**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI DIPLOMSKI STUDIJ**

**Specijalizacija Informaika**



Danijel Klobučar

**API METODE U JAVASCRIPTU**

**DIPLOMSKI RAD**

Zagreb, lipanj 2024.

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI DIPLOMSKI STUDIJ**

**Specijalizacija Informaika**



Danijel Klobučar

**API METODE U JAVASCRIPTU**

**DIPLOMSKI RAD**

Zagreb, lipanj 2024.

Sadržaj:

[1 Uvod 1](#_Toc168067420)

[2 API Metode u JavaScriptu 3](#_Toc168067421)

[2.1 Fetch API 3](#_Toc168067422)

[2.1.1 Fetch promise chaining 3](#_Toc168067423)

[2.1.2 Fetch s asinkronim funkcijama 4](#_Toc168067424)

[2.2 XMLHttpRequest 5](#_Toc168067425)

[2.3 Axios 6](#_Toc168067426)

[2.4 jQuery AJAX 7](#_Toc168067427)

[2.5 Implementacija API-a 8](#_Toc168067428)

[2.5.1 Prvo testiranje s manjom bazom 8](#_Toc168067429)

[2.5.2 Ponovljeno testiranje na bazi s više podataka 18](#_Toc168067430)

[3 Izrada web aplikacije Catering 26](#_Toc168067431)

[3.1 Opis aplikacije 26](#_Toc168067432)

[3.2 Pojmovnik 26](#_Toc168067433)

[3.3 Definicija korisničkih zahtjeva 27](#_Toc168067434)

[3.4 Specifikacija zahtjeva sustava 30](#_Toc168067435)

[3.4.1 Slučaj uporabe UC1 – Registracija vanjskih korisnika 32](#_Toc168067436)

[3.4.2 Slučaj uporabe UC2 – Prijava Managera i registriranih korisnika 34](#_Toc168067437)

[3.4.3 Slučaj uporabe UC3 – Kreiranje narudžbe za catering 36](#_Toc168067438)

[3.4.4 Slučaj uporabe UC4 – Upisivanje broja gostiju i željenih jelovnika 38](#_Toc168067439)

[3.4.5 Slučaj uporabe UC5 – Biranje lokacije 39](#_Toc168067440)

[3.4.6 Slučaj uporabe UC6 – Pregled narudžbi za registriranog korisnika 40](#_Toc168067441)

[3.4.7 Slučaj uporabe UC7 – Uređivanje narudžbi 41](#_Toc168067442)

[3.4.8 Slučaj uporabe UC8 – Ocjena završene narudžbe 42](#_Toc168067443)

[3.4.9 Slučaj uporabe UC9 – Uređivanje osobnih podataka 43](#_Toc168067444)

[3.4.10 Slučaj uporabe UC10 – Pregled narudžbi za Managera 45](#_Toc168067445)

[3.4.11 Slučaj uporabe UC11 – Odluka o potvrdi narudžbe 47](#_Toc168067446)

[3.4.12 Slučaj uporabe UC12 – Uvid u dostupne djelatnike 48](#_Toc168067447)

[3.4.13 Slučaj uporabe UC13 – Generiranje predračuna 49](#_Toc168067448)

[3.4.14 Slučaj uporabe UC14 – Uređivanje podataka o korisnicima 50](#_Toc168067449)

[3.4.15 Slučaj uporabe PUC15 – Uređivanje podataka o zaposlenicima 52](#_Toc168067450)

[3.4.16 Slučaj uporabe UC16 – Uređivanje podataka o dostupnim lokacijama 53](#_Toc168067451)

[3.4.17 Slučaj uporabe UC17 – Uređivanje podataka iz jelovnika 54](#_Toc168067452)

[3.5 Opis sustava 55](#_Toc168067453)

[3.6 Arhitektura sustava programske potpore 56](#_Toc168067454)

[3.7 Opis modela i baze podataka 59](#_Toc168067455)

[3.8 Pregled razreda 61](#_Toc168067456)

[3.8.1 Kontroler 63](#_Toc168067457)

[3.8.2 Manager (modul „Manager“) 64](#_Toc168067458)

[3.8.3 Repozitorij (modul „Repository“) 65](#_Toc168067459)

[3.8.4 Entitet (modul „Entity“) 65](#_Toc168067460)

[3.8.5 Servisi (modul „Services“) 66](#_Toc168067461)

[3.8.6 Enumeracija (modul “Enum“) 66](#_Toc168067462)

[3.8.7 Rukovoditelji događajima (modul „Event“) 67](#_Toc168067463)

[3.8.8 Twig ekstenzije 67](#_Toc168067464)

[3.9 UML dijagrami razreda 68](#_Toc168067465)

[3.10 Pregled atributa 73](#_Toc168067466)

[3.10.1 Kontroler (modul „Controller“) 73](#_Toc168067467)

[3.10.2 Manager (modul „Manager“) 75](#_Toc168067468)

[3.10.3 Repozitorij (modul “Repository”) 76](#_Toc168067469)

[3.10.4 Entitet (modul „Entity“) 76](#_Toc168067470)

[3.10.5 Servisi (modul „Services“) 77](#_Toc168067471)

[3.10.6 Enumeracije (modul „Enum“) 77](#_Toc168067472)

[3.10.7 Rukovoditelj događajima (modul „Event“) 77](#_Toc168067473)

[3.10.8 Twig ekstenzije 77](#_Toc168067474)

[3.11 Pregled metoda 78](#_Toc168067475)

[3.11.1 Kontroler 78](#_Toc168067476)

[3.11.2 Manager (modul „Manager“) 80](#_Toc168067477)

[3.11.3 Repozitorij (modul „Repository“) 81](#_Toc168067478)

[3.11.4 Entitet (modul „Entity“) 84](#_Toc168067479)

[3.11.5 Servisi (modul “Services”) 84](#_Toc168067480)

[3.11.6 Enumeracije (modul „Enum“) 84](#_Toc168067481)

[3.11.7 Rukovoditelj događajim (modul „Event“) 85](#_Toc168067482)

[3.11.8 Twig ekstenzije 85](#_Toc168067483)

[3.12 Opis izvedbe GUI-a 86](#_Toc168067484)

[3.12.1 GUI - Prijava 86](#_Toc168067485)

[3.12.2 GUI – Registracija 86](#_Toc168067486)

[3.12.3 GUI Korisnik – Moji Eventi 87](#_Toc168067487)

[3.12.4 GUI Korisnik – Lokacije 88](#_Toc168067488)

[3.12.5 GUI Korisnik – Meniji 90](#_Toc168067489)

[3.12.6 GUI Manager – Novi Eventi 90](#_Toc168067490)

[3.12.7 GUI Manager – Eventi 91](#_Toc168067491)

[3.12.8 GUI Manager – Korisnici 92](#_Toc168067492)

[3.12.9 GUI Manager – Zaposlenici 93](#_Toc168067493)

[3.12.10 GUI Manager – Lokacije 94](#_Toc168067494)

[3.12.11 GUI Manager – Meniji 95](#_Toc168067495)

[3.12.12 GUI Manager – Jela 96](#_Toc168067496)

[4 Zaključak 97](#_Toc168067497)

[Literatura 98](#_Toc168067498)

Popis slika

[Slika 2.1. Brzina dohvata podataka s Fetch promises chaining na maloj bazi 11](#_Toc168052792)

[Slika 2.2. Brzina obrade podataka s Fetch promises chaining na maloj bazi 11](#_Toc168052793)

[Slika 2.3. Brzina dohvata podataka s Fetch asikronim funkcijama na maloj bazi 12](#_Toc168052794)

[Slika 2.4. Brzina obrade podataka s Fetch asikronim funkcijama na maloj bazi 12](#_Toc168052795)

[Slika 2.5. Brzina dohvata podataka s XMLHttpRequest na maloj bazi 13](#_Toc168052796)

[Slika 2.6. Brzina obrade podataka s XMLHttpRequest na maloj bazi 14](#_Toc168052797)

[Slika 2.7. Brzina dohvata podataka s Axios-om na maloj bazi 15](#_Toc168052798)

[Slika 2.8. Brzina obrade podataka s Axios-om na maloj bazi 15](#_Toc168052799)

[Slika 2.9. Brzina dohvata podataka s jquery AJAX na maloj bazi 17](#_Toc168052800)

[Slika 2.10. Brzina obrade podataka s jquery AJAX na maloj bazi 17](#_Toc168052801)

[Slika 2.11. Brzina dohvata podataka s Fetch promises chaining-om na velikoj bazi 19](#_Toc168052802)

[Slika 2.12. Brzina obrade podataka s Fetch promises chaining-om na velikoj bazi 20](#_Toc168052803)

[Slika 2.13. Brzina dohvata podataka s Fetch asikronim funkcijama na velikoj bazi 20](#_Toc168052804)

[Slika 2.14. Brzina obrade podataka s Fetch askironim funkacijama na velikoj bazi 21](#_Toc168052805)

[Slika 2.15. Brzina dohvata podataka s Axios-om na velikoj bazi 21](#_Toc168052806)

[Slika 2.16. Brzina obrade podataka s Axios-om na velikoj bazi 22](#_Toc168052807)

[Slika 2.17. Brzina dohvata podataka s XMLHttpRequest-om na velikoj bazi 22](#_Toc168052808)

[Slika 2.18. Brzina obrade podataka s XMLHttpRequest-om na velikoj bazi 23](#_Toc168052809)

[Slika 2.19. Brzina dohvata podataka s jquery AJAX-om na velikoj bazi 23](#_Toc168052810)

[Slika 2.20. Brzina obrade podataka s jQuery AJAX-om na velikoj bazi 24](#_Toc168052811)

[Slika 3.1. Prijava u sustav 86](#_Toc168052812)

[Slika 3.2. Registracija korisnika u sustav 87](#_Toc168052813)

[Slika 3.3. Moji eventi 87](#_Toc168052814)

[Slika 3.4. Dodavanje novog eventa. 88](#_Toc168052815)

[Slika 3.5. Lokacije 89](#_Toc168052816)

[Slika 3.6. Popis svih lokacija. 89](#_Toc168052817)

[Slika 3.7. Meniji 90](#_Toc168052818)

[Slika 3.8. Novi eventi. 91](#_Toc168052819)

[Slika 3.9. Obrada evenata 91](#_Toc168052820)

[Slika 3.10. Prikaz evenata za rolu Manager 92](#_Toc168052821)

[Slika 3.11. Popis korisnika 92](#_Toc168052822)

[Slika 3.12. Uređivanje korisnika 92](#_Toc168052823)

[Slika 3.13. Popis zaposlenika 93](#_Toc168052824)

[Slika 3.14. Dodavanje novog zaposlenika 93](#_Toc168052825)

[Slika 3.15. Upravljanje lokacijama 94](#_Toc168052826)

[Slika 3.16. Dodavanje nove lokacije 94](#_Toc168052827)

[Slika 3.17. Upravljanje menijima 95](#_Toc168052828)

[Slika 3.18. Dodavanje novog jela 95](#_Toc168052829)

[Slika 3.19. Upravljanje jelima 96](#_Toc168052830)

[Slika 3.20. Uređivanje jelima 96](#_Toc168052831)

Popis tablica

[Tablica 2.1. Rezultati ispitivanja na maloj bazi 18](#_Toc168067865)

[Tablica 2.2. Rezultati ispitivanja na velikoj bazi 24](#_Toc168067866)

[Tablica 3.1. Popis funkcionalnosti prema korisničkim zahtjevima. 27](#_Toc168067867)

[Tablica 3.2. Popis svih korisničkih uloga (rola). 28](#_Toc168067868)

[Tablica 3.3. Lista funkcionalnosti navedene kao slučajevi uporabe 30](#_Toc168067869)

[Tablica 3.4. Slučaj uporabe UC1 – Registracija vanjskih korisnika 32](#_Toc168067870)

[Tablica 3.5.Slučaj uporabe UC2 – Prijava Managera i registriranih korisnika 34](#_Toc168067871)

[Tablica 3.6. Slučaj uporabe UC3 – Kreiranje narudžbe za catering 36](#_Toc168067872)

[Tablica 3.7.Slučaj uporabe UC6 – Pregled narudžbi za registriranog korisnika 40](#_Toc168067873)

[Tablica 3.8. Slučaj uporabe UC7 – Uređivanje narudžbi 41](#_Toc168067874)

[Tablica 3.9. Slučaj uporabe UC8 – Ocjena završene narudžbe 42](#_Toc168067875)

[Tablica 3.10.Slučaj uporabe UC9 – Uređivanje osobnih podataka 43](#_Toc168067876)

[Tablica 3.11. Slučaj uporabe UC10 – Pregled narudžbi za Managera 45](#_Toc168067877)

[Tablica 3.12. Slučaj uporabe UC11 – Odluka o potvrdi narudžbe 47](#_Toc168067878)

[Tablica 3.13. Slučaj uporabe UC12 – Uvid u dostupne djelatnike 48](#_Toc168067879)

[Tablica 3.14. Slučaj uporabe UC13 – Generiranje predračuna 49](#_Toc168067880)

[Tablica 3.15. Slučaj uporabe UC14 – Uređivanje podataka o korisnicima 50](#_Toc168067881)

[Tablica 3.16. Slučaj uporabe PUC15 – Uređivanje podataka o zaposlenicima 52](#_Toc168067882)

[Tablica 3.17. Slučaj uporabe UC16 – Uređivanje podataka o dostupnim lokacijama 53](#_Toc168067883)

[Tablica 3.18. Slučaj uporabe UC17 – Uređivanje podataka iz jelovnika 54](#_Toc168067884)

Popis dijagrama

[Dijagram 3.1. Dijagram slučajeva uporabe 29](#_Toc168067885)

[Dijagram 3.2. Dijagram slijeda: Registracija u sustav 33](#_Toc168067886)

[Dijagram 3.3. Dijagram slijeda: Prijava u sustav 35](#_Toc168067887)

[Dijagram 3.4. Dijagram aktivnosti obuhvaća slučajeve uporabe UC3,UC4,UC5,UC6,UC7 i UC8 37](#_Toc168067888)

[Dijagram 3.5. Slučaj uporabe UC4 – Upisivanje broja gostiju i željenih jelovnika 38](#_Toc168067889)

[Dijagram 3.6.Slučaj uporabe UC5 – Biranje lokacije 39](#_Toc168067890)

[Dijagram 3.7. Dijagram aktivnosti za slučaj uporabe UC9 44](#_Toc168067891)

[Dijagram 3.8. Dijagram aktivnosti za upravljanje narudžbama od strane Managera obuhvaća sljedeće slučajeve uporabe UC10,UC11,UC12 i UC13 46](#_Toc168067892)

[Dijagram 3.9. Dijagram aktivnosti na slici obuhvaća sljedeće slučajeve uporabe UC14,UC15,UC16 i UC17. 51](#_Toc168067893)

[Dijagram 3.10. Dijagram prikaza arhitekture sustava 55](#_Toc168067894)

[Dijagram 3.11. Arhitektura sustava programske potpore 56](#_Toc168067895)

[Dijagram 3.12. Dijagram MVC arhitekture [7] 57](#_Toc168067896)

[Dijagram 3.13. Konceptualni model baze podataka 60](#_Toc168067897)

[Dijagram 3.14. Fizički model baze podatka 60](#_Toc168067898)

[Dijagram 3.15. Dijagram HTTP & Presentation 61](#_Toc168067899)

[Dijagram 3.16. Business Logic 61](#_Toc168067900)

[Dijagram 3.17. Data Access 62](#_Toc168067901)

[Dijagram 3.18. Domain objects 62](#_Toc168067902)

[Dijagram 3.19. Dijagram razreda za EventManager 68](#_Toc168067903)

[Dijagram 3.20. Dijagram razreda za MenuManager 69](#_Toc168067904)

[Dijagram 3.21. Dijagram razreda za UserManager 70](#_Toc168067905)

[Dijagram 3.22. Dijagram razreda za VenueManager 71](#_Toc168067906)

[Dijagram 3.23. Dijagram razreda za poslovne entitete 72](#_Toc168067907)

Popis kôdova

[Kôd 2.1. Primjer korištenja Fetch API-a 3](#_Toc168067908)

[Kôd 2.2. Primjer korištenja Fetch API-a s GET zahtjevom 4](#_Toc168067909)

[Kôd 2.3. Primjer korištenja Fetch API-ja s asinkronim funkcijama za GET zahtjev 5](#_Toc168067910)

[Kôd 2.4. Primjer upotrebe XMLHttpRequest 6](#_Toc168067911)

[Kôd 2.5. Primjer GET zahtjev pomoću Axiosa 7](#_Toc168067912)

[Kôd 2.6. Primjer korištenja jQuery AJAX-a 7](#_Toc168067913)

[Kôd 2.7. Kôd od proizvodi.html 9](#_Toc168067914)

[Kôd 2.8. Kôd za proizvodi.js 10](#_Toc168067915)

[Kôd 2.9. Kôd napisan za proizvodi\_fetch\_async.js 11](#_Toc168067916)

[Kôd 2.10. Kôd za proizvodi\_XMLHttpRequest.js 13](#_Toc168067917)

[Kôd 2.11. Kôd za proizvodi\_axios.js 15](#_Toc168067918)

[Kôd 2.12. Kôd za proizvodi\_jquery.js 16](#_Toc168067919)

[Kôd 2.13. Kôd za users.html 19](#_Toc168067920)

# Uvod

U današnjem svijetu, web aplikacije su postale sastavni dio svakodnevnog života i poslovanja. Njihova primjena obuhvaća širok raspon područja, od društvenih mreža i e-trgovine do poslovnih aplikacija i Internet stvari (IoT). Web aplikacije omogućuju korisnicima pristup informacijama i uslugama brzo i učinkovito, bilo gdje i bilo kada. Ključni element koji omogućuje ovakvu povezivost i interoperabilnost web aplikacija su API-ji (Application Programming Interfaces).

API-ji omogućuju siguran i skalabilan način komunikacije između različitih aplikacija i sustava. Korištenjem API-ja, web aplikacije mogu pristupiti podacima i funkcionalnostima s udaljenih servera, što omogućava razvoj složenih aplikacija bez potrebe za dupliciranjem postojećih resursa. Na primjer, aplikacija može koristiti API za integraciju funkcionalnosti poput plaćanja, autentifikacije korisnika ili dohvaćanja vremenskih podataka iz specijaliziranih servisa.

JavaScript, kao jedan od najpopularnijih programskih jezika za razvoj web aplikacija, nudi nekoliko metoda za rad s API-ima. Među najčešće korištenim metodama su XMLHttpRequest, Fetch API, Axios i jQuery AJAX. Svaka od ovih metoda ima svoje prednosti i nedostatke, te se primjenjuju ovisno o specifičnim potrebama i kontekstu aplikacije.

Ovaj diplomski rad ima za cilj istražiti i usporediti nekoliko popularnih metoda za rad s API-ima u JavaScriptu. Fokus je stavljen na njihove karakteristike, implementaciju i performanse. Posebna pažnja posvećena je analizi performansi ovih metoda u različitim scenarijima kako bi se pružile smjernice za odabir najprikladnije metode u konkretnim slučajevima. Osim teorijskog dijela, rad uključuje i praktični dio gdje se razvija web aplikacija za usluge cateringa koja koristi jednu od testiranih API metoda za komunikaciju s poslužiteljima.

U drugom poglavlju opisane su različite API metode u JavaScriptu, uključujući XMLHttpRequest, Fetch API, Axios i jQuery AJAX. Svaka metoda je detaljno objašnjena s primjerima implementacije i specifičnim slučajevima uporabe. U ovom poglavlju provodi se implementacija i testiranje API metoda. Opisuje se kako su provedena testiranja na maloj i velikoj bazi podataka, te su prikazani rezultati performansi svake metode. Ovo poglavlje uključuje detaljnu analizu rezultata testiranja i usporedbu različitih metoda.

U trećem poglavlju opisana je izrada web aplikacije za catering usluge. Detaljno su navedeni korisnički zahtjevi, funkcionalnosti aplikacije, arhitektura sustava te implementacija pojedinih modula. Ovo poglavlje također uključuje UML dijagrame i detalje o bazi podataka korištenoj u aplikaciji.

U četvrtom poglavlju daje se zaključak rada, gdje se sumiraju rezultati istraživanja i testiranja, te se iznose preporuke za budući rad i moguća poboljšanja aplikacije.

Zadnje poglavlje sadrži popis literature, uključujući sve korištene izvore kao što su knjige, znanstveni radovi, članci i online resursi koji su korišteni tijekom istraživanja i izrade diplomskog rada.

# API Metode u JavaScriptu

## Fetch API

Fetch API se smatra modernom metodom za izvođenje HTTP zahtjeva u JavaScriptu.

Osmišljen je za jednostavnu i efikasnu asinkronu komunikaciju s web serverima. Fetch se koristi za slanje HTTP zahtjeva prema serveru i dobavljanje odgovora, što ga čini ključnim alatom u radu s API-ima. Zahtjev može biti bilo koje vrste API-ja koji vraća podatke u JSON ili XML. Metoda fetch() zahtijeva jedan parametar, URL za zahtjev, i vraća promise [1].

Fetch API podržava dva glavna načina rukovanja odgovorima: pomoću *promise chaining* i *asinkronim funkcijama* *s await*.

### Fetch promise chaining

Fetch promise chaining se koristi kao metoda za obradu HTTP zahtjeva pomoću niza povezanih "obećanja" (promises) u JavaScriptu. Ovaj pristup omogućava sekvencijalno izvršavanje asinkronih operacija, što je ključno za rukovanje odgovorima servera i obradom podataka na strukturiran način. Kada se koristi Fetch API, svaki poziv metodi fetch() vraća obećanje koje se može koristiti za daljnje operacije nakon što je zahtjev završen. Obećanje se obrađuje metodom .then(), koja se može povezati s drugim .then() metodama, stvarajući lanac operacija poznat kao promise chaining. Ovaj pristup omogućava precizno rukovanje svakom fazom HTTP zahtjeva, uključujući provjeru odgovora, parsiranje podataka i obradu grešaka [2]. Primjer korištenja Fetch za GET zahtjev:

A computer code with text

Description automatically generated

Kôd 2.1. Primjer korištenja Fetch API-a

**Parametri (***engl. Parameters***):** Ova metoda zahtijeva jedan parametar i prihvaća dva parametra:

* **URL:** To je URL na koji treba uputiti zahtjev.
* **Opcije** *(engl. Options)*: To je niz svojstava. To je izborni parametar.

**Povratna vrijednost** (*engl. Return Value*): vraća *promise*, bez obzira na status. Povratni podaci mogu biti JSON ili XML, niz objekata ili jedan objekt.

Fetch API omogućava izvođenje različitih HTTP zahtjeva, a to su: GET, POST, PUT i DELETE. Svaka od ovih metoda ima svoju svrhu i koristi se za različite operacije s resursima na web serveru. U ovom radu bavit ćemo se samo s GET metodom [3].

Metoda GET se koristi za dohvaćanje podataka s web servera. To je najčešća metoda za čitanje podataka s API-ja. Primjer korištenja Fetch za GET zahtjev:

A white screen with black text

Description automatically generated

Kôd 2.2. Primjer korištenja Fetch API-a s GET zahtjevom

### Fetch s asinkronim funkcijama

Korištenje Fetch API-ja s asinkronim funkcijama async/await je moderan pristup koji čini asinkroni JavaScript kod lakšim za čitanje i razumijevanje. Asinkrone funkcije omogućavaju pisanje koda koji izgleda i ponaša se kao sinkroni kod, eliminirajući potrebu za chainingom obećanja i olakšavajući rukovanje asinkronim operacijama [4].

Asinkrone funkcije koriste ključnu riječ async za označavanje funkcije kao asinkrone i await za čekanje na završetak obećanja. Kada se await koristi unutar asinkrone funkcije, izvršavanje funkcije se pauzira dok obećanje ne bude ispunjeno, nakon čega se vraća rezultat obećanja. Ovaj pristup omogućava linearnu strukturu koda, što ga čini intuitivnijim i lakšim za održavanje.

Primjer korištenja Fetch API-ja s asinkronim funkcijama za GET zahtjev može izgledati ovako [5]:

A computer code on a white background

Description automatically generated

Kôd 2.3. Primjer korištenja Fetch API-ja s asinkronim funkcijama za GET zahtjev

U ovom primjeru, funkcija *fetchData* je označena kao asinkrona korištenjem async ključne riječi. Unutar ove funkcije, await se koristi za čekanje na završetak fetch() obećanja. Ako je odgovor uspješan (response.ok), podaci se parsiraju u JSON formatu i prosljeđuju se za daljnju obradu. U slučaju greške, ona se hvata u catch bloku i ispisuje se poruka o grešci.

## XMLHttpRequest

Starija tehnologija za asinkronu komunikaciju s API-ima zove se *XMLHttpRequest*. Koristi se za slanje HTTP zahtjeva i obradu odgovora na tradicionalan način omogućujući web stranicama dinamičku interakciju s serverima bez potrebe za ponovnim učitavanjem cijele stranice. Iako su moderniji API-ji poput Fetch-a preuzeli primat zbog svoje jednostavnosti i fleksibilnosti, XMLHttpRequest se još uvijek koristi u mnogim aplikacijama zbog svoje široke podrške i pouzdanosti [6].

XMLHttpRequest omogućava slanje različitih vrsta HTTP zahtjeva, uključujući GET, POST, PUT i DELETE, te obradu odgovora koji mogu biti u raznim formatima poput teksta, XML-a ili JSON-a. Korištenje XMLHttpRequest-a uključuje nekoliko koraka: kreiranje novog xhre objekta, konfiguriranje zahtjeva, slanje zahtjeva, te rukovanje odgovorima i greškama.

Primjer upotrebe XMLHttpRequest [7]:

A black text on a white background

Description automatically generated

A white background with black text

Description automatically generated

Kôd 2.4. Primjer upotrebe XMLHttpRequest

U ovom primjeru, prvo se kreira novi XMLHttpRequest objekt. Metoda xhre.open() konfigurira zahtjev, postavljajući HTTP metodu (u ovom slučaju GET), URL i asinkroni način izvršavanja (true). Metoda xhre.onreadystatechange postavlja funkciju koja će se pozvati svaki put kada se stanje zahtjeva promijeni. xhre.readyState prati različite faze zahtjeva, dok xhre.status pokazuje HTTP statusni kod odgovora.

Unatoč svojim nedostacima, XMLHttpRequest ostaje koristan alat za razvoj web aplikacija, posebno u situacijama kada je potrebna podrška za starije preglednike ili kada postojeće aplikacije već koriste ovaj način komunikacije.

## Axios

Axios je popularna JavaScript biblioteka koja se koristi za izvođenje HTTP zahtjeva i koja pruža jednostavnost, fleksibilnost i dodatne mogućnosti za rad s API-ima.

Jedna od glavnih prednosti korištenja Axios-a je što podržava obećanja (promises), što olakšava pisanje i rukovanje asinkronim kodom. Osim toga, Axios automatski transformira podatke odgovora u JSON format, pojednostavljujući proces obrade podataka. Također nudi brojne korisne značajke kao što su interceptori zahtjeva i odgovora, podrška za prekide zahtjeva, automatsko otkrivanje XSRF tokena i još mnogo toga.

Primjer HTTP zahtjev pomoću Axiosa [8]:

A computer code with black text

Description automatically generated

Kôd 2.5. Primjer GET zahtjev pomoću Axiosa

U ovom primjeru, metoda axios.get šalje GET zahtjev na zadani URL. Ako je zahtjev uspješan, odgovor se obrađuje u .then() metodi, a podaci iz odgovora se ispisuju na konzolu. Ako se dogodi greška, ona se hvata u .catch() metodi i ispisuje se poruka o grešci.

## jQuery AJAX

jQuery AJAX je skup metoda i funkcija unutar jQuery biblioteke koje omogućavaju jednostavnu i efikasnu komunikaciju s udaljenim serverima bez potrebe za osvježavanjem cijele web stranice. Ovaj pristup omogućava dinamičko učitavanje i slanje podataka, čime se poboljšava korisničko iskustvo i interaktivnost web stranica [9].

Jedna od glavnih prednosti korištenja jQuery AJAX-a je njegova jednostavnost i integracija s jQuery bibliotekom. jQuery AJAX metode su intuitivne i pružaju jednostavan način za slanje HTTP zahtjeva, rukovanje odgovorima i upravljanje greškama. Najčešće korištena metoda je $.ajax(), koja omogućava slanje zahtjeva s različitim postavkama [10].

A computer code with text

Description automatically generated

Kôd 2.6. Primjer korištenja jQuery AJAX-a

U ovom primjeru, metoda $.ajax() šalje GET zahtjev na zadani URL. Ako je zahtjev uspješan, podaci se obrađuju u success funkciji i ispisuju na konzolu. Ako se dogodi greška, ona se obrađuje u error funkciji, gdje se ispisuje poruka o grešci.

Prednosti jQuery AJAX su jednostavnost korištenja, integracija s jQuery-om, široka podrška preglednika i razne metode koje dodatno pojednostavljuju ras s različitim vrstama zahtjeva.

Nedostaci jQuery AJAXje veličina biblioteke koja može povećati veličinu projekta, ne korištenje obećanja (promises) prema zadanim postavkama.

## Implementacija API-a

Planira se provesti testiranje na dvije različite baze podataka. Prva baza sadrži manji broj podataka i dostupna je putem endpointa <https://demo.cadcam-group.eu/api/>.

Ovom bazom će se dohvatiti naziv artikla, sadržaj i tip artikla, te će se dodati stupac "Red. Broj", gdje će se inkrementalno povećavati redni brojevi. Ukupno ima 40 artikala.

Druga baza podataka sadrži znatno više podataka. Isto tako, koristit će se iste metode API-ja za dohvaćanje. Ova baza bit će korištena za ocjenu performansi metoda API-ja na većem skupu podataka. Baza je dostupna putem endpointa: <https://my.api.mockaroo.com/users.json?key=d2d5f990>

Koristit će se samo GET metoda sa sljedećim API metodama:

* Fetch s promises chaining-om
* Fetch s asinkronim funkcijama
* XMLHttpRequest
* Axios
* jQuery AJAX

### Prvo testiranje s manjom bazom

Za sve primjere koristit ćemo *proizvodi.html* gdje je napisan HTML kod, a javaScript kôd će biti napisan u zasebnom file-u za svaki primjer.

Kôd od *proizvodi.html*:

A screenshot of a computer code

Description automatically generated

A computer code with black text

Description automatically generated

Kôd 2.7. Kôd od proizvodi.html

#### Korištenje Fetch promises chaining

U ovom primjer koristit ćemo Fetch promises chaining kako bi dohvatili sve artikle.

Kôd napisan za *proizvodi.js*:

A white text with black text

Description automatically generated with medium confidence

Kôd 2.8. Kôd za proizvodi.js

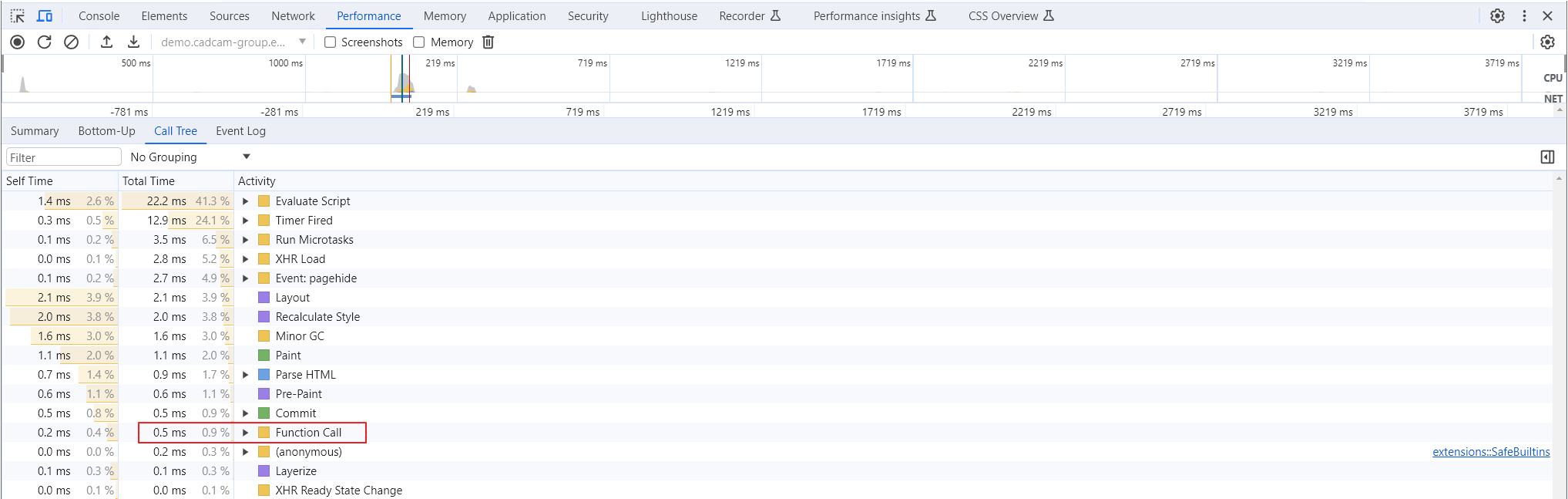
Brzina dohvata podataka s API-ja je iznosila 22 ms.

A screenshot of a computer

Description automatically generated

Slika 2.1. Brzina dohvata podataka s Fetch promises chaining na maloj bazi

Brzina obrade podataka unutar poziva Fetch metode iznosila je 0,2 ms.



Slika 2.2. Brzina obrade podataka s Fetch promises chaining na maloj bazi

#### Korištenje Fetch API s asikronim funkcijama

U ovom primjeru koristit će se Fetch API s *async/await* kako bi dohvatili sve artikle. Kôd napisan za *proizvodi\_fetch\_async.js*:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

Kôd 2.9. Kôd napisan za proizvodi\_fetch\_async.js

Brzina izvršavanja API-ja iznosila je 24 ms.

A screenshot of a computer

Description automatically generated

Slika 2.3. Brzina dohvata podataka s Fetch asikronim funkcijama na maloj bazi

Brzina obrade podataka unutar poziva Fetch-a s asikronim funkcijama iznosila je 0,3 ms.A screenshot of a computer

Description automatically generated

Slika 2.4. Brzina obrade podataka s Fetch asikronim funkcijama na maloj bazi

#### Korištenje XMLHttpRequest

U ovom primjer koristit ćemo XMLHttpRequest kako bi dohvatili sve artikle.

Kôd napisan za *proizvodi\_XMLHttpRequest.js*:

A screenshot of a computer code

Description automatically generated

Kôd 2.10. Kôd za proizvodi\_XMLHttpRequest.js

Brzina dohvata podataka s XMLHttpRequest iznosila je 33 ms.

A screenshot of a computer

Description automatically generated

Slika 2.5. Brzina dohvata podataka s XMLHttpRequest na maloj bazi

Brzina obrade podataka iznosila je 0,4 ms.

A screenshot of a computer

Description automatically generated

Slika 2.6. Brzina obrade podataka s XMLHttpRequest na maloj bazi

#### Korištenje Axios-a

U ovom primjer koristit će se Axios za dohvaćanje svih artikala. Axios je potrebno prvo instalirati u projektu, a to se može preko *npm* ili *yarn*-a. U slučaju *npm-a*, u terminalu treba upisati '*npm install axios'*, a uz pomoću *yarn*-a, instalira se preko naredbe '*yarn add axios'*. Ako se ne želi instalirati Axios, može ga se uključiti iz vanjskog CDN (*Content Delivery Network*) izravno u HTML datoteku putem <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>. Uključivanje Axios putem CDN-a, knjižnica će se preuzeti iz vanjskog izvora kada posjetitelji otvore web stranicu, umjesto iz lokalnih paketa na računalu.

Kôd napisan za *proizvodi\_axios.js*:

A screenshot of a computer code

Description automatically generated

A white screen with black text

Description automatically generated

Kôd 2.11. Kôd za proizvodi\_axios.js

Brzina dohvata podataka iznosila je 21 ms.

A screenshot of a computer

Description automatically generated

Slika 2.7. Brzina dohvata podataka s Axios-om na maloj bazi

Brzina obrade podataka iznosila je 0,2 ms.

A screenshot of a computer

Description automatically generated

Slika 2.8. Brzina obrade podataka s Axios-om na maloj bazi

#### Korištenje jquery AJAX

U ovom primjer koristit će se jQuery AJAX za dohvaćanje svih artikala. jQuery AJAX može se uključiti iz vanjskog CDN (Content Delivery Network) izravno u HTML datoteku putem:

<script src="https://code.jquery.com/jquery3.6.0.min.js"></script>

Kôd napisan za *proizvodi\_jquery.js*:

A computer code with black text

Description automatically generated

Kôd 2.12. Kôd za proizvodi\_jquery.js

Brzina dohvata podataka je iznosila 23 ms.

A screenshot of a computer

Description automatically generated

Slika 2.9. Brzina dohvata podataka s jquery AJAX na maloj bazi

Brzina obrade podataka je iznosila 7,5 ms.

A screenshot of a computer

Description automatically generated

Slika 2.10. Brzina obrade podataka s jquery AJAX na maloj bazi

#### Pregled rezultata s prvom bazom

U tablici 1. napisani su rezultati testiranja za dohvat i obradu podataka prema svakoj testiranoj metodi.

Tablica 2.1. Rezultati ispitivanja na maloj bazi

|  |  |  |
| --- | --- | --- |
| Metoda | Brzina dohvata podataka (ms) | Brzina obrade podataka (ms) |
| Fetch promises chaining | 22 | 0,2 |
| Fetch s asinkronim funkcijama | 24 | 0,3 |
| XMLHttpRequest | 33 | 0,4 |
| Axios | 21 | 0,2 |
| jQuery AJAX | 23 | 7,5 |

Na temelju provedenog istraživanja brzine dohvata i obrade podataka korištenjem različitih metoda za komunikaciju s poslužiteljem, mogu se izvući neki zaključci.

Prvo, Fetch promises chaining i Axios su se pokazali kao najbrže metode za dohvat podataka s prosječnim vremenom između 21 i 22 ms. Osim toga, imaju brzu obradu podataka s vremenom od 0,2 ms. Ove dvije metode su prilično učinkovite i brze za komunikaciju s poslužiteljem.

XMLHttpRequest i Fetch s asinkronim funkcijama imaju nešto sporiji dohvat podataka, s prosječnim vremenima od 24 i 33 ms, ali i dalje su prihvatljivi za većinu primjena. Brzina obrade podataka kod ovih metoda iznosi 0,3 odnosno 0,4 ms, što je također pristojno.

Međutim, jQuery AJAX se izdvaja po izrazitoj sporosti u obradi podataka, s vremenom od 7,5 ms, što je znatno sporije od svih ostalih metoda, dok je brzina dohvata podataka od 23 ms poprilično ista kao i kod Fetch promises chaining i Axios.

### Ponovljeno testiranje na bazi s više podataka

Baza je kreirana preko web aplikacije '*Mockaroo*' koja služi kao besplatan generator testnih podataka. Iz baze se dohvaćaju podaci o korisnicima kao što su: *id*, *first\_name*, *last\_name* i *email*. Baza sadrži 1000 redaka.

Za sve primjere koristit će se *users.html* gdje je napisan HTML kôd, a javaScript kôd će biti napisan u zasebnom file-u za svaki primjer.

Kôd od *users.html*:

A screen shot of a computer code

Description automatically generated

A computer code with text

Description automatically generated

Kôd 2.13. Kôd za users.html

JavaScript kôd neće biti prikazan budući da je vrlo sličan prethodnom kodu iz prvog testiranja. Umjesto toga, fokus će biti stavljen samo na rezultate testiranja.

#### Rezultati testiranja

1. Fetch s promises chaining-om

Brzina dohvata podataka iznosila je 1,3 sekunde.

A screenshot of a computer

Description automatically generated

Slika 2.11. Brzina dohvata podataka s Fetch promises chaining-om na velikoj bazi

Brzina obrade podataka unutar poziva Fetch metode iznosila je 0,5 ms.

A screenshot of a computer

Description automatically generated

Slika 2.12. Brzina obrade podataka s Fetch promises chaining-om na velikoj bazi

1. Fetch s asikronim funkcijama

Brzina dohvata podataka iznosila je 1,2 sekunde.

A screenshot of a computer

Description automatically generated

Slika 2.13. Brzina dohvata podataka s Fetch asikronim funkcijama na velikoj bazi

Brzina obrade podataka iznosila je 0,4 ms.

A screenshot of a computer

Description automatically generated

Slika 2.14. Brzina obrade podataka s Fetch askironim funkacijama na velikoj bazi

1. Axios

Brzina dohvata podataka iznosila je 1,17 sekunde.

A screenshot of a computer

Description automatically generated

Slika 2.15. Brzina dohvata podataka s Axios-om na velikoj bazi

Brzina obrade podataka iznosila je 0,4 ms.

A screenshot of a computer

Description automatically generated

Slika 2.16. Brzina obrade podataka s Axios-om na velikoj bazi

1. XmlHttpRequest

Brzina dohvata podataka iznosila je 1,91 sekunde.

A screenshot of a computer

Description automatically generated

Slika 2.17. Brzina dohvata podataka s XMLHttpRequest-om na velikoj bazi

Brzina obrade podataka iznosila je 21,8 ms.

A screenshot of a computer

Description automatically generated

Slika 2.18. Brzina obrade podataka s XMLHttpRequest-om na velikoj bazi

1. jQuery AJAX

Brzina dohvata podataka iznosila je 1,73 sekunde.

A screenshot of a computer

Description automatically generated

Slika 2.19. Brzina dohvata podataka s jquery AJAX-om na velikoj bazi

Brzina obrade podataka iznosila je 106,8 ms.

A screenshot of a computer

Description automatically generated

Slika 2.20. Brzina obrade podataka s jQuery AJAX-om na velikoj bazi

#### Usporedba rezultata

U tablici 2. napisani su rezultati testiranja za dohvat i obradu podataka prema svakoj testiranoj metodi za veliku bazu podataka.

Tablica 2.2. Rezultati ispitivanja na velikoj bazi

|  |  |  |
| --- | --- | --- |
| Metoda | Brzina dohvata podataka (s) | Brzina obrade podataka (ms) |
| Fetch promises chaining | 1,3 | 0,5 |
| Fetch s asikronim funkcijama | 1,2 | 0,4 |
| XMLHttpRequest | 1,91 | 21,8 |
| Axios | 1,17 | 0,4 |
| jQuery AJAX | 1,73 | 106,8 |

Axios, Fetch promises chaining i s asikronim funkcijama pokazali su se kao najbrže metode za dohvat i obradu podataka. To ukazuje na njihovu visoku učinkovitost i brzinu u usporedbi s drugim metodama.

XMLHttpRequest ima najsporiji dohvat podataka s vremenom od 1,91 sekunde, dok je njegova brzina obrade podataka druga najsporija metoda za obradu podataka.

jQuery AJAX se izdvaja kao najsporija metoda za obradu podataka, dok je druga najsporija metoda za dohvat podataka od 1,73 ms.

# Izrada web aplikacije Catering

## Opis aplikacije

Svrha aplikacije je olakšavanje usluge cateringa, kako kupcu tako i pružatelju usluga. Korisnici koji žele usluge cateringa, a nemaju otvoren račun, moraju se prvo registrirati. Korisnici usluga cateringa koji imaju postojeći račun trebaju se najprije prijaviti u sustav. Prilikom kreiranja narudžbe prijavljeni korisnik mora upisati broj gostiju te broj željenih menija. Slanje narudžbe uključuje i biranje unaprijed definirane lokacije za catering unutar slobodnog termina. Prijavljeni korisnici mogu pregledavati svoje narudžbe, modificirati narudžbe koje nisu aktivne i završene te ocijeniti završene narudžbe. Prijavljeni korisnici mogu uređivati svoje osobne podatke.

Manager treba imati mogućnost pregledavanja svih narudžbe te aktivne narudžbe odobriti ili odbiti. Manager za svaku aktivnu narudžbu može pregledati raspoložive djelatnike za odabrani termin narudžbe. Manager treba imati mogućnost ispisivanja računa za uslugu cateringa u PDF formatu. Manager treba imati mogućnost pregleda i izmjene zaposlenika, korisnika, lokacija i menija.

## Pojmovnik

* *Catering* - vrsta ugostiteljske usluge koja se brine za opskrbu hrane i pića uzvanika, gostiju na različitim društvenim događajima i zabavama
* *PDF* – skraćenica od „Portable Document Format“ - je format zapisa dokumenata kojeg je kreirala kompanija Adobe Systems 1993. godine. Koristi se za zapis dvodimenzionalnih dokumenata, nezavisno od uređaja i rezolucije ispisa.
* *Manager* – osoba koja upravlja sustavom za catering
* *Venue* – lokacija gdje se odvija catering
* *Event* – u kontekstu aplikacije radi se o narudžbi
* *Employee* – zaposlenik
* *Dish* – jelo
* *Menu* – jelovnik
* *Skill* – vještina
* *User* - korisnik

## Definicija korisničkih zahtjeva

Popis funkcionalnosti prema korisničkim zahtjevima prikazat ćemo u sljedećoj tablici.

Tablica 3.1. Popis funkcionalnosti prema korisničkim zahtjevima.

| ***R. br.*** | ***Zahtjev*** | ***Kratki opis zahtjeva*** | ***UC br.*** |
| --- | --- | --- | --- |
| 1 | Registracija vanjskih korisnika | Svaki posjetitelj web stranice cateringa DAR-MAR mora imati mogućnost registracije na web stranicama | UC1 |
| 2 | Prijava Managera i registriranih korisnika | Web aplikacija mora omogućavati prijavu za Managera i registrirane korisnike | UC2 |
| 3 | Kreiranje narudžbe za catering | Registrirani korisnik mora imat mogućnost kreiranja narudžbe za catering | UC3 |
| 4 | Upisivanje broja gostiju i željenih jelovnika | Registrirani prilikom kreiranja narudžbe mora upisati broj gostiju i broj željenih jelovnika | UC4 |
| 5 | Biranje lokacije | Korisnik prilikom kreiranja narudžbe mora odabrati željenu lokaciju za catering unutar slobodnog termina | UC5 |
| 6 | Pregled narudžbi za registriranog korisnika | Registrirani korisnik može pregledavati sve svoje narudžbe. Narudžbe koje još nisu aktivne mogu se mijenjati. | UC6 |
| 7 | Uređivanje narudžbi | Narudžbe koje još nisu aktivne mogu se mijenjati. | UC7 |
| 8 | Ocjena završene narudžbe | Registrirani korisnik treba imati mogućnost ocjena završene narudžbe. | UC8 |
| 9 | Uređivanje osobnih podataka | Registrirani korisnik može uređivati svoje podatke | UC9 |
| 10 | Pregled narudžbi za managera | Manager mora imati mogućnost pregleda i pretraživanja svih narudžbi. | UC10 |
| 11 | Odluka o potvrdi narudžbe | Manager može odobriti ili odbiti narudžbu. Kada odobri narudžbu, narudžba postaje aktivna. | UC11 |
| 12 | Uvid u dostupne djelatnike | Prilikom odluke o narudžbi Manager mora imati uvid u raspoložive djelatnike. | UC12 |
| 13 | Generiranje predračuna | Manager može generirati predračun i ispisati ga u PDF format dokumenta | UC13 |
| 14 | Uređivanje podataka o korisnicima | Manager mora imati mogućnost pregleda i uređivanja podataka za registrirane korisnike. | UC14 |
| 15 | Uređivanje podataka o zaposlenicima | Manager mora imati mogućnost pregleda i uređivanja podataka o zaposlenicima. | UC15 |
| 16 | Uređivanje podataka o dostupnim lokacijama | Manager mora imati mogućnost pregleda i uređivanja podataka o dostupnim lokacijama za catering. | UC16 |
| 17 | Uređivanje podataka iz jelovnika | Sustav manageru treba omogućiti upravljanje jelovnikom. | UC17 |

Popis svih korisničkih uloge (rola) nalazi se u sljedećoj tablici.

Tablica 3.2. Popis svih korisničkih uloga (rola).

| ***R. br.*** | ***Uloge*** | ***Opis uloge*** | ***UC br.*** |
| --- | --- | --- | --- |
| 1 | Gost | Posjetitelj web stranica koji još nije registriran u sustavu | UC1 |
| 2 | Registrirani korisnik | Korisnik sustava koji ima registrirani račun u sustavi i koji se uspješno prijavio u sustav | UC2, UC3, UC4, UC5, UC6, UC7, UC8, UC9 |
| 3 | Manager | Osoba koja upravlja sustavom za catering | UC2, UC10, UC11, UC12, UC13, UC14, UC15, UC16, UC17 |

Na temelju opisanih zahtjeva izrađen je dijagram slučajeva uporabe. Dijagram prikazuje različite aktivnosti koje obavljaju korisnici, gosti i menadžeri unutar sustava za catering. Prikazane su sve ključne funkcionalnosti, uključujući registraciju, prijavu, kreiranje narudžbi, pregled i uređivanje narudžbi te administraciju podataka o korisnicima, lokacijama i menijima. Ovim dijagramom vizualno je predstavljena interakcija korisnika sa sustavom kako bi se olakšalo razumijevanje i implementacija zahtjeva.

Dijagram 3.1. Dijagram slučajeva uporabe

A diagram of a diagram

Description automatically generated

## Specifikacija zahtjeva sustava

Na temelju prikupljenih zahtjeva izrađena je tablica s listom funkcionalnosti, gdje su navedeni svi slučajevi uporabe s odgovarajućim opisima. Ova tablica omogućava pregled svih ključnih aktivnosti unutar sustava za catering

Tablica 3.3. Lista funkcionalnosti navedene kao slučajevi uporabe

|  |  |  |
| --- | --- | --- |
| ***Oznaka*** | ***Naziv*** | ***Opis*** |
| UC1 | Registracija vanjskih korisnika | Svaki posjetitelj ima mogućnost registracije u sustav kako bi se kasnije mogao prijaviti u sustav |
| UC2 | Prijava Managera i registriranih korisnika | Manager i registrirani korisnici se najprije trebaju prijaviti u sustav kako bi bilo šta mogli raditi. |
| UC3 | Kreiranje narudžbe za catering | Registrirani korisnik kreira narudžbu za catering. |
| UC4 | Upisivanje broja gostiju i željenih jelovnika | Prilikom kreiranja narudžbe registrirani korisnik upisuje broj gostiju i broj željenih jelovnika |
| UC5 | Biranje lokacije | Prilikom kreiranja narudžbe registrirani korisnik bira unaprijed definiranu lokaciju za catering unutar slobodnog termina. |
| UC6 | Pregled narudžbi za registriranog korisnika | Registrirani korisnik može pregledavati sve svoje narudžbe. Narudžbe koje još nisu aktivne mogu se mijenjati. |
| UC7 | Uređivanje narudžbi | Narudžbe koje još nisu aktivne mogu se mijenjati. |
| UC8 | Ocjena završene narudžbe | Narudžbe koje su završene mogu se ocjenjivati |
| UC9 | Uređivanje osobnih podataka | Registrirani korisnik može uređivati svoje podatke |
| UC10 | Pregled narudžbi za managera | Manager može pregledavati sve narudžbe |
| UC11 | Odluka o potvrdi narudžbe | Manager može odobriti ili odbiti narudžbu. Kada odobri narudžbu, narudžba postaje aktivna. |
| UC12 | Uvid u dostupne djelatnike | Prilikom odluke o narudžbi Manager ima uvid u raspoložive djelatnike. |
| UC13 | Generiranje predračuna | Manager može generirati predračun i ispisati ga u PDF format dokumenta |
| UC14 | Uređivanje podataka o korisnicima | Manager može pregledavati i uređivati podatke o registriranim korisnicima. |
| UC15 | Uređivanje podataka o zaposlenicima | Manager može pregledavati i uređivati podatke o zaposlenicima. |
| UC16 | Uređivanje podataka o dostupnim lokacijama | Manager može pregledavati i uređivati podatke o dostupnim lokacijama. |
| UC17 | Uređivanje podataka iz jelovnika | Manager može pregledavati i uređivati podatke iz jelovnika. |

### Slučaj uporabe UC1 – Registracija vanjskih korisnika

Tablica 3.4. Slučaj uporabe UC1 – Registracija vanjskih korisnika

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Registracija vanjskih korisnika |
| **Namjena:** | Preduvjet za korištenje mogućnosti koje sustav pruža |
| **Uloge (*role*):** | Gost |
| **Opis:** | Svaki posjetitelj web stranice ima mogućnost registracije u sustav kako bi se kasnije mogao prijaviti u sustav |
| **Slijed aktivnosti:** | Gost pritisne poveznicu „Registracija“ koji ga zatim odvede do prozora za registraciju gdje gost popunjava osobne podatke te postavlja željeno korisničko ime i zaporku. Nakon uspješne registracije gost se preusmjerava na prozor za prijavu. |
| **Sadržaj podataka:** | email, lozinka, ime, prezime, telefon |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Omogućen pristup stranici za prijavu; Gost ne smije biti registriran u sustavu |
| **Posljedice:** | Kreiranje korisničkog računa u sustavu te mogućnost korištenja funkcionalnosti sustava |
| **Počinje kada?** | Pritiskom na gumb za registraciju |
| **Završava kada?** | Kada je primljena obavijest u uspješnoj registraciji |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

Na temelju tablice koja opisuje slučaj uporabe "Registracija vanjskih korisnika" izrađen je dijagram slijeda 3.2. Ovaj dijagram vizualno prikazuje korake koje gost prolazi tijekom procesa registracije na sustav.

Dijagram 3.2. Dijagram slijeda: Registracija u sustav

A diagram of a project

Description automatically generated

### Slučaj uporabe UC2 – Prijava Managera i registriranih korisnika

Tablica 3.5.Slučaj uporabe UC2 – Prijava Managera i registriranih korisnika

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Prijava Managera i registriranih korisnika |
| **Namjena:** | Neophodno za korištenje mogućnosti koje sustav pruža |
| **Uloge (*role*):** | Registrirani korisnik, Manager |
| **Opis:** | Kako bi registrirani korisnici ili Manager mogli koristiti sve funkcionalnosti koje sustav pruža, moraju se najprije prijaviti u sustav |
| **Slijed aktivnosti:** | Korisnik na prozoru za prijavu unosi e-mail adresu i lozinku. Ako su podaci ispravni korisnik se preusmjerava na svoje korisničke stranice, a u slučaju da podaci nisu ispravni korisnik prima obavijest o neuspješnoj prijavi. |
| **Sadržaj podataka:** | email, lozinka |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Korisnik je već registriran u sustavu |
| **Posljedice:** | Omogućen pristup funkcionalnostima sustava |
| **Počinje kada?** | Kada je pritisnut gumb „Prijava“ |
| **Završava kada?** | Kada je pritisnut gumb „Odjava“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

Na temelju tablice za slučaj uporabe "Prijava Managera i registriranih korisnika" izrađen je dijagram slijeda. Prikazani su koraci prijave, provjere podataka i povratne informacije o uspješnosti. Dijagram jasno prikazuje interakcije korisnika i sustava, olakšavajući implementaciju i provjeru procesa prijave.

Dijagram 3.3. Dijagram slijeda: Prijava u sustav

A diagram of a diagram

Description automatically generated

### Slučaj uporabe UC3 – Kreiranje narudžbe za catering

Tablica 3.6. Slučaj uporabe UC3 – Kreiranje narudžbe za catering

|  |  |
| --- | --- |
| Naziv slučaja uporabe: | Kreiranje narudžbe za catering |
| Namjena: | Kreiranje narudžbe za catering |
| Uloge (role): | Registrirani korisnik |
| Opis: | Registrirani korisnik kreira narudžbu za catering |
| Slijed aktivnosti: | Pritiskom na gumb „dodaj“ otvara se prozor za upis podataka o narudžbi: datum, vrijeme početka, vrijeme završetka, odabire se lokacija, broj gostiju te broj željenih menija. Kreiranje narudžbe završava pritiskom na gumb „Spremi“. |
| Sadržaj podataka: | Broj gostiju, Lista odabranih jelovnika s količinama, Odabrana lokacija za catering, vrijeme cateringa |
| Transakcija: |  |
| Aplikacija: |  |
| Preduvjeti: | Korisnik je prijavljen u sustav |
| Posljedice: | Narudžba je spremljena u bazu podataka sustava |
| Počinje kada? | Prilikom pritiska na gumb „Dodaj“ |
| Završava kada? | Prilikom pritiska na gumb „Spremi“ |
| Izuzeci: |  |
| Veza s drugim UC (ID): | UC4, UC5 |
| Otvorena pitanja: |  |
| Prioritet: | MUST |

Na temelju tablice za slučaj uporabe "Kreiranje narudžbe za catering" izrađen je dijagram aktivnosti. Dijagram prikazuje korake unosa, provjere i spremanja podataka o narudžbi. Omogućava jasan pregled procesa, olakšavajući implementaciju i provjeru upravljanja narudžbama.

Dijagram 3.4. Dijagram aktivnosti obuhvaća slučajeve uporabe UC3,UC4,UC5,UC6,UC7 i UC8

A diagram of a diagram

Description automatically generated

### Slučaj uporabe UC4 – Upisivanje broja gostiju i željenih jelovnika

U tablici je opisan slučaj uporabe UC4 – "Upisivanje broja gostiju i željenih jelovnika". Ovaj slučaj definira postupak unosa ključnih informacija za kreiranje narudžbe, gdje registrirani korisnik upisuje broj gostiju i željene jelovnike.

Tablica 3.7. Slučaj uporabe UC4 – Upisivanje broja gostiju i željenih jelovnika

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Upisivanje broja gostiju i željenih jelovnika |
| **Namjena:** | Informacija o broju gostiju koji će sudjelovati za vrijeme cateringa te informacija o broju željenih jelovnika |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Registrirani korisnik prilikom kreiranja narudžbe mora upisati broj gostiju te broj željenih jelovnika |
| **Slijed aktivnosti:** | Upisuje se broj gostiju i broj željenih menija |
| **Sadržaj podataka:** | Broj gostiju, broj željenih jelovnika |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Registrirani korisnik je prijavljen, postoje jelovnici u bazi podataka |
| **Posljedice:** | Narudžba sadrži informaciju o broju gostiju i broju željenih jelovnika |
| **Počinje kada?** | Prilikom unosa broja gostiju i željenih jelovnika |
| **Završava kada?** | Prilikom završetka unosa broja gostiju i željenih jelovnika |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** | UC3 |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC5 – Biranje lokacije

U tablici je opisan slučaj uporabe UC5 – "Biranje lokacije". Ovaj slučaj definira postupak odabira lokacije za catering, gdje registrirani korisnik prilikom kreiranja narudžbe odabire jednu od slobodnih lokacija unutar dostupnog vremena.

Tablica 3.8. Slučaj uporabe UC5 – Biranje lokacije

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Biranje lokacije |
| **Namjena:** | Informacija o lokaciji izvršenja cateringa |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Registrirani korisnik prilikom kreiranja narudžbe mora odabrati lokaciju za catering unutar slobodnog vremena za tu lokaciju |
| **Slijed aktivnosti:** | Korisnik odabire iz padajućeg izbornika jednu od slobodnih lokacija, unutar odabranog vremena, za catering |
| **Sadržaj podataka:** | Lokacija za catering, datum i vrijeme |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Postojanje unaprijed definiranih lokacija za catering |
| **Posljedice:** | Lokacija za catering je odabrana |
| **Počinje kada?** | Prilikom pritiska na padajući izbornik s lokacijama |
| **Završava kada?** | Prilikom odabira jedne od lokacija |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** | UC3 |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC6 – Pregled narudžbi za registriranog korisnika

U tablici je opisan slučaj uporabe UC6 – "Pregled narudžbi za registriranog korisnika". Ovaj slučaj definira mogućnost pregleda svih narudžbi registriranog korisnika, bez obzira na njihov status. Nakon prijave, korisnik može vidjeti svoje narudžbe, podatke i status obrade na početnom ekranu.

Tablica 3.9.Slučaj uporabe UC6 – Pregled narudžbi za registriranog korisnika

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Pregled narudžbi za registriranog korisnika |
| **Namjena:** | Informacija o svim narudžbama registriranog korisnika |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Registrirani korisnik ima mogućnost pregleda svih vlastitih narudžbi bez obzira na status (u obradi, aktivna, završena) |
| **Slijed aktivnosti:** | Korisnik nakon prijave dolazi na početni ekran gdje može vidjeti svoje narudžbe, podatke na njima te status obrade |
| **Sadržaj podataka:** | Datum i vrijeme, broj gostiju, broj željenih jelovnika, lokacija, datum i vrijeme lokacije |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Zapis o narudžbi postoji u bazi podataka sustava |
| **Posljedice:** | Narudžbe prikazane na predviđenoj stranici |
| **Počinje kada?** | Netom nakon prijave u sustav |
| **Završava kada?** | Odjavom iz aplikacije |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC7 – Uređivanje narudžbi

U tablici je opisan slučaj uporabe UC7 – "Uređivanje narudžbi". Ovaj slučaj omogućava registriranom korisniku izmjenu podataka na narudžbama koje nisu aktivne ili završene.

Tablica 3.10. Slučaj uporabe UC7 – Uređivanje narudžbi

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje narudžbi |
| **Namjena:** | U slučaju da registrirani korisnik želi promijeniti podatke za narudžbu koja je u obradi tj. nije aktivna ili završena |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Narudžbe koje nisu aktivne ili završene mogu se uređivati odnosno mogu se mijenjati svi podaci koji se nalaze na narudžbi |
| **Slijed aktivnosti:** | Korisnik pritiskom na gumb ima mogućnost izmjene podataka na narudžbama koje su u statusu „Zaprimljeno“ odnosno koje nemaju statuse „Prihvaćeno“ ili „Odbijeno“ |
| **Sadržaj podataka:** | Svi podaci koji se nalaze prilikom kreiranja narudžbe |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Narudžba je već kreirana i nalazi se u bazi podataka sustava |
| **Posljedice:** | Podaci na narudžbi su izmijenjeni |
| **Počinje kada?** | Prilikom pritiska na gumb „Uredi“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC8 – Ocjena završene narudžbe

U tablici je opisan slučaj uporabe UC8 – "Ocjena završene narudžbe". Ovaj slučaj omogućava registriranom korisniku ocjenjivanje završene narudžbe ocjenom od 1 do 5. Ocjenjivanje počinje odabirom ocjene, a završava nakon njenog unosa.

Tablica 3.11. Slučaj uporabe UC8 – Ocjena završene narudžbe

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Ocjena završene narudžbe |
| **Namjena:** | Ocjenjivanje završene narudžbe |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Svaku završenu narudžbu korisnik može ocijeniti od 1 do 5 |
| **Slijed aktivnosti:** | Korisnik može ocijeniti narudžbu ocjenom od 1 do 5 |
| **Sadržaj podataka:** | Ocjena |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Narudžba je završena |
| **Posljedice:** | Narudžba sadrži podatak o ocjeni |
| **Počinje kada?** | Prilikom odabira ocjene |
| **Završava kada?** | Nakon odabira ocjene |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | SHOULD |

### Slučaj uporabe UC9 – Uređivanje osobnih podataka

U tablici je opisan slučaj uporabe UC9 – "Uređivanje osobnih podataka". Ovaj slučaj omogućava registriranom korisniku izmjenu osobnih podataka unesenih prilikom registracije. Izmjene se vrše pritiskom na gumb "Profil", a zatim "Spremi".

Tablica 3.12.Slučaj uporabe UC9 – Uređivanje osobnih podataka

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje osobnih podataka |
| **Namjena:** | Izmjena osobnih podataka |
| **Uloge (*role*):** | Registrirani korisnik |
| **Opis:** | Registrirani korisnik može mijenjati svoje podatke u sustavu koje je unio prilikom registracije u sustav |
| **Slijed aktivnosti:** | Pritiskom na gumb „Profil“ korisnik se preusmjerava na stranicu gdje može uređivati osobne podatke koje je unio prilikom registracije. Pritiskom na gumb „Spremi“ korisnik se preusmjerava na početnu stranicu svog profila. |
| **Sadržaj podataka:** | Isti podaci koji se upisuju prilikom registracije: email, lozinka, ime, prezime, telefon |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Korisnik je registriran i prijavljen u sustav |
| **Posljedice:** | Podaci o korisniku su izmijenjeni |
| **Počinje kada?** | Prilikom pritiska na gumb „Profil“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

Nakon izrade tablice za slučaj uporabe UC9 – "Uređivanje osobnih podataka", izrađen je dijagram aktivnosti. Dijagram prikazuje korake koje registrirani korisnik prolazi prilikom izmjene osobnih podataka. Proces počinje pritiskom na gumb "Uredi" na profilu korisnika, zatim se korisnik preusmjerava na stranicu za izmjenu podataka. Nakon unosa izmjena, pritiskom na gumb "Spremi" podaci se provjeravaju i ako su ispravni, spremaju se u bazu podataka. Ako podaci nisu ispravni, korisnik se vraća na stranicu za unos podataka. Dijagram jasno prikazuje tijek aktivnosti, olakšavajući razumijevanje i implementaciju procesa.

Dijagram 3.7. Dijagram aktivnosti za slučaj uporabe UC9

A diagram with black circles and dots

Description automatically generated

### Slučaj uporabe UC10 – Pregled narudžbi za Managera

Na temelju opisa slučaja uporabe UC10 – "Pregled narudžbi za Managera" izrađena je tablica koja detaljno prikazuje namjenu, uloge, slijed aktivnosti, sadržaj podataka, preduvjete i posljedice ovog slučaja uporabe.

Tablica 3.13. Slučaj uporabe UC10 – Pregled narudžbi za Managera

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Pregled narudžbi za managera |
| **Namjena:** | Informacije o svim narudžbama u bazi podataka |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager ima mogućnost pregleda svih narudžbi iz baze podataka bez obzira na status i datum narudžbe |
| **Slijed aktivnosti:** | Prijavom u sustav Manager dolazi na stranicu gdje vidi sve narudžbe u sustavu. |
| **Sadržaj podataka:** | Datum i vrijeme, broj gostiju, broj željenih jelovnika, lokacija, datum i vrijeme lokacije |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Zapis o narudžbi postoji u bazi podataka sustava |
| **Posljedice:** | Narudžbe prikazane na predviđenoj stranici |
| **Počinje kada?** | Prijavom u sustav |
| **Završava kada?** | Napuštanjem stranice za pregled narudžbi |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

Nakon izrade tablice, izrađen je dijagram aktivnosti koji vizualno prikazuje korake koje Manager prolazi prilikom pregleda svih narudžbi u sustavu. Dijagram pomaže u boljem razumijevanju i implementaciji procesa pregleda narudžbi.

Dijagram 3.8. Dijagram aktivnosti za upravljanje narudžbama od strane Managera obuhvaća sljedeće slučajeve uporabe UC10,UC11,UC12 i UC13

A diagram with black and white text

Description automatically generated

### Slučaj uporabe UC11 – Odluka o potvrdi narudžbe

Na temelju opisa slučaja uporabe UC11 – "Odluka o potvrdi narudžbe" izrađena je tablica koja prikazuje namjenu, uloge, slijed aktivnosti, sadržaj podataka, preduvjete i posljedice ovog slučaja uporabe. Tablica omogućava Manageru da odobri ili odbije narudžbu, čime se mijenja status narudžbe.

Tablica 3.14. Slučaj uporabe UC11 – Odluka o potvrdi narudžbe

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Odluka o potvrdi narudžbe |
| **Namjena:** | Promjena statusa narudžbe |
| **Uloge (*role*):** |  |
| **Opis:** | Manager može odobriti ili odbiti narudžbu. Kada odobri narudžbu, narudžba postaje aktivna. |
| **Slijed aktivnosti:** | Pritiskom na gumb „Obradi“ pored konkretne narudžbe pojavljuje se prozor u kojem Manager može mijenjati status narudžbe. |
| **Sadržaj podataka:** | Svi podaci o narudžbi |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Narudžba nije završena, Manager je prijavljen u sustav |
| **Posljedice:** | Promjena statusa narudžbe |
| **Počinje kada?** | Klikom na gumb „Obradi“ |
| **Završava kada?** | Klikom na gumb „Spremi“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** | UC12 |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC12 – Uvid u dostupne djelatnike

Na temelju opisa slučaja uporabe UC12 – "Uvid u dostupne djelatnike" izrađena je tablica koja omogućuje Manageru pregled dostupnih djelatnika za predviđeni termin.

Tablica 3.15. Slučaj uporabe UC12 – Uvid u dostupne djelatnike

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uvid u dostupne djelatnike |
| **Namjena:** | Pregled djelatnika koji nisu zauzeti za predviđeni termin izvršenja narudžbe tj. cateringa |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager tijekom odluke o potvrdi narudžbe tj. promjeni statusa narudžbe, mora imati uvid u raspoložive djelatnike za odabrani termin i mjesto narudžbe |
| **Slijed aktivnosti:** | Prilikom obrade narudžbe Manager pritiskom na padajući izbornik za svaku pojedinu grupu djelatnika (konobar, kuhar itd.) vidi tko je slobodan u vrijeme konkretne narudžbe. |
| **Sadržaj podataka:** | Ime i prezime djelatnika |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Dostupni podaci u bazi o zauzetosti djelatnika |
| **Posljedice:** | Promjena statusa zauzeća djelatnika |
| **Počinje kada?** | Tijekom odluke o promjeni statusa narudžbe |
| **Završava kada?** | Netom nakon promjene statusa narudžbe |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** | UC11 |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC13 – Generiranje predračuna

Na temelju opisa slučaja uporabe UC13 – "Generiranje predračuna" izrađena je tablica koja omogućuje Manageru generiranje predračuna za narudžbu u PDF formatu.

Tablica 3.16. Slučaj uporabe UC13 – Generiranje predračuna

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Generiranje predračuna |
| **Namjena:** | Izrada predračuna za catering |
| **Uloge (*role*):** | Manager |
| **Opis:** | Generiranje predračuna za narudžbe te ispisivanje u PDF format |
| **Slijed aktivnosti:** | Pritiskom na gumb „Predračun“ na ekranu se pojavljuje predračun za narudžbu u PDF formatu kojeg Manager, ako želi, može ispisati. |
| **Sadržaj podataka:** | Svi zakonom propisani podaci za predračun uključujući i neke podatke iz narudžbe na koju se predračun odnosi. |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Postojanje zapisa o narudžbi u bazi podataka, Manager je prijavljen |
| **Posljedice:** | Predračun je generiran u PDF |
| **Počinje kada?** | Klikom na gumb „Predračun“ |
| **Završava kada?** | Pojavom računa u PDF formatu na zaslonu uređaja |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | SHOULD |

### Slučaj uporabe UC14 – Uređivanje podataka o korisnicima

Na temelju opisa slučaja uporabe UC14 – "Uređivanje podataka o korisnicima" izrađena je tablica, omogućujući Manageru izmjenu podataka korisnika u sustavu.

Tablica 3.17. Slučaj uporabe UC14 – Uređivanje podataka o korisnicima

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje podataka o korisnicima |
| **Namjena:** | Izmjena podataka o korisnicima |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager može mijenjati podatke u sustavu koje je unio korisnik prilikom registracije u sustav |
| **Slijed aktivnosti:** | Pritiskom na poveznicu „Korisnici“ Manager se preusmjerava na stranicu gdje može upravljati (uređivati podatke, brisati) korisnicima. |
| **Sadržaj podataka:** | Isti podaci koji se upisuju prilikom registracije: email, lozinka, ime, prezime, telefon |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Manager je prijavljen, korisnik je registriran u sustavu |
| **Posljedice:** | Podaci o korisniku su izmijenjeni |
| **Počinje kada?** | Prilikom pritiska na gumb „Uredi“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

Nakon izrade tablice, izrađen je dijagram aktivnosti koji prikazuje korake izmjene podataka o korisnicima, zaposlenicima, lokacijama i jelovnicima. Dijagram jasno prikazuje tijek aktivnosti, olakšavajući razumijevanje ovih procesa.

Dijagram 3.9. Dijagram aktivnosti na slici obuhvaća sljedeće slučajeve uporabe UC14,UC15,UC16 i UC17.

A diagram of a work flow

Description automatically generated

### Slučaj uporabe PUC15 – Uređivanje podataka o zaposlenicima

Na temelju opisa slučaja uporabe UC15 – "Uređivanje podataka o zaposlenicima" izrađena je tablica. Tablica omogućuje Manageru izmjenu podataka o zaposlenicima u sustavu.

Tablica 3.18. Slučaj uporabe PUC15 – Uređivanje podataka o zaposlenicima

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje podataka o zaposlenicima |
| **Namjena:** | Izmjena podataka o zaposlenicima |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager može mijenjati podatke o zaposlenicima/djelatnicima u baza podataka sustava |
| **Slijed aktivnosti:** | Pritiskom na poveznicu „Zaposlenici“ Manager se preusmjerava na stranicu gdje može upravljati (uređivati podatke, brisati) zaposlenicima. |
| **Sadržaj podataka:** | Podaci o zaposlenicima: ime, prezime |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Manager je prijavljen, zaposlenik postoji u bazi podataka |
| **Posljedice:** | Podaci o zaposleniku su izmijenjeni |
| **Počinje kada?** | Prilikom pritiska na gumb „Uredi“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ ili „Obriši“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC16 – Uređivanje podataka o dostupnim lokacijama

Na temelju opisa slučaja uporabe UC16 – "Uređivanje podataka o dostupnim lokacijama" izrađena je tablica. Tablica omogućuje Manageru izmjenu podataka o lokacijama za catering u sustavu, uključujući naziv, adresu, kapacitet gostiju i potreban broj djelatnika.

Tablica 3.19. Slučaj uporabe UC16 – Uređivanje podataka o dostupnim lokacijama

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje podataka o dostupnim lokacijama |
| **Namjena:** | Izmjena podataka o dostupnim lokacijama |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager može mijenjati podatke o dostupnim lokacijama za catering koje se nalaze u bazi podataka sustava |
| **Slijed aktivnosti:** | Pritiskom na poveznicu „Lokacije“ Manager se preusmjerava na stranicu gdje može upravljati (uređivati podatke, brisati) lokacijama. |
| **Sadržaj podataka:** | Naziv, adresa, kapacitet gostiju, potreban broj djelatnika |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Manager je prijavljen, lokacija postoji u bazi podataka |
| **Posljedice:** |  |
| **Počinje kada?** | Prilikom pritiska na gumb „Uredi“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ ili „Obriši“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

### Slučaj uporabe UC17 – Uređivanje podataka iz jelovnika

Na temelju opisa slučaja uporabe UC17 – "Uređivanje podataka iz jelovnika" kreirana je tablica. Tablica omogućuje Manageru izmjenu podataka iz jelovnika u sustavu, uključujući naziv, sastojke i cijenu.

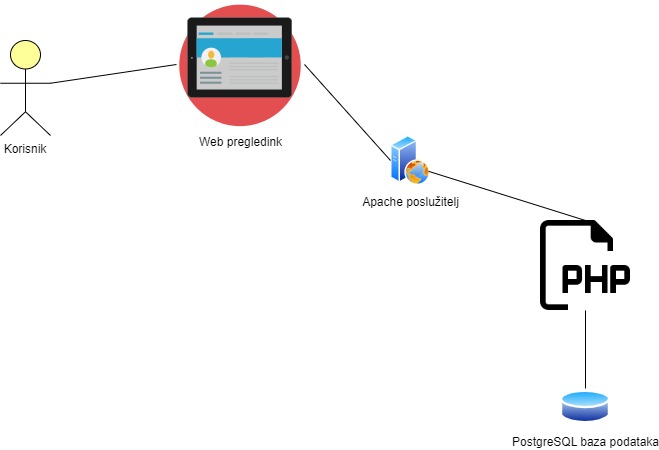
Tablica 3.20. Slučaj uporabe UC17 – Uređivanje podataka iz jelovnika

|  |  |
| --- | --- |
| **Naziv slučaja uporabe:** | Uređivanje podataka iz jelovnika |
| **Namjena:** | Izmjena podataka iz jelovnika |
| **Uloge (*role*):** | Manager |
| **Opis:** | Manager može mijenjati podatke iz jelovnika kao što su naziv, sastojci i cijena |
| **Slijed aktivnosti:** | Pritiskom na poveznicu „Meniji“ Manager se preusmjerava na stranicu gdje može upravljati (uređivati podatke, brisati) jelovnicima. |
| **Sadržaj podataka:** | Naziv, sastojci , cijena |
| **Transakcija:** |  |
| **Aplikacija:** |  |
| **Preduvjeti:** | Manager je prijavljen, jelovnik postoji u bazi podataka |
| **Posljedice:** | Podaci o jelovniku su izmijenjeni |
| **Počinje kada?** | Prilikom pritiska na gumb „Uredi“ |
| **Završava kada?** | Prilikom pritiska na gumb „Spremi“ |
| **Izuzeci:** |  |
| **Veza s drugim UC (ID):** |  |
| **Otvorena pitanja:** |  |
| **Prioritet:** | MUST |

## Opis sustava

Aplikacija se temelji na klijent-server arhitekturi. Korisnik aplikacije koristeći internet preglednik preko adrese „http://catering“ šalje zahtjev prema „Apache“ poslužitelju. Poslužitelj zatim prosljeđuje zahtjev „PHP“ web aplikaciji koja obrađuje zahtjev. Tijekom obrade zahtjeva, aplikacija komunicira s „PostgreSQL“ bazom podataka kako bi se odradile „CRUD“ (Create, Read, Update, Delete) operacije. Rezultat obrade zahtjeva tj. odgovor se zatim prosljeđuje natrag prema poslužitelju odnosno prema Internet pregledniku korisnika.

Dijagram 3.10. Dijagram prikaza arhitekture sustava



## Arhitektura sustava programske potpore

Dijagram 3.11. Arhitektura sustava programske potpore

A diagram of a software company

Description automatically generated

Aplikacija koristi višeslojnu arhitekturu, koja se sastoji od prezentacijskog, logičkog i podatkovnog sloja.

Prvi sloj je prezentacijski sloj (eng. “presentation layer”) koji je implementiran pomoću Model-View-Controller (MVC) arhitekturalnog obrasca. Zadaća ovog sloja je zaprimati korisničke zahtjeve (HTTP request), proslijediti ili dohvatiti podatke iz ostalih slojeva te po završetku potrebnih operacija vratiti odgovor korisniku.

Prezentacijski sloj obrađuje HTTP zahtjeve korisnika, provjerava i obrađuje dobivene podatke iz zahtjeva te prosljeđuje obrađene podatke u logički sloj. Nakon što zaprimi odgovor od logičkog sloja, obrađuje ga i vraća korisniku u obliku HTML stranice ili, u slučaju potrebe, preusmjerava korisnika na neku drugu stranicu unutar aplikacije [11].

Dijagram 3.12. Dijagram MVC arhitekture [12]

A screenshot of a computer

Description automatically generated

Prezentacijski sloj je implementiran korištenjem MVC arhitekture (eng. "Model-View-Controller"). Svakom dijelu MVC arhitekture dodijeljena je specifična odgovornost:

* Model: Predstavlja podatke i poslovnu logiku aplikacije. Model može uključivati bazu podataka, klase i metode koje se koriste za dohvaćanje i manipuliranje podacima. Model ne zna ništa o View-u ili Controller-u.
* View: Predstavlja korisničko sučelje i prikazuje podatke Model-a korisniku. View ne obrađuje nikakve događaje i nema poslovne logike. View samo prikazuje podatke i dozvoljava korisnicima da interagiraju s aplikacijom.
* Controller: Predstavlja upravljačku jedinicu koja prima zahtjeve korisnika i obrađuje ih. Controller obrađuje korisničke akcije, mijenja Model ako je to potrebno, te zatim ažurira View kako bi se prikazale promjene.

Ova arhitektura omogućava programerima da rade na različitim dijelovima aplikacije bez da utječu na druge dijelove. Osim toga, omogućava jednostavnije održavanje aplikacije i lakše testiranje. Ako je potrebno, Model se može mijenjati bez utjecaja na View ili Controller, što omogućava veću fleksibilnost i prilagodljivost aplikacije.

Logički sloj (eng. “business layer”) implementira poslovnu logiku aplikacije – procedure i pravila izvršavanja korisničkih zahtjeva, prikladno raspoređene u odgovarajuće razrede. Nakon obrade, logički sloj prema potrebi prosljeđuje podatke na spremanje u podatkovni sloj (ili ga obavještava da je neke podatke potrebno obrisati) te vraća odgovor prezentacijskom sloju.

Podatkovni sloj (eng. “data access layer”) služi za pristup i spremanje podataka proslijeđenih iz logičkog sloja, te je ujedno apstrakcija korištene relacijske baze podataka.

Razrede koji predstavljaju domenske entitete (primjerice, korisnik, narudžbenica i sl.) i koriste se u svim slojevima aplikacije svrstavamo u set domenskih objekata (eng. “domain objects”).

## Opis modela i baze podataka

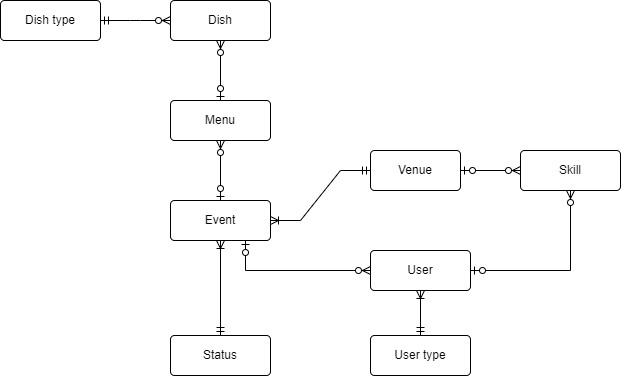
Baza podataka sadrži odgovarajuće tablice za sve entitete i prateće podatke potrebne za rad aplikacije. Osnovni poslovni entiteti su korisnik (tablica “users”), event (tablica “events”), lokacija (tablica “venues”) te meni (tablica “menus”).

Prateći entiteti/podaci su sadržani u ostalim tablicama:

* “status” - status pojedinog eventa (može biti “zaprimljen”, “prihvaćen”, “odbijen”, “završen”)
* “user\_types” - tipovi korisnika (“manager”, “zaposlenik” i “korisnik”)
* “skills” - kompetencije zaposlenika (“konobar”, “kuhar” i “vozač”)
* “employee\_skills” - kompetencija pojedinog zaposlenika
* “venue\_employees” - potrebe za zaposlenicima određenih kompetencija na pojedinoj lokaciji (primjerice, na lokaciji X potrebno je 10 konobara, 2 kuhara i 2 vozača za uspješno održavanje eventa)
* “event\_employees” - zaposlenici zaduženi za određeni event (odabire ih manager prilikom odobravanja eventa)
* “event\_menus” - broj određenih menija koji treba biti poslužen na eventu (primjerice, na eventu X treba poslužiti 80 toplo-hladnih menija i 5 vegetarijanskih menija)
* “dish” - jelo dostupno za posluživanje
* “dish\_types” - dostupni tipovi jela i pića (primjerice, hladno predjelo, glavno jelo, aperitiv i sl.)
* “menu\_dishes” - označava pridruživanje nekog jela određenom meniju (meni se sastoji od više jela, a jedno jelo može biti dio više menija)

Radi se o relacijskoj bazi implementiranoj u PostgreSQL-u, te se koriste prikladna ograničenja (eng. “constraint”) i strani ključevi (eng. “foreign key”). U nastavku su dijagrami konceptualnog i fizičkog modela baze podataka:

Dijagram 3.13. Konceptualni model baze podataka



Dijagram 3.14. Fizički model baze podatka

A screenshot of a computer

Description automatically generated

## Pregled razreda

Sljedeći dijagrami prikazuju raspored razreda po gore opisanim slojevima i modulima (imenskim prostorima, eng. “namespace”) unutar aplikacije:

Dijagram 3.15. Dijagram HTTP & Presentation

A screenshot of a computer

Description automatically generated

Dijagram 3.16. Business Logic

A screenshot of a computer

Description automatically generated

Dijagram 3.17. Data Access

A screenshot of a computer

Description automatically generated

Dijagram 3.18. Domain objects

A screenshot of a computer

Description automatically generated

### Kontroler

Kontroleri obrađuju pristigle HTTP zahtjeve korisnika, obrađene parametre prema potrebi prosljeđuju odgovarajućim razredima u logičkom sloju aplikacije te vraćaju odgovor korisniku u obliku HTML stranice ili objekta u JSON formatu

#### Modul „Auth“

* **LoginController** - sadrži metode za autentikaciju korisnika (prijava i odjava u aplikaciju). Kao parametere prima UserManager i SessionRepository objekte (koji će biti opisani u nastavku teksta)
* **RegisterController** – metode za registraciju novih korisnika

#### Modul „Manager“

* **DishController** – metode za upravljanje dostupnim jelima (pregled, dodavanje, uređivanje i brisanje)
* **EmployeeController** – metode za upravljanje zaposlenicima (pregled, dodavanje, uređivanje i brisanje)
* **EventController** – metode za upravljanje događajima od strane korisnika koji imaju tip “manager” (pregled svih evenata, prihvaćanje ili odbijanje zahtjeva za pojedinim eventima, odabir zaposlenika zaduženih za pojedini event u slučaju prihvaćanja, ispis predračuna/računa, brisanje zahtjeva koji jos nisu obrađeni)
* **ManagerHomeController** – prikaz početne stranice za managera (popis zahtjeva koji još nisu obrađeni)
* **MenuController** – metode za upravljanje menijima (dodavanje, uređivanje i brisanje menija, dodavanje ili uklanjanje jela s pojedinog menija)
* **UserController** – metode za upravljanje korisnicima (uređivanje i brisanje registriranih korisnika, pregled evenata za pojedinog korisnika)
* **VenueController** – metode za pregled i uređivanje podataka o dostupnim lokacijama

#### Modul „User“

**UserHomeController** – sve dostupne metode za vanjske korisnike (dodavanje zahtjeva za eventom, brisanje neobrađenih zahtjeva, ocjenjivanje eventa nakon završetka, ispis računa/predračuna, uređivanje osobnih podataka, pregled dostupnih lokacija i menija)

### Manager (modul „Manager“)

Razredi u modulu “Manager” implementiraju poslovnu logiku aplikacije te po potrebi komuniciraju s ostalim slojevima aplikacije.

* **EventManager** - sadrži metode za sve operacije nad eventima od strane korisnika ili managera - prima podatke iz pozivajućeg kontrolera/managera, izvršava potrebnu obradu i prema potrebi prosljeđuje podatke repozitorijima za spremanje u bazu podataka te na kraju vraća podatke pozivatelju. EventManager sadrži metode za:
* Procesiranje zahtjeva za eventom od strane korisnika
* Obradom zahtjeva od strane managera
* Dohvat dostupnih zaposlenika za određeni vremenski period
* Dohvat dostupnih lokacija za određeni vremenski period
* Procesiranje završenih evenata
* Generiranje podataka potrebnih za ispis predračuna/računa
* Ocjenjivanje završenih evenata
* Brisanje evenata (koji imaju status “zaprimljeno” ili “odbijeno”)
* **MenuManager** - sadrži logiku povezanu s dodavanjem, uređivanjem i brisanjem pojedinih jela i menija, implementiranu u metodama za:
* Dodavanje jela
* Uređivanje jela
* Brisanje jela
* Dodavanje jela meniju
* Uklanjanje jela s menija
* **UserManager** - sadrži metode za autentikaciju i upravljanje korisnicima:
* Autentikacija korisnika (provjera ispravnosti korisničkog imena i lozinke prilikom prijave u aplikaciju)
* Dodavanje novog korisnika (prilikom registracije korisnika)
* Uređivanje korisnika (prilikom uređivanja osobnih podataka od strane korisnika ili uređivanja zaposlenika od strane managera)
* Uklanjanje korisnika iz sustava
* **VenueManager** - sadrži metode za upravljanje lokacijama:
* Dodavanje nove lokacije
* Uređivanje podataka o pojedinoj lokaciji
* Brisanje lokacije
* Uređivanje potrebnog broja zaposlenika za lokaciju

### Repozitorij (modul „Repository“)

Repozitoriji obavljaju dohvat i spremanje podataka iz baze podataka, te ujedno služe i kao apstrakcija za korištenu bazu. Radi fleksibilnosti implementacije i jednostavnijeg testiranja aplikacije, svaki domenski entitet ima pripadajući repozitorij odgovarajućeg naziva (prati tip entiteta, primjerice razred “UserRepository” obavlja operacije s entitetima/objektima tipa “User”). Izuzetak od navedene konvencije je razred “SessionRepository” koji nema prateći domenski entitet, već se koristi za dohvat i spremanje podataka o trenutno prijavljenom korisniku (trenutno aktivna sesija korisnika). Repozitoriji su implementirani u slijedećim razredima (atributi i metode pojedinih repozitorija su navedeni u poglavljima 29 i 30):

* **DishRepository**
* **DishTypeRepository**
* **EmployeeSkillRepository**
* **EventEmployeeRepository**
* **EventMenuRepository**
* **EventRepository**
* **MenuDishRepository**
* **MenuRepository**
* **SessionRepository**
* **SkillRepository**
* **StatusRepository**
* **UserRepository**
* **UserTypeRepository**
* **VenueEmployeesRepository**
* **VenueRepository**

### Entitet (modul „Entity“)

Entiteti su osnovni razredi aplikacije, te opisuju korisnike, lokacije i sve ostale aktore/veze među aktorima sustava. Entiteti se ujedno koriste i prilikom interakcije s bazom podataka (repozitoriji prilikom dohvata podataka iz baze vraćaju kolekcije entiteta, te prilikom spremanja podataka također primaju podatke u obliku instance objekta entiteta). Implementirani su u slijedećim razredima:

* **User** – korisnik sustava. Sadrži metode za dohvat/postavljanje svih potrebnih osobnih podataka (ime, prezime, e-mail adresa i sl.) te tipa korisnika
* **UserType** – tip korisnika (“manager”, “zaposlenik” ili “korisnik”)
* **Event** – event, sa pripadajućim metodama za dohvat/postavljanje lokacije, broja gostiju, statusa etc.
* **EventEmployee** – predstavlja poveznicu između eventa i zaposlenika
* **EventMenu** – predstavlja poveznicu izmedu jednog eventa i jednog menija (uključujući broj menija koji treba biti posluzen)
* **Status** – status eventa (“zaprimljen”, “odobren”, “odbijen”, “završen”)
* **Venue** – lokacija, sa pripadajućim metodama za dohvat/postavljanje naziva, adrese etc
* **VenueEmployees** – poveznica broja potrebnih kompetencija zaposlenika i određene lokacije
* **Menu** – meni
* **Dish** - jelo
* **DishType** – tip jela (“hladno predjelo”, “aperitiv” i sl.)
* **MenuDish** – poveznica između pojedinog menija i jela
* **Skill** – kompetencija zaposlenika
* **EmployeeSkill** – poveznica između korisnika koji ima tip “zaposlenik” i određene kompetencije

### Servisi (modul „Services“)

Servisi su namjenski razredi koji obavljaju neku jasno definiranu zadaću koja je periferna samoj poslovnoj logici aplikacije – primjerice, ispis PDF dokumenata, zapisivanje datoteka na poslužitelj i slično. Aplikacija ima definiran jedan razred tog tipa, PDFService– sastoji se od jedne metode koja prima tekstualni HTML dokument i iz njegovog sadržaja generira PDF datoteku. Koristi se prilikom ispisa računa i predračuna za korisnike i managere.

### Enumeracija (modul “Enum“)

U modulu “Enum” je definirano nekoliko enumeracija čije vrijednosti odgovaraju identifikatorima (primarnim ključevima) u bazi podataka za pojedine entitete. Koriste se prilikom prikaza povezanih entiteta na korisničkom sučelju, spremanja entiteta i ostalih zadaća gdje je to prikladno – namjena im je izbjeći suvišne upite na bazu ako je već poznato koji entitet (njegov identifikator) će biti potreban za prikaz ili obavljanje neke druge zadaće. Enumeracije su, zbog načina rada PHP jezika, definirane kao apstraktni razredi te se za dohvat željenih vrijednosti koriste konstante definirane unutar razreda. U aplikaciji su definirane slijedeće enumeracije:

* **SkillEnum** (identifikatori vještina zaposlenika iz entiteta Skill)
* **StatusEnum** (identifikatori statusa eventa iz entiteta Status)
* **UserTypeEnum** (identifikatori tipa korisnika iz entiteta UserType)

### Rukovoditelji događajima (modul „Event“)

Rukovatelji događajima (eng. “event handler” ili “event listener”) su razredi čija je zadaća reagirati na određene događaje prilikom rada aplikacije. Navedene događaje izlaže korišteni programski okvir, te također omogućava registraciju rukovatelja korištenjem konfiguracije okvira. Aplikacija koristi slijedeće rukovatelje (prije izvršavanja kontrolera):

* **SessionListener** – prilikom svakog HTTP zahtjeva razred provjerava je li korisnik od kojeg zahtjev dolazi trenutno prijavljen u aplikaciju i koju metodu korisničkog sučelja pokušava pozvati. Ako korisnik nije prijavljen i metoda zahtjeva prijavu, razred će preusmjeriti korisnika na stranicu za prijavu i zaustaviti daljnje izvršavanje dotičnog zahtjeva.
* **PermissionListener** – prilikom svakog HTTP zahtjeva razred provjerava smije li korisnik pristupiti određenoj metodi (u sebi sadrži polje s definicijama koji tip korisnika smije pristupiti kojim metodama sučelja). Ako korisnik ne smije pristupiti određenoj metodi, razred ga vraća na početnu metodu za njegov tip korisnika
* **CSRFListener** – prilikom svakog POST HTTP zahtjeva razred provjerava prisutnost i ispravnost CSRF tokena u zahtjevu. CSRF (eng. “Cross-Site Request Forgery”) je oblik sigurnosne ranjivosti u kojoj napadač korištenjem nekoliko metoda pokreće POST HTTP zahtjev umjesto legitimnog korisnika, te se korištenjem CSRF tokena može spriječiti u većini slučajeva. Razred u slučaju neispravnog/nepostojećeg korisnika “baca” iznimku koja zaustavlja daljnji tijek zahtjeva.

### Twig ekstenzije

Za prikazivanje sučelja se koriste predlošci u programskom okviru Twig, koji daje mogućnost definiranja ekstenzija za predloške - metoda koje implementiraju željene funkcionalnosti prikaza koje sam okvir nema. Aplikacija ima definirane dvije Twig ekstenzije – **CSRFExtension** (registrira metodu proširenja u okvir i definira njene mogućnosti) i **CSRFTokenGenerator** (generira izlazni tekst u HTML formatu prilikom poziva metode definirane u CSRFExtension). Namjena obje ekstenzije je dohvat i prikazivanje CSRF tokena na korisničkom sučelju.

## UML dijagrami razreda

Obzirom da bi dijagram s svim razredima u aplikaciji bio velik i nepregledan, odlučio sam prikazati dijagrame razreda u modulu “Manager” s pripadajućim asocijacijama, te zaseban dijagram poslovnih entiteta - većina razreda u modulima “Entity” i “Manager” koristi implementacije nekoliko poslovnih entiteta (pa bi njihovo uključivanje značajno narušilo preglednost dijagrama).

Dijagram 3.19. Dijagram razreda za EventManager

A group of white paper with text on it

Description automatically generated

Dijagram 3.20. Dijagram razreda za MenuManager

A screenshot of a computer screen

Description automatically generated

Dijagram 3.21. Dijagram razreda za UserManager

A group of white papers on a black background

Description automatically generated

Dijagram 3.22. Dijagram razreda za VenueManager

A group of white papers on a black background

Description automatically generated

Dijagram 3.23. Dijagram razreda za poslovne entitete

A group of white papers on a black background

Description automatically generated

## Pregled atributa

### Kontroler (modul „Controller“)

Gotovo svi atributi u razredima modula Controller su instance razreda iz modula “Manager” i “Repository” te su dobiveni u konstruktoru objekta iz DI (eng. “dependency injection”) kontejnera kojeg koristi Symfony programski okvir. Osim toga, gotovo svi kontroleri definiraju atribut “viewPath” - string koji definira putanju poddirektorija u kojem se nalaze predlošci koji će se prikazivati korisnicima prilikom poziva neke od metode definirane u tom kontroleru. Za atribute koji ne pripadaju u jednu od dvije gore opisane grupe će biti dodan opis/značenje. Atributi su navedeni u obliku **NAZIV:TIP** (primjerice $viewPath: string).

#### Modul „Auth“

**LoginController**:

* $session: SessionRepository – SessionRepository
* $userManager: UserManager – UserManager
* $viewPath: string

**RegisterController**

* $userManager: UserManager
* $viewPath: string

#### Modul „Manager“

**DishController**

* $dishes: DishRepository
* $dishTypes: DishTypeRepository
* $menuManager: MenuManager
* $viewPath: string

**EmployeeController**

* $employeeSkills: EmployeeSkillRepository
* $skills: SkillRepository
* $userManager: UserManager
* $users: UserRepository
* $viewPath: string

**EventController**

* $eventManager: EventManager
* $events: EventRepository
* $statuses: StatusRepository
* $viewPath: string

**ManagerHomeController**

* $eventManager: EventManager
* $events: EventRepository
* $viewPath: string

**MenuController**

* $dishes: DishRepository
* $dishTypes: DishTypeRepository
* $menuManager: MenuManager
* $menus: MenuRepository
* $viewPath: string

**UserController**

* $events: EventRepository
* $userManager: UserManager
* $users: UserRepository
* $viewPath: string

**VenueController**

* $venueManager: VenueManager
* $venues: VenueRepository
* $viewPath: string

#### Modul „User“

**UserHomeController**

* $dishTypes: DishTypeRepository
* $eventManager: EventManager
* $events: EventRepository
* $exportPath: string
* $menus: MenuRepository
* $session: SessionRepository
* $userManager: UserManager
* $users: UserRepository
* $venues: VenueRepository
* $viewPath: string

### Manager (modul „Manager“)

Atributi u razredima modula “Manager” su velikom većinom instance repozitorija ili drugih razreda iz istog modula – dobivaju se iz DI kontejnera putem konstruktora, te će u nastavku biti naveden njihov naziv i tip (u istom obliku kao i u prethodnim odlomcima):

**EventManager**

* $employeeSkills: EmployeeSkillRepository
* $eventEmployees: EventEmployeeRepository
* $eventMenus: EventMenuRepository
* $events: EventRepository
* $statuses: StatusRepository
* $users: UserRepository
* $venues: VenueRepository

**MenuManager**

* $dishes: DishRepository
* $menuDishes: MenuDishRepository
* $menus: MenuRepository

**UserManager**

* $employeeSkills: EmployeeSkillRepository
* $skills: SkillRepository
* $users: UserRepository
* $userTypes: UserTypeRepository

**VenueManager**

* $venueEmployees: VenueEmployeesRepository
* $venues: VenueRepository

### Repozitorij (modul “Repository”)

Razredi iz modula “Repository” nemaju vlastitih definiranih atributa - nasljeđuju razrede ServiceEntityRepository i EntityRepository kojeg definira Doctrine programski okvir (aplikacija ga koristi kao ORM modul, eng. “Object Relation Mapper”). Razred EntityRepository definira slijedeće atribute:

* **$\_entityName**: string – naziv poslovnog entiteta uz kojeg se repozitorij “veže”
* **$\_em**: EntityManagerInterface – objekt koji implementira EntityManagerInterface, u kojem su definirane generičke metode za dohvat i spremanje podataka (primjerice, findById(int id), all() i slične)
* **$\_class**: ClassMetadata – ClassMetadata objekt koji sadrži anotacije i atribute definirane unutar poslovnih entiteta (anotacije i atributi definiraju na koji stupac u bazi podataka se veže pojedini atribut entiteta, preko kojih stupaca dohvaća povezane entitete i sl.)
* **$inflector**: Inflector - razred s metodama za manipulaciju tekstualnim (“string”) varijablama. Koristi se potrebne manipulacije anotacijama i atributima prilikom instanciranja repozitorija

U nastavku su, radi konzistencije dokumenta, nabrojani razredi u modulu “Repository”:

* DishRepository
* DishTypeRepository
* EmployeeSkillRepository
* EventEmployeeRepository
* EventMenuRepository
* EventRepository
* MenuDishRepository
* MenuRepository
* SessionRepository
* SkillRepository
* StatusRepository
* **UserRepository**
* **UserTypeRepository**
* **VenueEmployeesRepository**
* **VenueRepository**

### Entitet (modul „Entity“)

* User
* UserType
* Event
* EventEmployee
* EventMenu
* Status
* Venue
* VenueEmployees
* Menu
* Dish
* DishType
* MenuDish
* Skill
* EmployeeSkill

### Servisi (modul „Services“)

* PDFService

### Enumeracije (modul „Enum“)

* SkillEnum
* StatusEnum
* UserTypeEnum

### Rukovoditelj događajima (modul „Event“)

* SessionListener
* PermissionListener
* CSRFListener

### Twig ekstenzije

* CSRFExtension
* CSRFTokenGenerator

## Pregled metoda

### Kontroler

#### Modul „Auth“

**LoginController**

* \_\_construct() : mixed
* baseRoute(): Response
* login(): Response
* loginAction(): Response
* logout(): RedirectResponse

**RegisterController**

* \_\_construct(): mixed
* register(): Response
* registerAction(): RedirectResponse|Response
* validateNewUser(): bool

#### Modul „Manager“

**DishController**

* \_\_construct(): mixed
* add(): Response
* addAction(): RedirectResponse
* delete(): RedirectResponse
* edit(): Response
* editAction(): RedirectResponse
* index(): Response

**EmployeeController**

* \_\_construct(): mixed
* add(): Response
* addAction(): RedirectResponse
* delete(): RedirectResponse
* edit(): Response
* editAction(): RedirectResponse
* index(): Response

**EventController**

* \_\_construct(): mixed
* delete(): mixed
* generateInvoice(): mixed
* index(): mixed
* processEvent(): Response
* processEventAction(): RedirectResponse

**ManagerHomeController**

* \_\_construct(): mixed
* home(): Response

**MenuController**

* \_\_construct(): mixed
* addDishToMenu(): Response
* addDishToMenuAction(): RedirectResponse
* addMenu(): mixed
* addMenuAction(): mixed
* deleteMenu(): mixed
* editMenu(): mixed
* editMenuAction(): mixed
* getAvailableDishes(): mixed
* index(): Response
* removeDishFromMenu(): RedirectResponse

**UserController**

* \_\_construct(): mixed
* events(): Response
* index(): Response
* remove(): RedirectResponse
* update(): Response
* updateAction(): RedirectResponse

**VenueController**

* \_\_construct(): mixed
* addVenue(): Response
* addVenueAction(): RedirectResponse
* deleteVenue(): mixed
* editVenue(): Response
* editVenueAction(): RedirectResponse
* index(): Response

#### Modul „User“

**UserHomeController**

* \_\_construct(): mixed
* addEvent(): Response
* addEventAction(): RedirectResponse
* deleteEvent(): RedirectResponse
* downloadEventReceipt(): Response
* editEvent(): mixed
* editEventAction(): mixed
* getAvailableVenues(): mixed
* home(): Response
* menus(): Response
* profile(): Response
* profileUpdateAction(): RedirectResponse
* rateEvent(): Response
* rateEventAction(): RedirectResponse
* venues(): Response
* createDates(): array

### Manager (modul „Manager“)

**EventManager**

* \_\_construct(): mixed
* deleteEvent(): bool - Deletes the event and associated data
* generateReceiptData(): array<string|int, mixed>|null
* getAvailableEmployees(): array<string|int, EmployeeSkill>|null - Returns the available employees (not assigned to any event) for the requested time range
* getAvailableVenues(): array<string|int, mixed>|null - Returns the available venues (venues that currently don't have confirmed events) for the requested time range
* processCompletedEvents(): bool - Marks events whose end time has passed as completed
* processEvent(): Event|null - Process an event request by the manager
* rateEvent(): Event|null
* requestEvent(): Event|null - Process an incoming request by the user

**MenuManager**

* \_\_construct(): mixed
* addDish(): Dish|null
* addDishToMenu(): MenuDish|null
* editDish(): Dish|null
* removeDish(): bool
* removeDishFromMenu(): bool

**UserManager**

* \_\_construct(): mixed
* addNewUser(): User|null - Saves the user to the database
* authenticateUser(): User|null
* removeUser(): bool
* updateUser(): User|null
* setUserData(): User- Sets data shared by all user types from the provided array

**VenueManager**

* \_\_construct(): mixed
* addVenue(): Venue|null
* deleteVenue(): bool
* editVenue(): Venue|null
* addEmployees(): VenueEmployees
* setEmployeeCount(): void

### Repozitorij (modul „Repository“)

**DishRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, Dish>
* getByID(): Dish|null
* remove(): void
* save(): Dish

**DishTypeRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, DishType>

**EmployeeSkillRepository**

* \_\_construct(): mixed
* getForAllEmployees(): array<string|int, EmployeeSkill>
* getForEmployee(): EmployeeSkill
* save(): EmployeeSkill

**EventEmployeeRepository**

* \_\_construct(): mixed
* getByTimeInterval(): array<string|int, EventEmployee>
* getForEvent(): array<string|int, EventEmployee>
* save(): EventEmployee

**EventMenuRepository**

* \_\_construct(): mixed
* getForEvent(): array<string|int, mixed>
* save(): EventMenu

**EventRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, Event>
* getByID(): Event|null
* getByStatus(): array<string|int, Event>
* getByTimeInterval(): array<string|int, Event>
* getForUser(): array<string|int, Event>
* remove(): void
* save(): Event

**MenuDishRepository**

* \_\_construct(): mixed
* getByID(): MenuDish
* remove(): void
* save(): MenuDish

**MenuRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, Menu>

**SessionRepository**

* \_\_construct(): mixed
* clear(): void
* getEmail(): string
* getFirstName(): string
* getGender(): string
* getLastName(): string
* getOIB(): string
* getPhone(): string
* getUserID(): string
* getUserType(): int|string
* isAuthenticated(): bool
* isManager(): bool
* isUser(): bool
* session(): SessionInterface
* setAuthenticated(): void
* setEmail(): void
* setFirstName(): void
* setGender(): void
* setLastName(): void
* setOIB(): void
* setPhone(): void
* setUser(): void
* setUserID(): void
* setUserType(): void

**SkillRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, Skill>

**StatusRepository**

* \_\_construct(): mixed
* getAll(): array<string|int, Status>

**UserRepository**

* delete(): void
* getAllUsersByType(): array<string|int, User>
* getByEmail(): User|null
* getByID(): User|null
* save(): User
* update(): object|null
* setParameters(): mixed

**UserTypeRepository**

* all(): array<string|int, UserType>
* getByID(): UserType

**VenueEmployeesRepository**

* delete(): void
* getForVenue(): array<string|int, mixed>
* save(): VenueEmployees

**VenueRepository**

* all(): array<string|int, Venue>
* delete(): void
* getByID(): Venue|null
* save(): Venue
* update(): object|null
* setParameters(): mixed

### Entitet (modul „Entity“)

* User
* UserType
* Event
* EventEmployee
* EventMenu
* Status
* Venue
* VenueEmployees
* Menu
* Dish
* DishType
* MenuDish
* Skill
* EmployeeSkill

### Servisi (modul “Services”)

* PDFService

### Enumeracije (modul „Enum“)

* SkillEnum
* StatusEnum
* UserTypeEnum

### Rukovoditelj događajim (modul „Event“)

* SessionListener
* PermissionListener
* CSRFListener

### Twig ekstenzije

* CSRFExtension
* CSRFTokenGenerator

## Opis izvedbe GUI-a

U ovom poglavlju opisuje se koncept realizacije GUI-a. Aplikacija je podijeljena na dva dijela. Prvi dio simulira prikaz korisnika, a drugi dio simulira prikaz menadžera. Funkcionalnosti se razlikuju. GUI je izrađen tako da je jednostavan za korištenje i funkcionalan. Svaki pogled i njegova funkcionalnost će biti kratko opisani s malim opisom te će se prikazati kako pogled izgleda.

### GUI - Prijava

Ovo je prva stranica koja se prikazuje pri ulasku u aplikaciju. Za pristup aplikaciji potreban je korisnički račun, koji uključuje email adresu i lozinku. Ako korisnički račun ne postoji, registracija je potrebna.

Graphical user interface, application

Description automatically generated

Slika 3.1. Prijava u sustav

### GUI – Registracija

Prikazuje se sučelje za registraciju. Novi korisnici trebaju biti registrirani kako bi koristili aplikaciju. Za registraciju su potrebni sljedeći podaci: ime, prezime, email, lozinka, broj telefona, adresa i spol.

Graphical user interface, application

Description automatically generated

Slika 3.2. Registracija korisnika u sustav

### GUI Korisnik – Moji Eventi

Prikaz 'Moji eventi' je prvi prikaz korisničkog sučelja aplikacije. U ovom pogledu prikazuju se svi eventi koje je korisnik izradio. Svaki event uključuje datum početka i završetka, lokaciju, broj gostiju, broj menija, status narudžbe i ocjenu usluge.

Klikom na gumb 'Dodaj' otvara se novi prozor za kreiranje eventa. Eventi se mogu sortirati i pretraživati. Event može biti u statusu zaprimljen, odobren, odbijen ili završen. Klikom na gumb 'Predračun', korisniku se ispisuje predračun za traženi event. Event se također može obrisati.

A screenshot of a computer

Description automatically generated

Slika 3.3. Moji eventi

Klikom na 'Novi event' otvara se nova forma za unos podataka. U formi se traže datum, vrijeme početka i završetka, lokacija, broj gostiju i broj menija.

Event se može urediti, pri čemu se otvara forma s već postojećim podacima i promjene se mogu spremiti.

A screenshot of a phone

Description automatically generated

Slika 3.4. Dodavanje novog eventa.

### GUI Korisnik – Lokacije

Pogled 'Lokacije' prikazuje sve moguće lokacije za provedbu eventa u obliku kartica, uključujući ime lokacije, kratki opis i sliku lokacije.

Graphical user interface, website

Description automatically generated

Slika 3.5. Lokacije

U tablici je prikazan popis svih lokacija, uključujući njihovo ime, adresu, broj stolova i broj gostiju.

Graphical user interface, text, application

Description automatically generated

Slika 3.6. Popis svih lokacija.

### GUI Korisnik – Meniji

U pogledu 'Meniji' prikazana su tri menija, od kojih se svaki sastoji od više kategorija jela.

Graphical user interface, text

Description automatically generated with medium confidence

Slika 3.7. Meniji

### GUI Manager – Novi Eventi

Početni zaslon 'Novi eventi' je prvi pogled menadžera. Prikazuju se informacije o eventu te ga je moguće odobriti ili odbiti.

Graphical user interface, text, application, chat or text message

Description automatically generated

Slika 3.8. Novi eventi.

Prilikom pregleda eventa zaposlenicima se dodjeljuju uloge. Ako nema dovoljno zaposlenika, ispisuje se obavijest.

A screenshot of a computer

Description automatically generated

Slika 3.9. Obrada evenata

### GUI Manager – Eventi

Pogled 'Svi eventi' je sličan pogledu 'Novi eventi', ali u ovom pogledu prikazuju se svi eventi. Za svaki event prikazuje se ocjena.

Graphical user interface, text, application

Description automatically generated

Slika 3.10. Prikaz evenata za rolu Manager

### GUI Manager – Korisnici

Pogled 'Korisnici' prikazuje listu korisnika i njihove osobne podatke. Klikom na gumb 'Eventi' prikazuju se svi eventi odabranog korisnika. Podaci korisnika mogu se uređivati i brisati.

Graphical user interface, text, application

Description automatically generated

Slika 3.11. Popis korisnika

Prozor za uređivanje podataka korisnika.

A screenshot of a computer

Description automatically generated

Slika 3.12. Uređivanje korisnika

### GUI Manager – Zaposlenici

Pogled 'Zaposlenici' prikazuje sve zaposlenike u obliku liste. Svaki zaposlenik ima svoje osobne podatke i kompetenciju.

Graphical user interface, text, application

Description automatically generated

Slika 3.13. Popis zaposlenika

Dodavanjem ili uređivanjem korisnika otvara se novi prozor za unos podataka. Potrebno je upisati osobne podatke zaposlenika i odabrati njegovu kompetenciju.

A screenshot of a computer

Description automatically generated

Slika 3.14. Dodavanje novog zaposlenika

### GUI Manager – Lokacije

Pogled 'Lokacije' prikazuje listu postojećih lokacija. Svaka lokacija ima svoje osnovne podatke, uključujući broj stolova, broj gostiju i broj zaposlenika.

Graphical user interface, text, application

Description automatically generated

Slika 3.15. Upravljanje lokacijama

Lokacija se može dodati ili urediti. Klikom na gumb otvara se prozor za unos potrebnih podataka. Svaka lokacija ima određeni broj potrebnih zaposlenika.

A screenshot of a computer

Description automatically generated

Slika 3.16. Dodavanje nove lokacije

### GUI Manager – Meniji

Pogled 'Meniji' prikazuje koja jela su u kojem meniju ili grupi. Postoje tri menija i u ovom pogledu mogu se dodavati i brisati određena jela iz pojedine grupe menija.A screenshot of a computer

Description automatically generated

Slika 3.17. Upravljanje menijima

Dodavanjem jela upisuje se ime jela koje postoji, a spremanjem se jelo automatski uvrštava u pripadajuću kategoriju.

A screenshot of a computer

Description automatically generated

Slika 3.18. Dodavanje novog jela

### GUI Manager – Jela

Pogled 'Jelo' sastoji se od više kategorija jela. U svakoj kategoriji jelo se može dodati, urediti ili obrisati.

Graphical user interface, application

Description automatically generated

Slika 3.19. Upravljanje jelima

A screenshot of a computer

Description automatically generated

Slika 3.20. Uređivanje jelima

# Zaključak

U ovom radu istražene su i uspoređene glavne metode za rad s API-ima u JavaScriptu: Fetch s promises chainingom, Fetch s asinkronim funkcijama, XMLHttpRequest, Axios i jQuery AJAX. Cilj je bio analizirati njihove karakteristike, implementaciju i performanse u različitim scenarijima kako bi se pružile smjernice za odabir najprikladnije metode.

Rezultati testiranja na dvije baze podataka pokazali su da su Fetch API i Axios najučinkovitiji u smislu brzine dohvata i obrade podataka. Nasuprot tome, jQuery AJAX pokazao je značajno sporije performanse, osobito pri obradi podataka. XMLHttpRequest, iako starija tehnologija, i dalje je relevantan zbog svoje široke podrške u preglednicima.

Na temelju analize, Fetch API se ističe kao moderna i učinkovita metoda, posebno kada se koristi s asinkronim funkcijama. Axios nudi jednostavnost i dodatne funkcionalnosti koje ga čine vrlo korisnim.

Kao dio ovog rada, razvijena je web aplikacija za usluge cateringa koristeći jQuery za rad s API-ima. Odluka o korištenju jQuery-a bila je povezana s činjenicom da je za frontend dio aplikacije korišten Bootstrap, koji je zavisan o jQuery-u. Budući da ovaj projekt nije bio namijenjen za produkciju, brzina nije bila ključni faktor.

Aplikacija omogućuje korisnicima registraciju, kreiranje narudžbi, odabir lokacija, pregled i uređivanje narudžbi te ocjenjivanje završenih usluga. Također, menadžerima se omogućuje upravljanje narudžbama, pregled dostupnih djelatnika i generiranje predračuna.

Zaključno, Fetch API i Axios su najprikladnije metode za većinu modernih web aplikacija zbog svoje brzine i fleksibilnosti. Međutim, izbor metode može varirati ovisno o specifičnim potrebama projekta, kao što su podrška za starije preglednike ili specifične funkcionalnosti.

Za buduća istraživanja preporučuje se ispitivanje ovih metoda u različitim realnim scenarijima, kao i istraživanje novih tehnologija koje se pojavljuju u web razvoju. Kombinacija ovih metoda također može pružiti dodatna poboljšanja u performansama i funkcionalnostima web aplikacija.

# Literatura

[1] <https://developer.mozilla.org/en-US/docs/Web/API/fetch>, 31.10.2023

[2] Haverbeke, Marijin, Eloquent JavaScript: A Modern Introduction to Programming, No Starch Press, San Francisco, 2024, str. 472

[3] <https://www.geeksforgeeks.org/javascript-fetch-method/>, 24.10.2023

[4] Simpson, Kylea ; You Don't Know JS: Async & Performance, O'Reilly Media, San Francisco, 2015, str. 272

[5]<https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Statements/async_function>, 23.10.2023

[6] Johnson Alex, Mastering XMLHttpRequest: Asynchronous JavaScript and XML, TechPress, New York, 2021, str. 320

[7]<https://developer.mozilla.org/en>US/docs/Web/API/XMLHttpRequest/Using\_XMLHttpRequest, 25.10.2023

[8] https://axios-http.com/docs/intro, 23.10.2023

[9] Brown, Samantha, Mastering jQuery AJAX: Building Interactive Web Applications, WebDev Publishing, San Francisco, 2019, str. 340

[10] https://api.jquery.com/jQuery.ajax/, 24.10.2023

[11] Erich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, Boston, MA, 1994, str. 395

[12] <https://developer.mozilla.org/en-US/docs/Glossary/MVC>, 23.10.2023