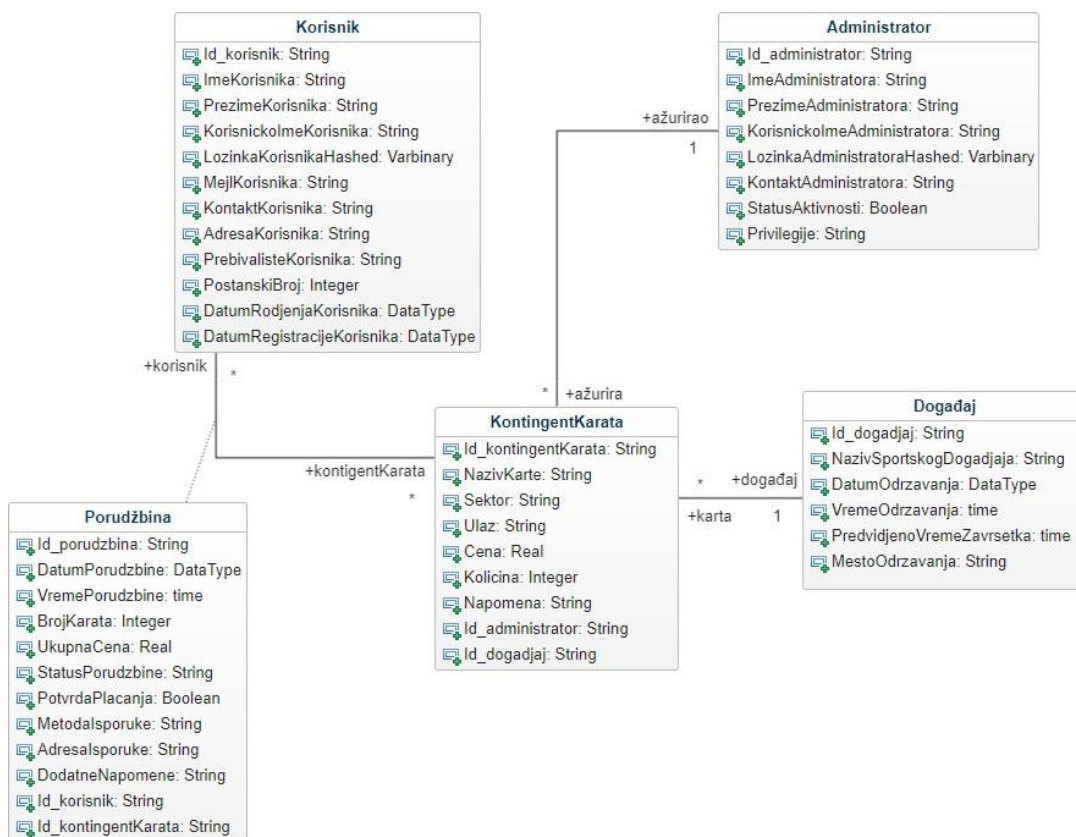
Tabela 9. Plaćanje porudžbine

Plaćanje porudžbine	
Kratak opis slučaja upotrebe	Korisnik plaća porudžbinu.
Učesnici	Korisnik
Uslovi koji moraju biti zadovoljeni pre izvršenja	Klijent je izabrao kartu i kreirao porudžbinu čije plaćanje mora da izvrši.

Opis	Za potrebe sistema odabran način za plaćanje porudžbine jeste plaćanje karticom (online). Korisnik vrši plaćanje nakon čega dobija kartu za predstojeći sportski događaj. Nakon izvršenog plaćanja vrši se ažuriranje dostupnosti karata u sistemu tako da ta karta više nije dostupna. [Izuzeci: /]
Izuzeci	/
Uslovi koji moraju biti zadovoljeni nakon izvršenja	Porudžbina je uspešno plaćena.

3.2 Dijagram klasa

Sistem za podršku poslovanja prodaje karata za sportske događaje obuhvata administratore koji čija uloga jeste da doda, modifikuje i briše karte. S druge strane, jednu kartu dodao je jedan administrator. Korisnik ima mogućnost da kupi jednu ili više karata, ali i da uopšte ne kupi kartu. Jednu kartu može da kupi jedan korisnik zbog ograničenja mesta. Korisnik može da kreira više porudžbina, ali se jedna porudžbina odnosi na tačno jednog korisnika. Slika 11 prikazuje prethodno opisani dijagram klasa.



Slika 11. Dijagram klasa

3.3 Dijagram sekvenci

Dijagram sekvenci se koristi za prikaz dinamičke saradnje između objekata u vremenu. Objekti komuniciraju preko sekvenci poruka. Dijagram sekvenci opisuje tok poruka između objekata kojima se realizuje odgovarajuća operacija u sistemu.

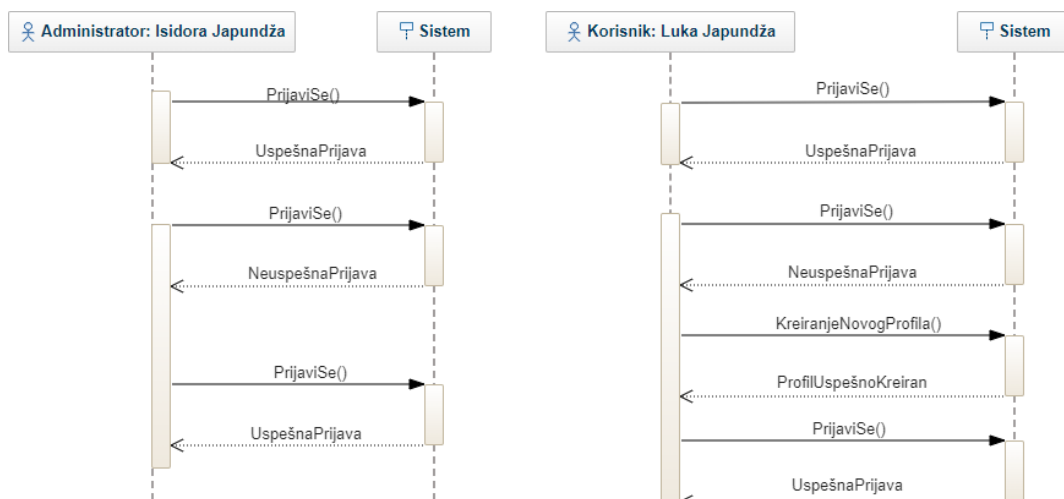
U nastavku će redom biti prikazani dijagrami sekvenci (slike 12–15) i njihovi opisi.

[1] Dijagram sekvenci koji prikazuje proces logovanja administratora u sistem:

Dijagram prikazuje proces logovanja administratora u sistem. Nakon unošenja ispravnih kredencijala omogućava se prijava u sistem. U suprotnom to nije moguće i potrebno je izvršiti ponovno prijavljivanje u sistem. Analogno važi i za korisnika.

[2] Dijagram sekvenci koji prikazuje proces logovanja korisnika u sistem:

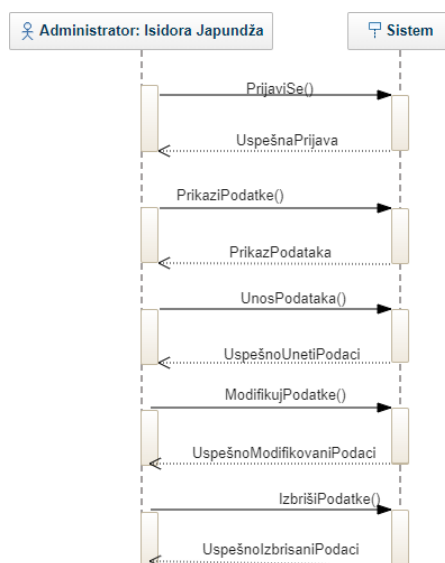
Dijagram prikazuje proces logovanja umetnika u sistem. Nakon unošenja ispravnih kredencijala omogućava se prijava u sistem. Ukoliko korisnik nema kreiran profil prethodno, neophodno je prvobitno ga kreirati, a potom izvršiti prijavljivanje u sistem.



Slika 12. Dijagram sekvenci – [1] i [2]

[3] Dijagram sekvenci koji prikazuje proces prikaza evidencije o korisnicima:

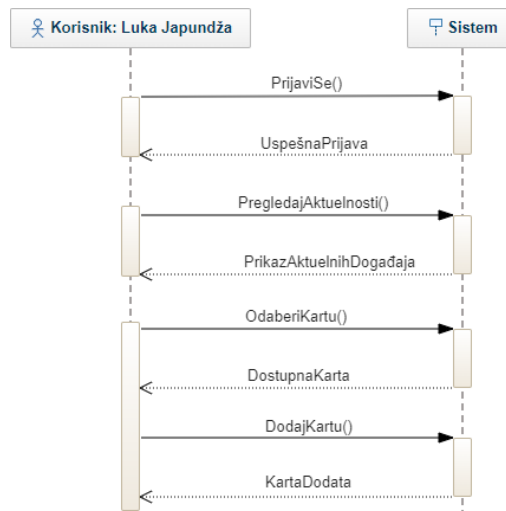
Nakon uspešnog logovanja, administrator ima uvid u podatke o korisnicima. Administrator može da manipuliše podacima, dodaje nove, modifikuje ili briše postojeće korisnike. Analogno, dijagrami sekvenci [4] i [5] bi podrazumevali ekvivalentne operacije. Međutim, razlika je u tome što bi obuhvatali podatke o kartama ([4]) odnosno porudžbinama ([5]).



Slika 13. Dijagram sekvenci – [3], [4] i [5]

[6] Dijagram sekvenci koji prikazuje proces prikaza dostupnih karata, kao i odabir i dodavanje karte:

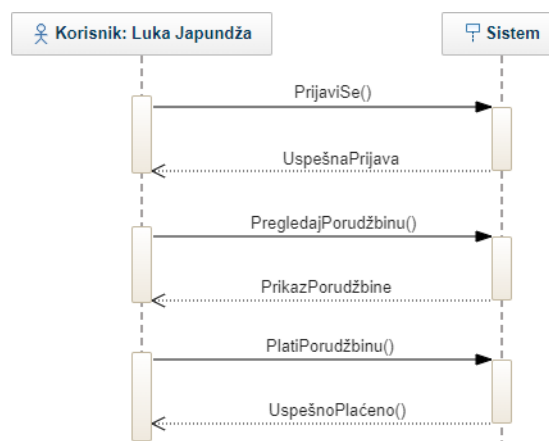
Nakon uspešnog logovanja, korisnik ima mogućnost da pregleda trenutno aktuelne događaje i izabere kartu za konkretan događaj. Ukoliko je karta dostupna vrši se dodavanje karte u korpu za porudžbinu.



Slika 14. Dijagram sekvenci – [6]

[7] Dijagram sekvenci koji prikazuje proces pregleda i plaćanja porudžbine:

Nakon uspešnog logovanja i dodavanja karte u korpu, korisnik ima mogućnost pregleda same korpe odnosno porudžbine. Nakon toga vrši se plaćanje porudžbine.



Slika 15. Dijagram sekvenci – [7]

4. Baza podataka

Nakon završene prve faze projekta koja je na detaljan način opisana u prethodnom poglavlju prelazi se na proces projektovanja. Za kreiranje baze podataka koristi se Microsoft SQL Server Management Studio.

4.1 Data Definition Language – DDL

DDL (Data Definition Language) predstavlja skup naredbi u *SQL*-u koje se koriste za definisanje i upravljanje strukturom baze podataka. DDL se koristi za definisanje objekata baze podataka kao što su tabele, sekvence, funkcije, procedure, trigeri, pogledi, uloge itd. Najznačajnije DDL naredbe su *CREATE*, *ALTER*, *DROP* i *TRUNCATE* koje služe, redom, za kreiranje, modifikaciju i brisanje objekata baze podataka. U nastavku su prikazane naredbe korišćene u projektu (slika 16 – 17).

```
----- DROP TABLE -----
3IF OBJECT_ID('EONIS_IT34_2020.Karta', 'U') IS NOT NULL
    DROP TABLE EONIS_IT34_2020.Karta;
GO

3IF OBJECT_ID('EONIS_IT34_2020.Administrator', 'U') IS NOT NULL
    DROP TABLE EONIS_IT34_2020.Administrator;
GO

3IF OBJECT_ID('EONIS_IT34_2020.Porudzbina', 'U') IS NOT NULL
    DROP TABLE EONIS_IT34_2020.Porudzbina;
GO

3IF OBJECT_ID('EONIS_IT34_2020.Korisnik', 'U') IS NOT NULL
    DROP TABLE EONIS_IT34_2020.Korisnik;
GO
```

Slika 16. DDL – Primer korišćenja naredbe *DROP* za brisanje tabela

Naredba *DROP* se koristi za brisanje objekata baze podataka. Na Slika 16 je prikazana *DROP* naredba brisanje tabela ukoliko one postoje.

Na narednoj slici (Slika 17) prikazana je *DDL* naredba za kreiranje tabele *Korisnik*. Prilikom kreiranja tabele potrebno je navesti naziv tabele, zatim skup obeležja koja opisuju tu tabelu, kao i njihove tipove podataka i dužinu podataka. Pri definisanju obeležja tabele navodi se i podatak o tome da li obeležje dozvoljava *NULL* vrednosti. Zatim se navode sva ograničenja koja je potrebno definisati za određenu tabelu. Na Slika 17, u okviru tabele *Korisnik*, definisana su različita ograničenja.

```
----- CREATE TABLE -----
CREATE TABLE EONIS_IT34_2020.Korisnik
(
    Id_korisnik uniqueidentifier NOT NULL,
    ImeKorisnika VARCHAR(12) NOT NULL,
    PrezimeKorisnika VARCHAR(20) NOT NULL,
    KorisnickoImeKorisnika VARCHAR(30) NOT NULL,
    LozinkaKorisnikaHashed VARBINARY(MAX),
    --saltKorisnika VARCHAR(30),
    MejlKorisnika VARCHAR(30) NOT NULL,
    KontaktKorisnika VARCHAR(14) NOT NULL,
    AdresaKorisnika VARCHAR(30),
    PrebivalisteKorisnika VARCHAR(15),
    PostanskiBroj Int,
    DatumRodjenjaKorisnika Date NOT NULL,
    DatumRegistracijeKorisnika Date NOT NULL CONSTRAINT DFT_Korisnik_DatumRegistracije DEFAULT(GETDATE())
    CONSTRAINT PK_Korisnik PRIMARY KEY (Id_korisnik),
    CONSTRAINT CHK_Korisnik_PostanskiBroj CHECK (len(PostanskiBroj) = 5),
    CONSTRAINT UC_Korisnik_KorisnickoImeKorisnika UNIQUE (KorisnickoImeKorisnika),
    CONSTRAINT UC_Korisnik_MejlKorisnika UNIQUE (MejlKorisnika)
);
```

Slika 17. DDL – Primer korišćenja naredbe *CREATE* - kreiranje tabele *Korisnik*

4.2 Data Modeling Language – DML

DML (Data Manipulation Language) predstavlja jezik koji se koristi za manipulaciju podataka u tabelama baze podataka. DML se koristi za unos, modifikaciju, brisanje i pretraživanje torki u tabeli. Najznačajnije *DML* naredbe su *SELECT*, *INSERT*, *UPDATE* i *DELETE* koje služe, redom, za pretraživanje, unos, modifikaciju i brisanje torki u tabelama baze podataka.

```
select * from EONIS_IT34_2020.Administrator
select * from EONIS_IT34_2020.Korisnik
select * from EONIS_IT34_2020.KontingentKarata
select * from EONIS_IT34_2020.Dogadjaj
select * from EONIS_IT34_2020.Porudzbina
```

Slika 18. DML – Primer korišćenja naredbe *SELECT* za ispisivanje podataka iz tabele

Naredba *INSERT* se koristi za unos novih torki u tabelu u bazu podataka. Nakon ključne reči *INSERT INTO* navodi se naziv tabele u koju se unose torke i može se navesti lista obeležja tabele, zatim sledi ključna reč *VALUES* i navode se vrednosti koje se unose u tabelu. Na Slika 19 prikazana je naredba *INSERT* za unos nove torke u tabelu *Korisnik*.

```
----- INSERT INTO TABLE -----
INSERT INTO EONIS_IT34_2020.Korisnik(Id_korisnik, ImeKorisnika, PrezimeKorisnika, KorisnickoImeKorisnika, LozinkaKorisnikaHashed,
MejlKorisnika, KontaktKorisnika, AdresaKorisnika, PrebivalisteKorisnika,
PostanskiBroj, DatumRodjenjaKorisnika, DatumRegistracijeKorisnika)
VALUES (
'24726427-d1c1-4c21-979e-de7047a3c330',
'Luka',
'Japundža',
'lukajapundza',
(CONVERT(VARBINARY(MAX), '48656C6C6F20576F726C64', 2)),
'luka.japundza@gmail.com',
'060456789',
'Adi Endrea 1/4',
'Temerin',
21235,
convert(date, '2005-03-19', 23),
GETDATE()
)
```

Slika 19. DML – Primer korišćenja naredbe *INSERT INTO* za dodavanje novih vrednosti u tabelu *Korisnik*

4.3 Triger

Predstavlja specijalnu uskladištenu proceduru koja se vezuje za DML događaje: *INSERT*, *UPDATE*, *DELETE* (ili DDL događaje), pri čemu ti događaji predstavljaju okidač za izvršavanje trigeru. To je SQL blok koji se izvršava kao odgovor na određeni događaj ili akciju nad tabelom. Koriste se za implementaciju poslovnih pravila ili za automatsko ažuriranje podataka. U telu trigeru, moguće je pristupiti privremenim tabelama inserted i deleted. Tabele sadrže torke koje su pod uticajem DML događaja koji je doveo do okidanja trigeru. U slučaju *INSERT* i *UPDATE* događaja, inserted tabela sadrži novo stanje odnosno nove torke. U slučaju *UPDATE* i *DELETE* događaja, deleted tabela sadrži staro stanje odnosno stare torke. Triger se kreira pomoću naredbe *CREATE TRIGGER*. U okviru projekta kreirana su dva trigeru.

Prvi triger koji je kreiran u okviru projekta prikazan je na Slika 20. Obezbeđuje da korisnik ostvaruje popust (10%) na cenu karte ukoliko je datum održavanja sportskog događaja jednak datumu njegovog rođenja.

```

----- CREATE TRIGGER -----
IF OBJECT_ID('ProveriDatumRodjenja', 'TR') IS NOT NULL
    DROP TRIGGER ProveriDatumRodjenja;

CREATE TRIGGER ProveriDatumRodjenja
ON EONIS_IT34_2020.Porudzbina
AFTER INSERT
AS
BEGIN
    DECLARE @Id_porudzbina uniqueidentifier;
    DECLARE @Id_korisnika uniqueidentifier;
    DECLARE @Id_kontingentaKarata uniqueidentifier;
    DECLARE @DatumRodjenjaKorisnika DATE;
    DECLARE @DatumOdrzavanja DATE;
    DECLARE @OriginalnaCena INT;
    DECLARE @NovaCena INT;

    -- vrednosti iz tabele inserted
    SELECT @Id_porudzbina = Id_porudzbina, @Id_korisnika = Id_korisnik, @Id_kontingentaKarata = Id_kontingentaKarata
    FROM inserted;

    -- datum rođenja korisnika
    SELECT @DatumRodjenjaKorisnika = DatumRodjenjaKorisnika
    FROM EONIS_IT34_2020.Korisnik
    WHERE Id_korisnik = @Id_korisnika;

    -- datum održavanja događaja i originalne cene karte
    SELECT @DatumOdrzavanja = D.DatumOdrzavanja, @OriginalnaCena = K.Cena
    FROM EONIS_IT34_2020.Dogadjaj D
    JOIN EONIS_IT34_2020.KontingentaKarata K ON D.Id_dogadjaj = K.Id_dogadjaj
    WHERE K.Id_kontingentaKarata = @Id_kontingentaKarata;

    -- Provera da li su samo dan i mesec datuma jednaki
    IF MONTH(@DatumRodjenjaKorisnika) = MONTH(@DatumOdrzavanja) AND DAY(@DatumRodjenjaKorisnika) = DAY(@DatumOdrzavanja)
    BEGIN
        -- Izračunavanje nove cene sa popustom od 10%
        SET @NovaCena = @OriginalnaCena * 0.9;

        -- Ažuriranje cene u tabeli Porudzbina
        UPDATE EONIS_IT34_2020.Porudzbina
        SET UkupnaCena = @NovaCena * BrojKarata
        WHERE Id_porudzbina = @Id_porudzbina AND Id_korisnik = @Id_korisnika AND Id_kontingentaKarata = @Id_kontingentaKarata;
    END
END;

```

Slika 20. Primer korišćenja trigeru - ProveriDatumRodjenja

Triger je definisan nad tabelom `EONIS_IT34_2020.Porudzbina` i aktivira se kada se vrši operacija (*INSERT*) nad datom tabelom. Umesto da se podaci direktno ubace u tabelu, ovaj triger menja ponašanje umetanja i izvršava određene akcije.

Prvobitno je potrebno kreirati i definisati naziv trigeru, kao i navesti nad kojom tabelom je triger definisan. Nakon toga, naredbom *AFTER INSERT*, definiše se da će triger umesto običnog umetanja podataka izvršiti navedene akcije. *AS BEGIN* označava početak bloka koda trigeru.

Potrebno je izdvojiti dan i mesec iz obeležja koje predstavlja datum održavanja sportskog događaja. Analogno važi i za datum rođenja korisnika. Ukoliko su ove vrednosti jednake potrebno je ažurirati cenu karte odnosno umanjiti je za 10%.

Drugi triger koji je kreiran u okviru projekta prikazan je na Slika 21 i Slika 22. Obezbeđuje proveru karata na stanju prilikom kreiranja porudžbine.

```

----- CREATE TRIGGER ProveriDostupnostKarata -----
IF OBJECT_ID('ProveriDostupnostKarata', 'TR') IS NOT NULL
    DROP TRIGGER ProveriDostupnostKarata;

CREATE TRIGGER ProveriDostupnostKarata
ON EONIS_IT34_2020.Porudzbina
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @Id_porudzbina uniqueidentifier;
    DECLARE @dostupna INT;
    DECLARE @Id_kontingentKarata uniqueidentifier;
    DECLARE @BrojKarata INT;
    DECLARE @UkupnaCena INT;
    DECLARE @StatusPorudzbine VARCHAR(12);
    DECLARE @PotvrdaPlacanja VARCHAR(10);
    DECLARE @MetodaIsporuke VARCHAR(12);
    DECLARE @AdresaIsporuke VARCHAR(12);
    DECLARE @DodatneNapomene VARCHAR(50);
    DECLARE @Id_korisnik uniqueidentifier;
    DECLARE @DatumPorudzbine DATE;
    DECLARE @VremePorudzbine TIME;

    SELECT
        @Id_porudzbina = Id_porudzbina,
        @Id_kontingentKarata = Id_kontingentKarata,
        @BrojKarata = BrojKarata,
        @UkupnaCena = UkupnaCena,
        @StatusPorudzbine = StatusPorudzbine,
        @PotvrdaPlacanja = PotvrdaPlacanja,
        @MetodaIsporuke = MetodaIsporuke,
        @AdresaIsporuke = AdresaIsporuke,
        @DodatneNapomene = DodatneNapomene,
        @Id_korisnik = Id_korisnik,
        @DatumPorudzbine = DatumPorudzbine,
        @VremePorudzbine = VremePorudzbine
    FROM inserted;

```

Slika 21. Primer korišćenja trigera – ProveriDostupnostKarata [1]

```

-- dostupne količine karata za dati kontingent karata
SELECT @dostupna = Kolicina
FROM EONIS_IT34_2020.KontingentKarata
WHERE Id_kontingentKarata = @Id_kontingentKarata;

-- Provera da li je tražena količina veća od dostupne količine
IF @BrojKarata > @dostupna
BEGIN
    RAISERROR ('Nema dovoljno dostupnih karata za ovu porudžbinu', 16, 1);
    ROLLBACK TRANSACTION;
END
ELSE
BEGIN
    -- Ažuriranje dostupne količine karata u kontingentu
    UPDATE EONIS_IT34_2020.KontingentKarata
    SET Kolicina = Kolicina - @BrojKarata
    WHERE Id_kontingentKarata = @Id_kontingentKarata;

    -- Umetanje nove porudžbine
    INSERT INTO EONIS_IT34_2020.Porudzbina
    (
        Id_porudzbina, DatumPorudzbine, VremePorudzbine, BrojKarata, UkupnaCena, StatusPorudzbine,
        PotvrdaPlacanja, MetodaIsporuke, AdresaIsporuke, DodatneNapomene,
        Id_korisnik, Id_kontingentKarata
    )
    VALUES
    (
        @Id_porudzbina, @DatumPorudzbine, @VremePorudzbine, @BrojKarata, @UkupnaCena, @StatusPorudzbine,
        @PotvrdaPlacanja, @MetodaIsporuke, @AdresaIsporuke, @DodatneNapomene,
        @Id_korisnik, @Id_kontingentKarata
    );
END
END;

```

Slika 22. Primer korišćenja trigera – ProveriDostupnostKarata [2]

Triger je definisan nad tabelom `EONIS_IT34_2020.Porudzbina` i aktivira se kada se vrši operacija (*INSERT*) nad datom tabelom. Umesto da se podaci direktno ubace u tabelu, ovaj triger menja ponašanje umetanja i izvršava određene akcije.

Prvobitno je potrebno kreirati i definisati naziv trigera, kao i navesti nad kojom tabelom je triger definisan. Nakon toga, naredbom *AFTER INSERT*, definiše se da će triger umesto običnog umetanja podataka izvršiti navedene akcije. *AS BEGIN* označava početak bloka koda trigera.

Preuzimaju se dostupne količine kontingenta karata za određene događaje. Proverava se da li je tražena količina veća od dostupne količine karata. Ukoliko jeste pokreće se okidač, odnosno izbacuje *RAISERROR*. U suprotnom, ažurira se dostupna količina karata u kontingentu i dodaje se nova porudžbina.

5. Opis predloženog rešenja

Izgradnja veb sajta za podršku poslovanja prodaje karata za sportske događaje ima značajnu vrednost kako za korisnike, tako i za sportske organizacije i same događaje. Neki od ključnih aspekata koji ukazuju na važnost idejnog rešenja predstavljeni su u nastavku. Digitalizacija prodaje karata čini ih dostupnim širem auditorijumu. Ljudi iz različitih regiona mogu lako pristupiti informacijama o događajima i kupiti karte, bez potrebe da fizički posete lokacije za prodaju karata. Online platforma za prodaju karata omogućava korisnicima da brzo i jednostavno pregledaju dostupne događaje, uporede opcije i obave kupovinu u svega nekoliko koraka. Povećava efikasnost i smanjuje vreme potrebno za proces kupovine karata. Kroz detaljne opise događaja, pregled sektora, korisnici mogu donositi odluke o kupovini karata. Transparentnost cena i dostupnosti karata povećava poverenje korisnika

Kada je reč o idejnom rešenju važno je obraditi sledeće stavke, a to su funkcionalnosti veb sajta, tehnološki stek koji će biti korišćen, kao i prednosti samog rešenja. Što se tiče funkcionalnosti veb sajta, one podrazumevaju:

- **Administratorski panel** obuhvata upravljanje korisnicima, događajima, kao i samim porudžbinama. Ima mogućnost dodavanja, ažuriranja informacija o postojećim događajima odnosno kontingentima karata za iste. To uključuje postavljanje cena, određivanje sektora i ulaza.
- **Korisnički nalozi** podrazumeva da korisnik ima mogućnost da se registruje na platformu unosom osnovnih podataka. Proces registracije je brz i jednostavan. Nakon registracije korisnici mogu da se loguju pomoću svojih kredencijala. Takođe, svaki korisnik ima uvid u sopstveni profil.
- **Pregled dostupnih karata** podrazumeva da korisnici imaju uvid u dostupne karte koristeći različite kategorije i filtere. Svaki sportski događaj ima sopstvenu stranicu sa detaljnim informacijama.
- **Kupovina karata** podrazumeva da korisnici dodaju karte u korpu tokom pretrage. Korpa omogućava korisnicima da pregledaju sve dodate karte, promene veličinu ili uklone karte pre nego što završe kupovinu. Platforma podržava *Stripe* metod plaćanja. Proces plaćanja je siguran i jednostavan.
- **Upravljanje porudžbinama** podrazumeva da korisnik ima pregled svojih porudžbina.

Tehnološki stek koji se koristi je sledeći i, za *backend* *.NET Core* i *MSSQL*, a za *frontend* deo *Angular* i *HTML/CSS/JavaScript*:

- **.NET Core** omogućava jednostavnu integraciju sa bazom podataka, efikasno upravljanje podacima i razvoj API-ja koji podržava frontend aplikaciju.
- **MSSQL** je pouzdana i skalabilna baza podataka koja omogućava efikasno skladištenje i upravljanje podacima u sistemu. Integracija sa *.NET* omogućava brzu i sigurnu razmenu podataka.
- **Angular** omogućava izradu responzivnih i dinamičnih korisničkih interfejsa, sa brzim učitavanjem stranica. Podržava modularni razvoj što olakšava održavanje i proširivanje aplikacije.
- **HTML/CSS/JavaScript** predstavljaju osnovne tehnologije za izradu veb stranica koje se koriste u kombinaciji sa Angularom.

Kada je reč o sigurnosti samog sistema koristi se OAuth, JWT (JSON Web Tokens) koji predstavlja standarde za autorizaciju i autentifikaciju korisnika. Omogućavaju sigurnu prijavu i zaštitu korisničkih podataka.

Važno je napomenuti i prednosti ovakvog rešenja koje su sledeće:

- Personalizovano korisničko iskustvo,
- jednostavan i intuitivan interfejs,
- efikasna korisnička podrška,
- transparentnost i sigurnost,
- optimizacija za mobilne uređaje, kao i
- integracija sa društvenim mrežama.

Ovakvo rešenje za podršku poslovanja prodaje karata za sportske događaje nudi inovativan pristup koji prevazilazi mnoge nedostatke postojećih platformi. Fokus na personalizaciju, jednostavnost korišćenja, efikasnu korisničku podršku i druge funkcionalnosti čini idejnu platformu adekvatnom za širok spektar korisnika. Implementacija ovih funkcionalnosti i tehnologija omogućava stvaranje konkurentske prednosti na tržištu i pruža korisnicima pozitivno iskustvo kupovine karata za sportske događaje.

Na narednim slikama je izgled same aplikacije, odnosno predloženog rešenja:

Registracija korisnika

Ime <input type="text"/>	Prezime <input type="text"/>
Korisnicko ime <input type="text"/>	Lozinka <input type="password"/>
Email <input type="text"/>	Kontakt telefon <input type="text"/>
Adresa (nije admin polje) <input type="text"/>	Prebivaliste (nije admin polje) <input type="text"/>
Postanski broj (nije admin polje) <input type="text"/>	Datum rođenja (nije admin polje) <input type="text"/>

Unesite petocifren postanski broj

mm/dd/yyyy

Registriraj se

Registriraj se kao ADMIN

Slika 23. Registracija korisnika

Prijava Registracija

Korisnicka prijava

Korisnicko ime

isidorajapundza

Lozinka

.....

Prijavi se

Prijavi se kao ADMIN

Slika 24. Prijava korisnika

Prijava Registracija

Pretraži događaje

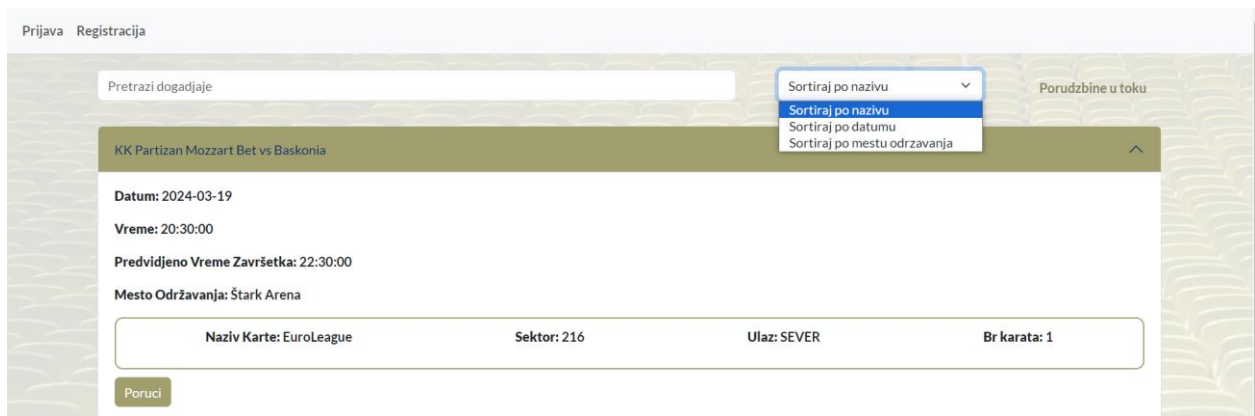
Sortiraj po nazivu

Porudžbine u toku

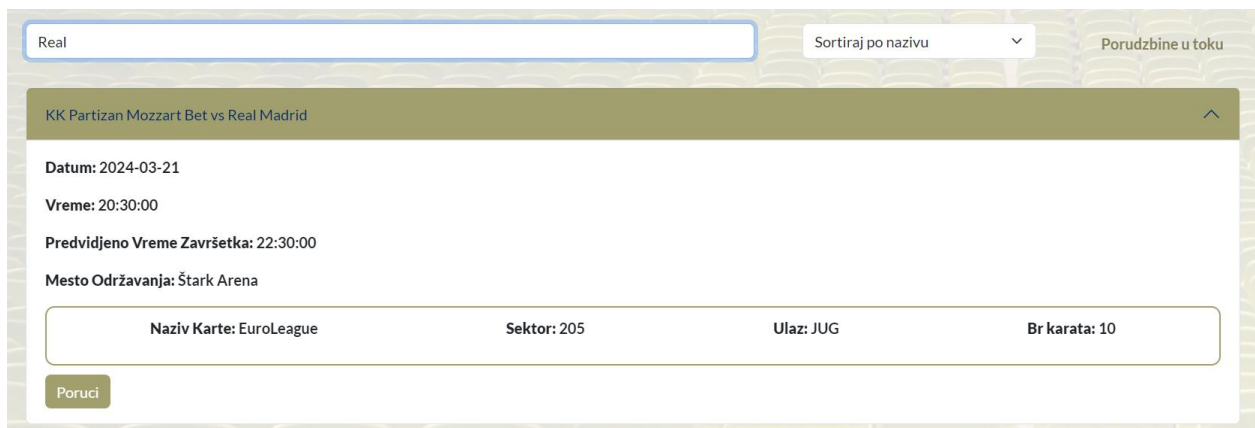
KK Partizan Mozart Bet vs Baskonia	▼
KK Partizan Mozart Bet vs Real Madrid	▼
KK Partizan Mozart Bet vs Olympiacos	▼
KK Partizan Mozart Bet vs KK FMP	▼
FK Partizan vs Spartak Subotica	▼

Sledeća stranica

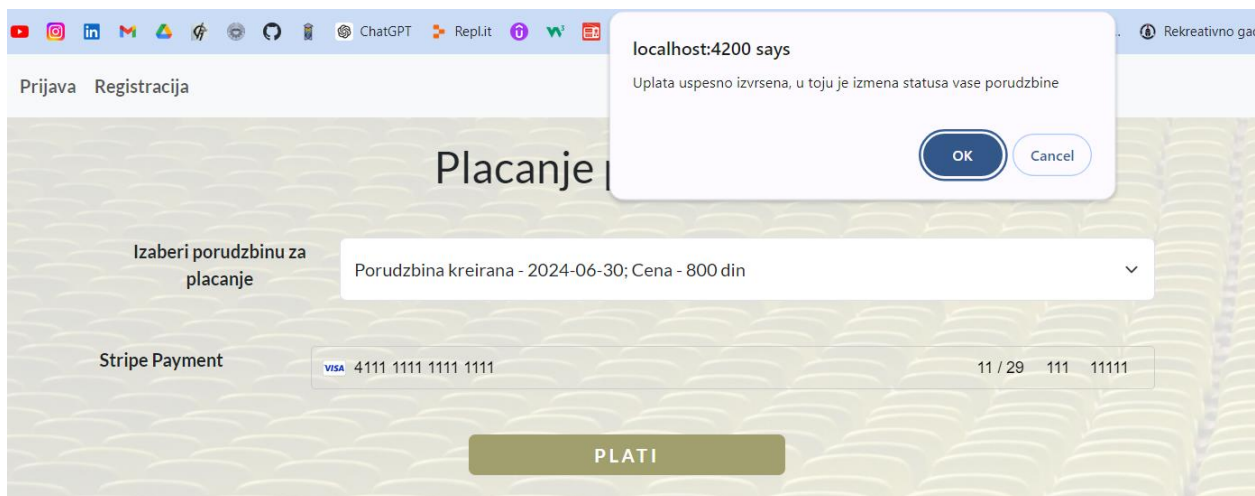
Slika 25. Početna strana



Slika 26. Prikaz proizvoda



Slika 27. Pretraga proizvoda



Slika 28. Plaćanje porudzbine

Webhooks

https://2232-46-40-27-35.ngrok-free.app/api/webhooks

Status

Enabled

Listening for

2 events

API version

2024-04-10

Signing secret

Reveal

Configuration

View logs

All

Succeeded

Failed

payment_intent.succeeded

Response

HTTP status code200 (OK)

Slika 29. Payment succeeded

Prijava Registracija

Porudzbine			
Datum porudzbine	Broj karata	Cena	Status porudzbine
2024-06-30	1	7500 din	Završena
2024-06-30	1	6500 din	Završena
2024-06-30	1	800 din	U toku
2024-06-30	1	7500 din	Završena
2024-06-30	1	800 din	U toku
2024-06-30	1	7500 din	Završena

Slika 30. Prikaz porudžbina [1]

Prijava Registracija

Porudzbine			
KK Partizan Mozart Bet vs Baskonia			
Datum: 2024-03-19			
Vreme: 20:30:00			
Predvidjeno Vreme Završetka: 22:30:00			
Mesto Održavanja: Štark Arena			
Naziv Karte: EuroLeague	Sektor: 216	Ulaz: SEVER	Br karata: 1
KK Partizan Mozart Bet vs Real Madrid			
KK Partizan Mozart Bet vs Olympiacos			
KK Partizan Mozart Bet vs KK FMP			
FK Partizan vs Spartak Subotica			

Slika 31. Prikaz porudžbina [2]

Prijava Registracija

Porudzbine Dogadjaji Dodavanje dogadjaja

Dogadjaj

Naziv sportskog dogadjaja Datum održavanja

Vreme pocetka Vreme zavrsetka

Mesto održavanja

Kontingent karata

Naziv karte Sektor

Ulaz Cena

Kolicina Napomena

Kreiraj dogadjaj

Slika 32. Dodavanje događaja

6. Zaključak

Idejno rešenje za podršku poslovanja prodaje karata za sportske događaje ima značajan potencijal za uspeh. Analizom trenutnog stanja na tržištu, proučavanjem postojećih rešenja, te identifikacijom njihovih prednosti i nedostataka, stekli smo jasan uvid u izazove i mogućnosti koje ovaj projekat nosi. Ulaganje u tehnološki napredne i korisnički orijentisane funkcionalnosti može privući veliku bazu korisnika i osigurati dugoročan rast i uspeh platforme. Važnost ove ideje leži u njenoj sposobnosti da poboljša dostupnost, efikasnost i iskustvo korisnika u procesu kupovine karata, dok primena modernih tehnologija i strategija omogućava efikasno i skalabilno upravljanje sistemom. Mogućnosti primene su široke, uključujući sportske događaje, koncerte, pozorišne predstave i festivale, što omogućava širenje poslovanja na različite segmente tržišta. Saradnja sa relevantnim partnerima dodatno povećava šanse za uspeh platforme. Ovakvo idejno rešenje ne samo da zadovoljava potrebe korisnika, već i doprinosi razvoju digitalizacije u oblasti prodaje karata, donoseći brojne prednosti kako korisnicima, tako i sportskim organizacijama i događajima.

Cilj ovog rada bio je istraživanje mogućnosti implementacije idejnog rešenja. Kako bi se došlo do traženog cilja, prvo su predstavljene teorijske osnove. Nakon toga definisani su zahtevi koje je potrebno implementirati kako bi se ispunio zadati cilj. Naredni korak predstavljao je implementaciju tih zahteva, odnosno kreirano je softversko rešenje. Takođe je u okviru rada nije implementirana samo serverska strana, već je omogućeno i kreiranje korisničkog interfejsa za ovu aplikaciju kako bi se omogućilo njeno korišćenje. U slučaju objavljivanja aplikacije, od velikog značaja bi bilo korišćenje Cloud resursa i korišćenje Docker-a.

Link Github-a:

<https://github.com/IsidoraJapundza/EONIS-2023-24-Japundza-Isidora.git>