# Razvoj aplikacije Aukcije

Autori:

*Andrija Nekić, Dario Pešić, Nikola Benić, Sanel Halačević, Davor Nikšić*

# Opis aplikacije

Web aplikacija Aukcije služi za oglašavanje predmeta na aukciji. Korisniku se nudi opcija za registraciju i prijavu.

Bilo koji registrirani korisnik može sudjelovati u aukciji: oglasiti neki predmet (uslugu) i licitirati za neki oglašeni predmet.

Na početnoj stranici korisnik može pregledavati sve trenutne aukcije koje izabire u slučaju želje za licitacijom te mu se nudi mogućnost sortiranja prikazanih predmeta prema određenim kategorijama - namještaj, automobili, nakit, mobiteli i ostalo.

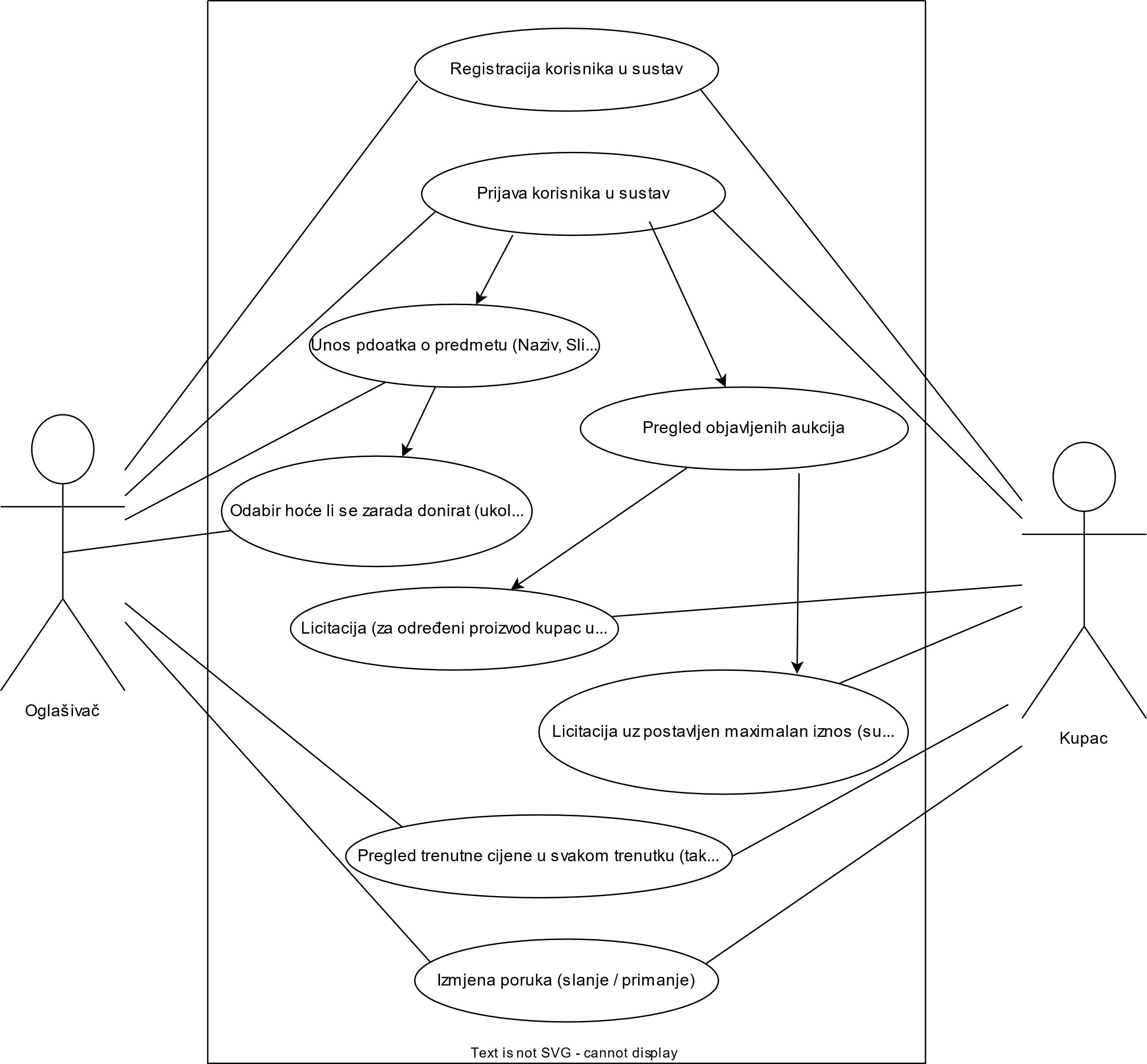
Kod oglašavanja predmeta registrirani korisnik unosi naziv predmeta koji postavlja na aukciju, početnu cijenu, odabire kategoriju u koju se predmet svrstava, dodaje sliku, aktivno razdoblje aukcije (datum i vrijeme početka i završetka aukcije) te opis predmeta. Može se postaviti da dobiveni iznos ide u humanitarne svrhe te ovdje korisnik izabire svrhu (npr. za osobe pogođene potresom).

Za licitiranje predmeta korisnik odabire neki od oglašenih predmeta za koji želi ponuditi određenu cijenu te ulazi u prikaz tog predmeta. Kod prikaza predmeta na licitaciji vidljivi su podaci: naziv predmeta, svrha donacije, slika, opis predmeta, početno i završno vrijeme aukcije te početna i trenutna cijena aukcije. Klikom na “Ponuda” korisniku se otvara mogućnost unošenja svote za koju želi povisiti trenutnu cijenu aukcije te se tako cijena ponudama stalno povećava do isteka aukcije.

Prijavljeni korisnik može otići na svoj profil te su tamo vidljivi njegovi podaci: ime i prezime, datum rođenja, e-mail te OIB. Također, može pregledavati svoje predmete na aukciji – koje je on postavio – te ih eventualno urediti ili obrisati.

# ANALIZA

## Funkcionalni zahtjevi



Slika 1 UML dijagram korištenja

|  |  |  |
| --- | --- | --- |
| **Funkcionalnost** | Opis | Realizacija |
| **Registracija korisnika** | Omogućuje novim korisnicima registraciju u aplikaciji za aukcije. | Realizirano |
| **Prijava u sustav** | Korisnici se mogu prijaviti putem svog korisničkog imena i lozinke koje su stvorene prilikom registracije. | Realizirano |
| **Mogućnost oglašavanja predmeta (usluge) na aukciji** | Omogućuje registriranom korisniku postavljanje oglasa za prodaju predmeta ili usluge putem aplikacije za aukcije. Korisnik može navesti naziv predmeta ili usluge, opis, sliku, aktivno razdoblje aukcije, kategoriju, svrhu i početnu cijenu. | Realizirano |
| **Mogućnost licitiranja za predmet (uslugu) oglašen na aukciji** | Omogućuje registriranim korisnicima da licitiraju za predmet ili uslugu oglašenu na aukciji. Korisnik može unijeti željeni iznos za koji želi podići aukciju. | Realizirano |
| **Postavljanje podataka o predmetu (naziv, opis, slika, aktivno razdoblje aukcije, početna cijena i kategorija) pri oglašavanju** | Omogućuje registriranim korisnicima postavljanje podataka o predmetu koji se prodaje, uključujući naziv, opis, sliku predmeta, vrijeme aktivnosti aukcije, kategoriju i početnu cijenu. | Realizirano |
| **Mogućnost odabira humanitarne svrhe za dobiveni iznos od prodaje** | Omogućuje korisnicima postavljanje dobivenog iznosa od prodaje predmeta u humanitarne svrhe s mogućnošću odabira svrhe od strane korisnika. | Realizirano |
| **Prikaz trenutne/postignute cijene za predmet (uslugu) na aukciji** | Omogućuje prikaz trenutne ili postignute cijene predmeta ili usluge na aukciji, tako da korisnici mogu vidjeti koliko je trenutno ponuđeno za predmet ili uslugu. | Realizirano |
| **Prikaz zadnje izmjene cijene za predmet na aukciji** | Omogućuje prikaz zadnje izmjene cijene za predmet ili uslugu na aukciji, tako da korisnici mogu vidjeti koliko se cijena promijenila od prethodne ponude. | Treba realizirati |
| **Prikaz pojedinosti aukcije** | Omogućuje prikaz pojedinosti o aukciji ulaskom na pojedinu aukciju. | Realizirano |
| **Sustav licitacije koji omogućuje registriranim korisnicima sudjelovanje u licitaciji za predmet na aukciji** | Omogućuje korisnicima sudjelovanje u licitaciji za predmet ili uslugu na aukciji, aplikacija osigurava sustav licitacije koji osigurava fer i transparentan proces licitacije. Korisnici mogu unijeti svoje ponude i vidjeti trenutni vodeći iznos. Aplikacija prati sve ponude i automatski povećava vodeću ponudu kada drugi korisnik ponudi višu cijenu. Licitacija se završava kada istekne aktivno razdoblje aukcije ili kada nema novih ponuda u određenom razdoblju. | Realizirano |
| **Mogućnost pregledavanja prethodnih aukcija** | Omogućuje korisnicima da pregledaju prethodne aukcije koje su završene. Korisnici mogu vidjeti koja su bila predmeti ili usluge na aukciji i koja je bila konačna cijena. | Realizirano |
| **Mogućnost postavljanja ograničenja na licitaciju** | Omogućuje registriranim korisnicima postavljanje ograničenja na licitaciju. Korisnici mogu odrediti najveći iznos koji su spremni platiti za predmet ili uslugu na aukciji. Aplikacija će automatski licitirati za korisnika sve dok iznos ne dosegne to ograničenje. | Treba realizirati |
| **Mogućnost slanja poruka drugim korisnicima** | Omogućuje registriranim korisnicima slanje poruka drugim korisnicima u vezi s aukcijama. Korisnici mogu postavljati pitanja o predmetima na aukciji ili razgovarati o detaljima licitacije. | Treba realizirati |
| **Sortiranje po kategorijama** | Omogućuje korisnicima da na početnoj stranici, klikom na gumb određene kategorije sortiraju predmete u toj kategoriji te se oni prikazuju na početnoj stranici. | Realizirano |
| **Pregled svih aukcija** | Na početnoj stranici nudi se pregled svih predmeta koji su na aukciji | Realizirano |
| **Prikaz vlastitiih aukcija na profilu** | Omogućuje korisniku odlazak na svoj profil gdje su vidljivi njegovi podaci te sve postavljene aukcije. | Realizirano |
| **Uređivanje postavljene aukcije na profilu** | Omogućuje korisnku uređivanje postavljene aukcije na svom profilu. | Treba realizirati |
| **Brisanje vlastite aukcije na profilu** | Omogućuje korisnku brisanje postavljene aukcije. | Treba realizirati |

## Korisničke priče i kriteriji prihvatljivosti

1. Kao kupac, želim imati mogućnost pretraživanja aukcija prema ključnoj riječi ili kategoriji, tako da mogu lako pronaći artikle za koje želim licitirati.
   * Sustav treba pružiti mogućnost pretraživanja aukcija na temelju ključnih riječi ili kategorija.
   * Pretraživanje treba biti učinkovito i brzo, pružajući relevantne rezultate koji odgovaraju upitu kupca.
2. Kao kupac, želim vidjeti trenutnu najvišu ponudu za stavku, tako da mogu donijeti informiranu odluku hoću li dati ponudu.
   * Sustav treba prikazivati trenutnu najvišu ponudu za svaki predmet na aukciji.
   * Informacije o najvišoj ponudi trebaju biti ažurirane u stvarnom vremenu kako bi kupac imao najnovije informacije pri donošenju odluke.
3. Kupac može pregledati detalje o pojedinom predmetu na aukciji - naziv, sliku, opis, početnu cijenu, trenutnu cijenu, početak aukcije i kad završava.
   * Sustav treba prikazivati sve relevantne informacije o predmetu na aukciji, uključujući naziv, sliku, opis, početnu cijenu, trenutnu cijenu, datum i vrijeme početka aukcije te datum i vrijeme završetka aukcije.
4. Kao prodavač, želim biti u mogućnosti kreirati nove aukcije brzo i jednostavno, tako da mogu svoje predmete staviti na prodaju što prije moguće.
   * Sustav treba pružiti korisničko sučelje koje omogućuje prodavačima jednostavno stvaranje novih aukcija.
   * Prodavači trebaju moći unijeti sve relevantne informacije o predmetu na aukciji, uključujući naziv, sliku, opis, početnu cijenu, datum i vrijeme početka aukcije te datum i vrijeme završetka aukcije.

A screenshot of a computer

Description automatically generated with low confidence

Slika 2 Dijagram aktivnosti postavljanja aukcije

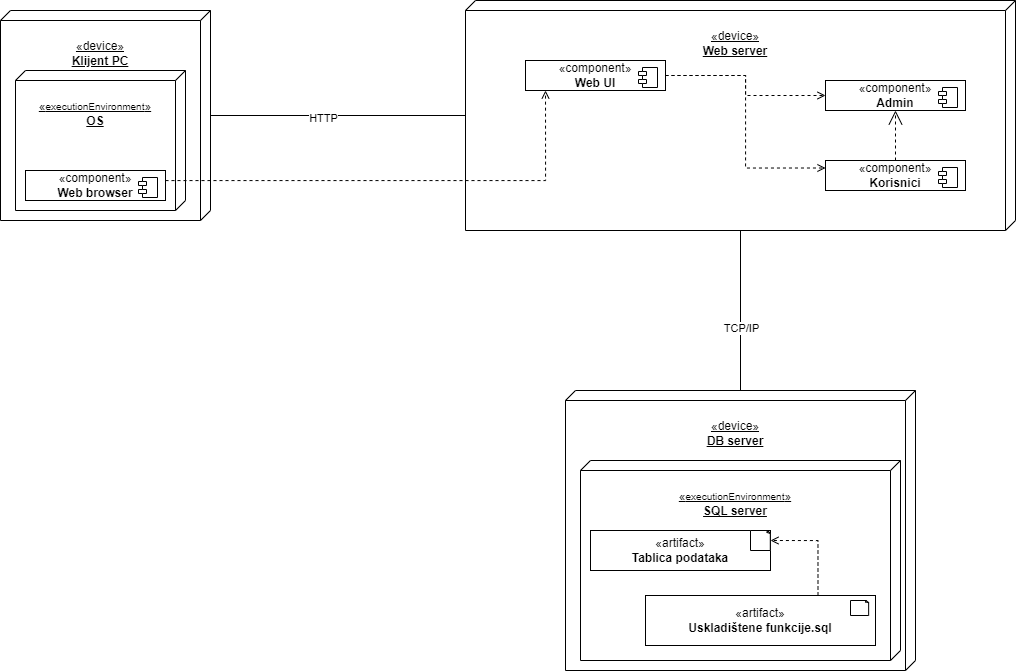
A screenshot of a black background

Description automatically generated with low confidence

Slika 3 Dijagram aktivnosti pregleda aukcija

# ARHITEKTURA SUSTAVA

Dijagram nudi implementacijski pogled na sistem, softverske i hardverske komponente. Najvažniji su čvorovi device i executionEnvironment. Klijent PC predstavlja hardver koji je potreban da se određene softverske komponente vrte ili hardver koji će koristiti klijent za pristup aplikaciji Aukcije. Može to biti i laptop, naravno. Web server je čvor koji služi za prikaz aplikacije, a DB server služi za bazu podataka. To su 3 hardverske komponente.



Slika 4: UML dijagram rasporeda

Čvorovi su povezani komunikacijskim linkovima kako bi se vidjelo tko s kime komunicira. Klijent komunicira s web serverom jer nema potrebe za komunikaciju s DB serverom, a web server komunicira s DB serverom. Komunikacija klijenta s web serverom se obavlja pomoću HTTP-a(može HTTPS za sigurnost), a web servera s DB serverom se obavlja pomoću TCP/IP protokola.

OS predstavlja operacijski sustav koji je neophodan za pristup aplikaciji. Može biti Linux, Windows itd. Rješenje koje se koristi kako bi se hostirala baza je SQL server pod DB server.

Komponenta koja se mora koristiti na klijentu da bi se pristupilo web aplikaciji je web browser i nalazi se kod Klijent PC. Dakle, pokreće se web browser s određenim OS-om i to se sve nalazi kod računala klijenta. Admin komponenta se tiče administracije aplikacije i postavki, a Korisnici komponenta ovisi o Admin komponenti(isprekidana linija). Označuje da promjene jedne komponente utječu na promjene druge komponente. Uskladištene funkcije ili stored procedures služi kao SQL datoteka. Koriste Tablica podataka za pristup podacima i manipulaciju podacima. Označuje se isprekidanom linijom. Uskladištene funkcije su ovisne o Tablice podataka.

Ono što treba biti dostupno na svakom web serveru je web UI tj. user interface ili korisničko sučelje gdje se nalaze html-ovi. Ovisan je o Admin i Korisnici. Različit je i za oboje.

Ovisnost PC i web servera je moguća samo kod komponenti koje služe za prikaz korisniku tj. klijentu, a to su web browser i web UI. Web browser je ovisan o web UI komponenti.

## Tehnologije za razvoj aplikacije

* Klijentski dio aplikacije - Quasar.js razvojni okvir
* Poslužiteljski dio aplikacije (engl. backend) - Node.js i Express razvojni okviri
* Upiti prema poslužiteljskoj strani aplikacije
* API koji će na temelju zahtjeva od strane klijenta slati podatke

## Klijentski razvojni okvir Quasar

Quasar je klijentski razvojni okvir otvorenog koda za izradu modernih web aplikacija i mobilnih aplikacija. On se temelji na Vue.js, popularnom JavaScript okviru za izgradnju korisničkog sučelja.

Quasar pruža širok spektar komponenti i funkcionalnosti koje olakšavaju razvoj klijentske strane aplikacije. To uključuje gotove komponente za različite elemente korisničkog sučelja poput gumba, obrazaca, kartica, navigacije i još mnogo toga. Ove komponente su prilagodljive i pružaju mogućnost stiliziranja i prilagođavanja prema potrebama aplikacije.

Jedna od ključnih značajki Quasara je mogućnost izgradnje iste aplikacije za web, mobilne uređaje i desktop. Koristeći Quasar, možete stvoriti aplikaciju koja se može izvoditi u pregledniku, kao instalirana mobilna aplikacija na iOS-u i Androidu te kao izvršna datoteka na stolnom računalu. To pruža fleksibilnost i jednostavnost pri razvoju i distribuciji aplikacija na različitim platformama.

Quasar je moćan klijentski razvojni okvir koji olakšava izgradnju modernih web aplikacija i mobilnih aplikacija. Pruža bogate komponente, podršku za više platformi i integraciju s raznim alatima, što ga čini popularnim izborom za razvoj interaktivnih i responzivnih aplikacija.

## Poslužiteljski razvojni okviri Node.js i Express

Node.js je JavaScript okruženje koje korisniku omogućuje pokretanje svog koda na poslužitelju (izvan web preglednika). Node ima vlastiti upravitelj paketa tj. NPM (Node Package Manager) koji omogućuje programeru odabir između tisuća besplatnih paketa (node modula) za preuzimanje. Neke od prednosti Node.js-a su sljedeće:

• JavaScript radno okruženje otvorenog koda - omogućava da se JavaScript kod  
pokreće na serveru.

• Skalabilnost - omogućava skaliranje aplikacije horizontalno preko nekoliko servera ili vertikalno tako da se optimiziraju performance jednog servera

• Brzina – Koristi Chrome-ov V8 JavaScript engine napisan u C++ programskom jeziku

Express.js koji radi unutar Node.js poslužitelja. Express.js sebe predstavlja kao "brz, samostalan, minimalistički web okvir za Node.js", i to je doista točno ono što jest. Express je okvir Node.js-a. Umjesto pisanja koda pomoću Node.js, može se jednostavnije pisati poslužiteljski kod pomoću Expressa. Express pomaže u dizajniranju API-ja (Application Programming Interface), koji su ključan koncept svake backend tehnologije. Iz React.js aplikacije izradom HTTP zahtjeva (GET, POST, PUT, DELETE) povezujemo se s Express.js funkcijama koje pokreću poslužiteljsku stranu aplikacije. Neke od prednosti korištenja Expressa su sljedeće: Učinkovit, brzo i skalabilan

Kombinacija Node.js i Express omogućuje programerima izgradnju brzih, skalabilnih i efikasnih poslužiteljskih aplikacija. Node.js pruža performanse i fleksibilnost, dok Express olakšava upravljanje rutama, obradu zahtjeva i izgradnju web aplikacija. Ova kombinacija je popularna u svijetu razvoja web aplikacija i često se koristi za izgradnju API-ja, mikroservisa i web aplikacija općenito.

## Web servisi i API

Web servisi su softverske komponente koje omogućuju razmjenu podataka i funkcionalnosti između različitih aplikacija putem interneta. Oni omogućuju komunikaciju i integraciju između različitih sistema, nezavisno o platformi ili tehnologiji koju koriste.

API (Application Programming Interface) je skup definiranih metoda i pravila koje omogućuju komunikaciju između različitih softverskih aplikacija. API-ji definraju načine na koje jedna aplikacija može da pristupi i koristi funkcionalnosti ili podatke druge aplikacije. Oni omogućuju programerima da integriraju funkcionalnosti drugih aplikacija u svoje projekte, bez potrebe za razumijevanjem detalja implementacije. API-ji se često koriste u kombinaciji sa web servisima. Web servisi obično pružaju funkcionalnosti i podatke putem API-ja, omogućavajući drugim aplikacijama da ih koriste putem standardnih HTTP (Hypertext Transfer Protocol) zahtjeva. API-ji definiraju dostupne rute, parametre i formate podataka koji se koriste prilikom komunikacije s web servisima. Kroz web servise i API-je, aplikacije mogu razmjenjivati podatke, izvršavati određene funkcionalnosti ili čak koristiti funkcionalnosti drugih aplikacija kako bi proširile svoje mogućnosti.

# DIZAJN SUSTAVA AUKCIJE

## Model podataka za Aukcije

Slika na kojoj se prikazuje tekst, dijagram, crta, skeč

Opis je automatski generiran

Slika 5. Model podataka – EVA[[1]](#footnote-1)

**Tablica entiteta i njihovih opisa**

|  |  |
| --- | --- |
| ENTITETI | OPIS |
| **KORISNIK** | Korisnik aplikacije, njegovi atributi su: id\_korisnika (PK), ime\_korisnika, prezime\_korisnika, e-mail, lozinka, adresa\_korisnika |
| **PREDMET** | Predmet na aukciji, njegovi atributi su: sifra\_predmeta(PK), naziv\_predmeta, slika, opis\_predmeta, vrijeme\_pocetka, vrijeme\_zavrsetka, pocetna\_cijena, svrha\_donacije, trenutna\_cijena |
| **KATEGORIJA** | Kategorija predmeta, njeni atributi su: sifra\_kategorije(PK), naziv\_kategorije |
| **PONUDA** | Ponuda za predmet, njeni atributi su: id\_ponude(PK), vrijednost\_ponude, vrijeme\_ponude |

Tablica 1. Tablica entiteta i njihovih opisa

**Tablica atributa entiteta i njihovih opisa**

|  |  |
| --- | --- |
| ATRIBUTI ENTITETA KORISNIK | OPIS |
| **id\_korisnika(PK)** | Broj koji identificira korisnika, te je primarni ključ |
| **ime\_korisnika** | Atribut koji predstavlja ime korisnika |
| **prezime\_korisnika** | Atribut koji predstavlja prezime korisnika |
| **e-mail** | Atribut koji predstavlja e-mail adresu korisnika |
| **lozinka** | Atribut koji predstavlja lozinku korisnika |
| **adresa\_korisnika** | Atribut koji predstavlja adresu korisnika |
| ATRIBUTI ENTITETA KATEGORIJA | OPIS |
| **sifra\_kategorije(PK)** | Broj koji identificira kategoriju, te je primarni ključ |
| **naziv\_kategorije** | Atribut koji predstavlja ime kategorije |
| ATRIBUTI ENTITETA PREDMET | OPIS |
| **sifra\_predmeta(PK)** | Broj koji identificira predmet, te je primarni ključ |
| **naziv\_predmeta** | Atribut koji predstavlja ime predmeta |
| **slika** | Atribut koji predstavlja sliku predmeta |
| **opis\_predmeta** | Atribut koji predstavlja opis predmeta |
| **vrijeme\_pocetka** | Atribut koji predstavlja vrijeme početka |
| **vrijeme\_zavrsetka** | Atribut koji predstavlja vrijeme završetka |
| **pocetna\_cijena** | Atribut koji predstavlja početnu cijenu |
| **svrha\_donacije** | Atribut koji predstavlja svrhu donacije |
| **trenutna\_cijena** | Atribut koji predstavlja trenutnu cijenu |
| **id\_korisnika(FK1)** | Vanjski ključ, povučen iz entiteta KORISNIK |
| **sifra\_kategorije(FK2)** | Vanjski ključ, povučen iz entiteta KATEGORIJA |
| ATRIBUTI ENTITETA PONUDA | OPIS |
| **id\_ponude(PK)** | Broj koji identificira ponudu, te je primarni ključ |
| **vrijednost\_ponude** | Atribut koji predstavlja vrijednost ponude |
| **vrijeme\_ponude** | Atribut koji predstavlja vrijeme ponude |
| **id\_korisnika(FK1)** | Vanjski ključ, povučen iz entiteta KORISNIK |
| **sifra\_predmeta(FK2)** | Vanjski ključ, povučen iz entiteta PREDMET |

Tablica 2. Tablica atributa i njihovih opisa

**Tablica veza i njihovih opisa**

|  |  |
| --- | --- |
| VEZE | OPIS |
| **KORISNIK - PREDMET** | Korisnik postavlja jedan ili više predmeta na aukciju, predmet je postavljen od strane jednog i samo jednog korisnika |
| **KATEGORIJA - PREDMET** | Kategorija ima jedan ili više predmeta, predmet ima jednu i samo jednu kategoriju |
| **KORISNIK - PONUDA** | Korisnik daje jednu ili više ponuda, ponuda je dana od strane jednog i samo jednog korisnika |
| **PREDMET - PONUDA** | Predmet sadrži jednu ili više ponuda, ponuda je dana za jedan i samo jedan predmet |

Tablica 3. Tablica veza i njihovih opisa

**Relacijski model podataka**

**Popis relacijskih shema**

KORISNIK(**id\_korisnika(PK),** ime,\_korisnika, prezime\_korisnika, e-mail, lozinka, adresa\_korisnika)

KATEGORIJA(**sifra\_kategorije(PK),** naziv\_kategorije)  
PREDMET(**sifra\_predmeta(PK),** naziv\_predmeta, slika, opis\_predmeta, vrijeme\_pocetka, vrijeme\_zavrsetka, svrha\_donacjie, trenutna\_cijena, id\_korisnika(FK1), sifra\_kategorije(FK2))

PONUDA( **id\_ponude(PK),** vrijednost\_ponude, vrijeme\_ponude, id\_korisnika(FK1), sifra\_predmeta(FK2))

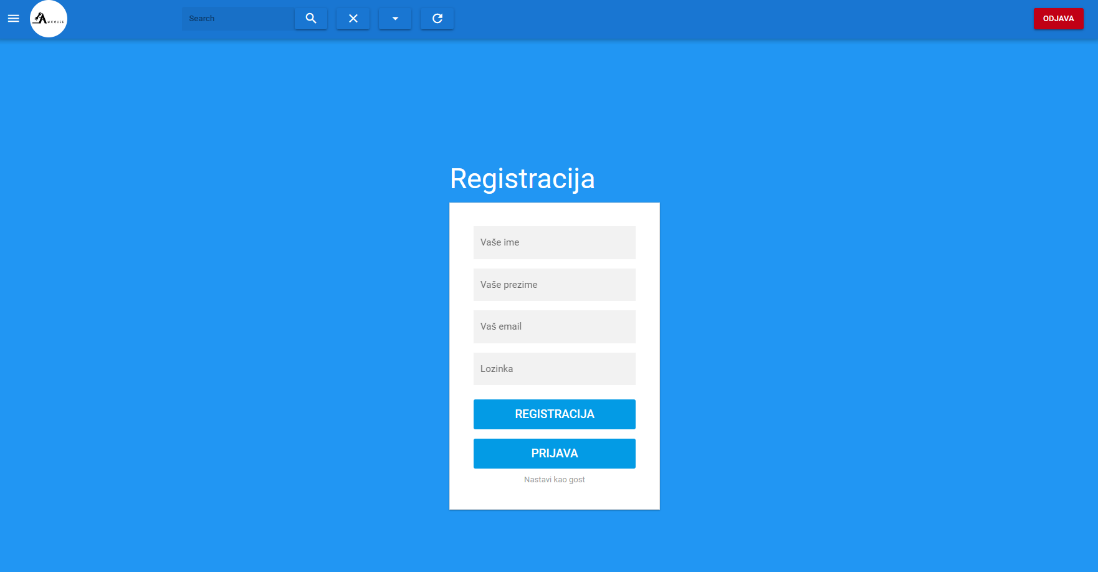
Slika na kojoj se prikazuje tekst, dijagram, snimka zaslona, paralelno

Opis je automatski generiran

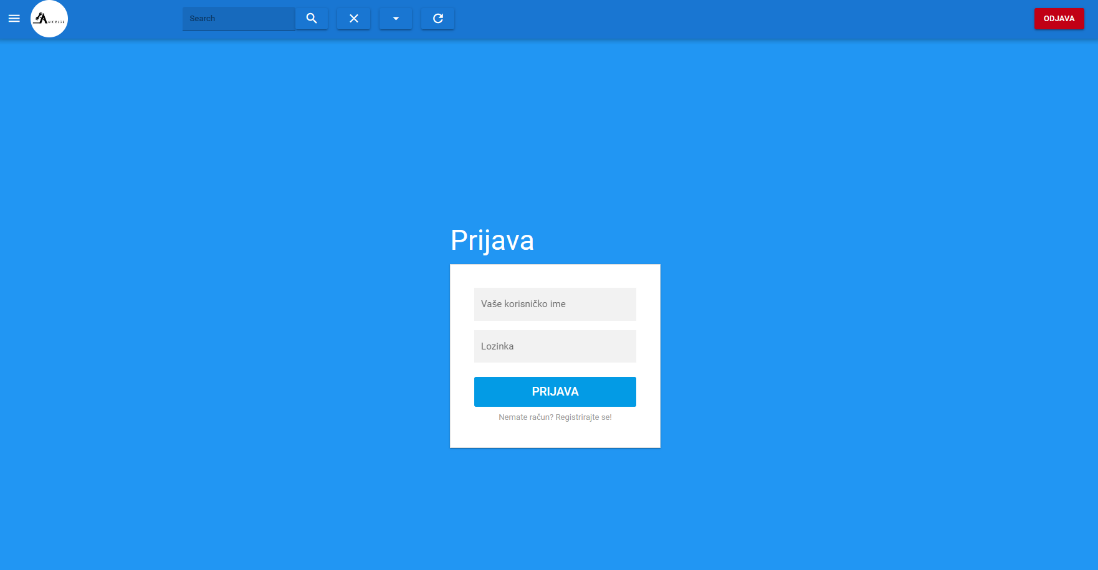
Slika 6. Relacijski model podataka

# Grafičko korisničko sučelje

Na slici 7 je prikazano sučelje za registraciju korisnika. Unosi se ime, prezime, email te lozinka te se korisnik registrira. Nakon registracije korisnik se može prijaviti unosom korisničkog imena i lozinke, preko sučelja koje je prikazano na slici 8. Registracija i prijava još nisu realizirane.

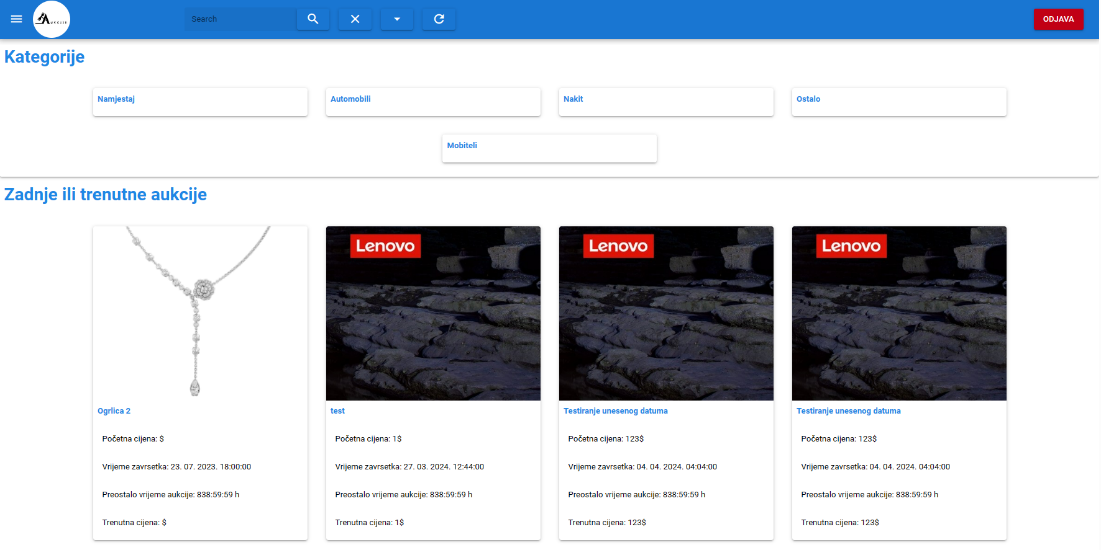


Slika 7 Registracija

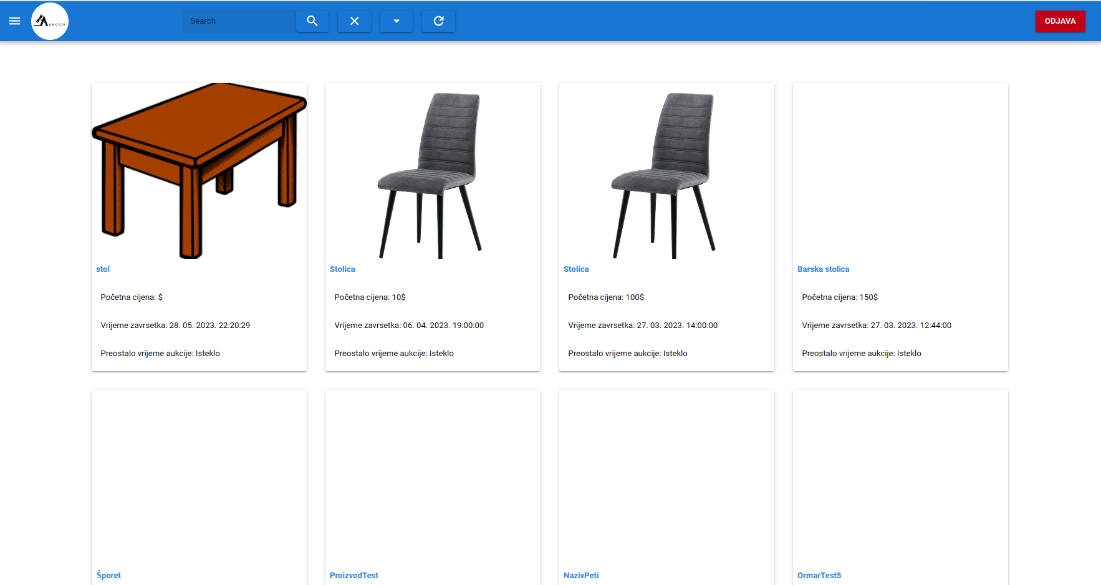


Slika 8 Prijava

Na slici 9 prikazana je početna stranica. Ovdje se prikazuju sve aukcije sa osnovnim informacijama – slika, naziv, početna cijena, vrijeme završetka aukcije, preostalo vrijeme aukcije te trenutna cijena. Na vrhu su prikazane moguće kategorije u koje se svrstavaju predmeti kod unosa aukcije. Klikom na kategoriju “Namještaj” nude se sve aukcije koje su postavljene u toj kategoriji što se može vidjeti na slici 10.

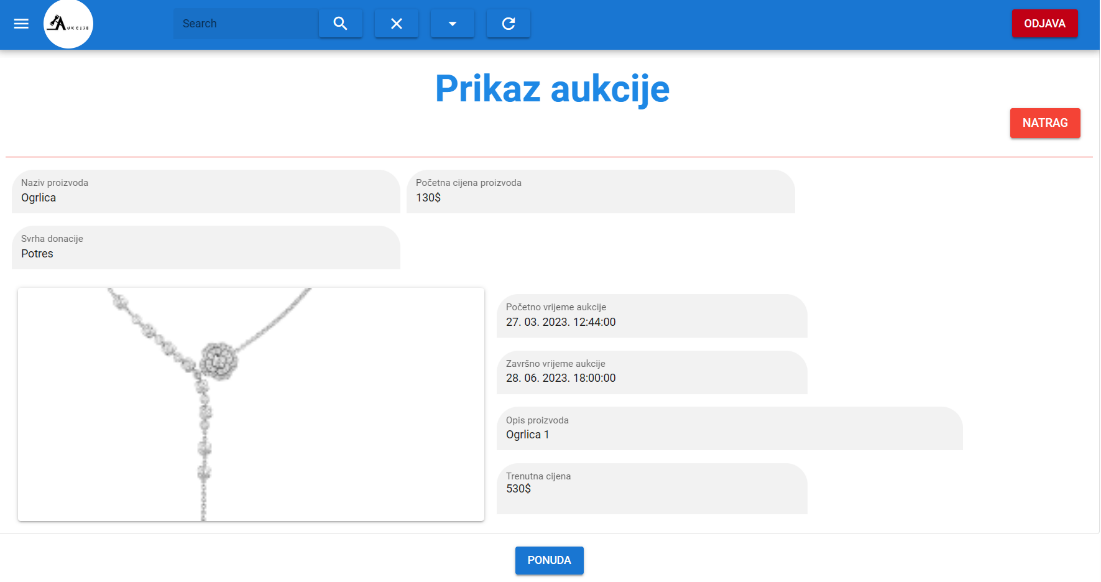


Slika 9 Početna stranica (prikaz aukcija)

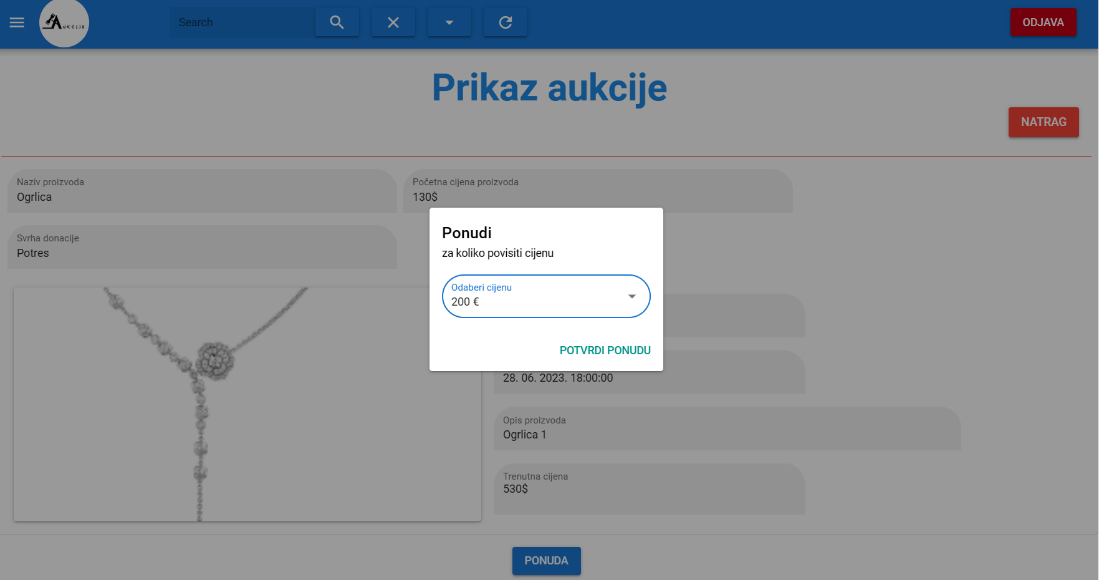


Slika 10 Prikaz predmeta iz kategorije “Namještaj”

Slika 11 prikazuje sučelje prikaza aukcije na kojem se vide svi podaci – naziv, slika, svrha donacije, početna cijena proizvoda, početno vrijeme aukcije, završno vrijeme aukcije, opis proizvoda te trenutna cijena. Na dnu se nalazi gumb “Ponuda”. Klikom na njega otvara se opcija za povisiti cijenu aukcije kao što je prikazano na slici 6. Odabirom opcije “200 €”, trenutna cijena se povisuje na 730 € te tako rade licitacije.

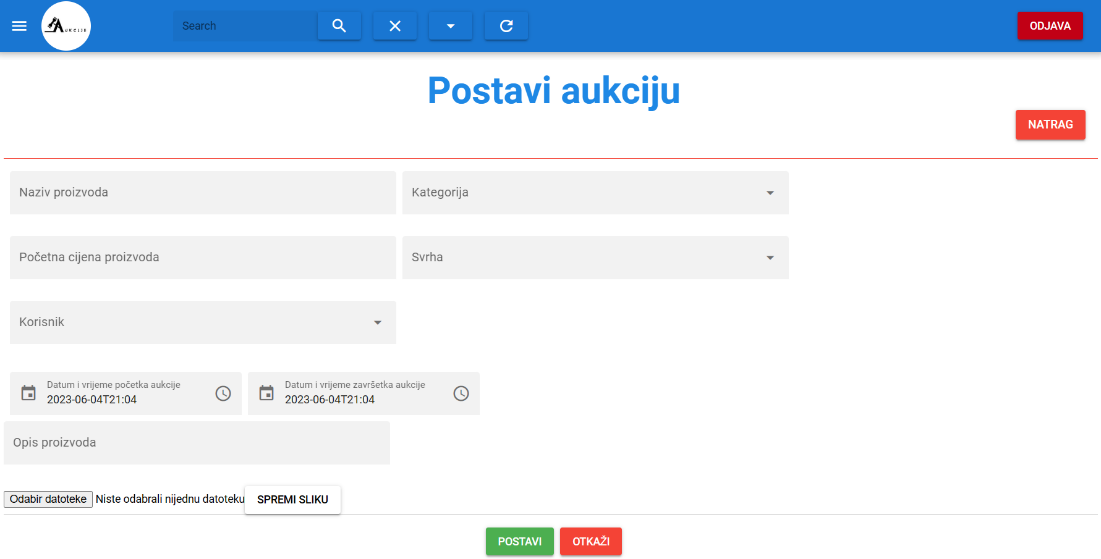


Slika 11 Prikaz pojedine aukcije



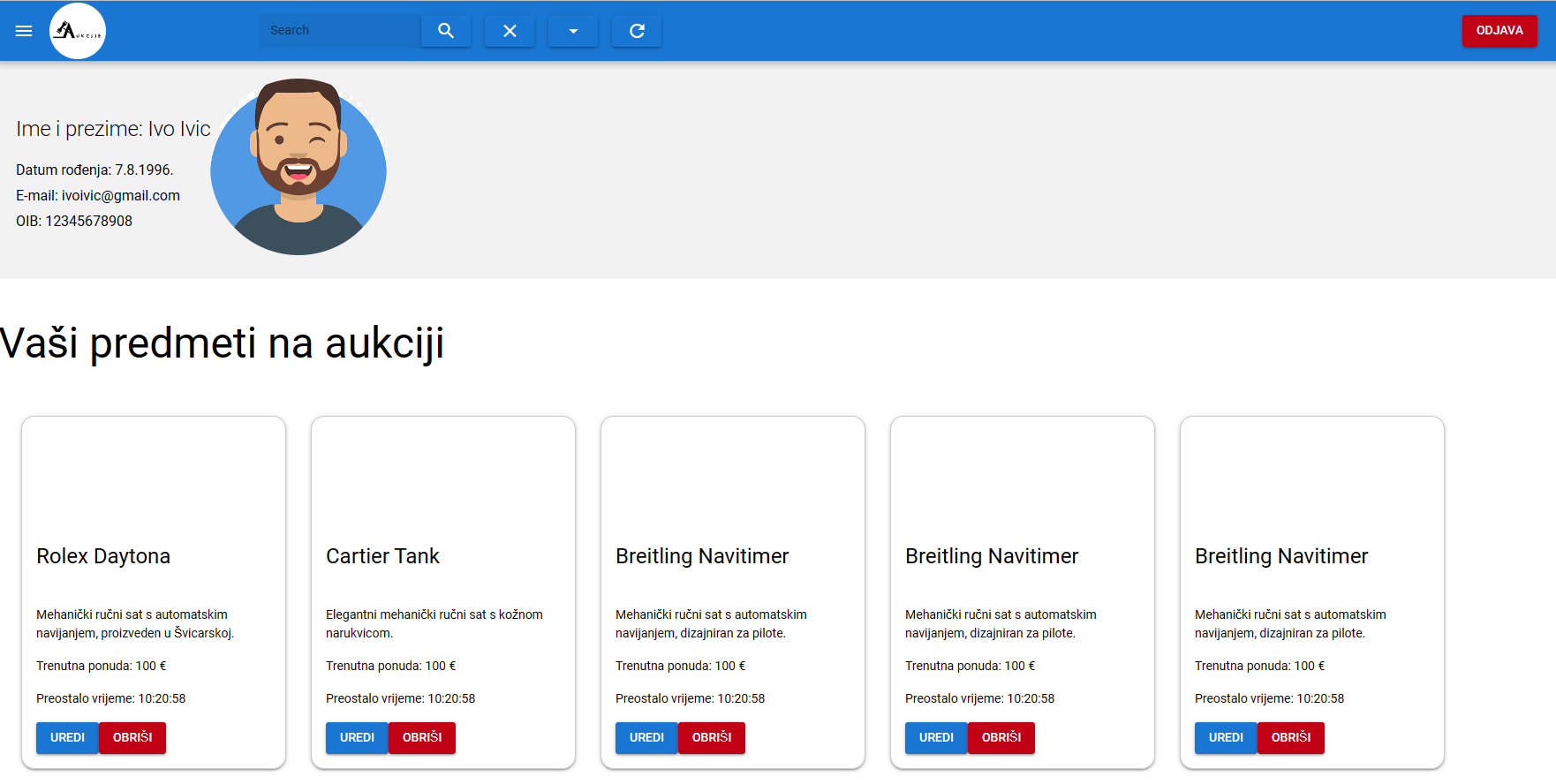
Slika 12 Podizanje cijene predmeta na aukciji za određeni iznos

Slika 13 prikazuje sučelje za postavljanje aukcije. Ovdje se unosi naziv proizvoda, početna cijena, odabire se korisnik, kategorija, humanitarna svrha, datum i vrijeme početka i završetka aukcije, opis proizvoda te slika predmeta. Nudi se mogućnost “Postavi” koja sprema aukciju ili “Otkaži” koja briše unesena polja.



Slika 13 Postavljanje aukcije

Na slici 14 prikazano je sučelje profila korisnika. Ovdje se vide podaci korisnika – ime i prezime, datum rođenja, e-mail te OIB korisnika. Ispod se nudi popis predmeta na aukciji koje je postavio taj korisnik. Za svaki predmet nudi se opcija “Uredi” za uređivanje ili “Obriši” za brisanje vlastite aukcije. Te dvije opcije još nisu funkcionalne.



Slika 14 Profil korisnika

# Kreiranje API-ja

Uloga serverskog dijela aplikacije je omogućiti komunikaciju između baze podataka i klijentskog dijela aplikacije odnosno omogućiti CRUD (engl. *Create, Read, Update, Delete*) funkcionalnosti putem RESTful arhitekture, te će izvršavati operacije spremanja podataka. Sam serverski dio aplikacije izgrađen je prema RESTful arhitekturi koristeći URI (engl. *Uniform Resource Identifier*) u kombinaciji s HTTP metodama POST, PUT, PATCH, DELETE i GET.

Za izradu serverskog dijela aplikacije korišten je razvojni okvir Express.js, te su same akcije definirane kao putanje (engl. *route*) koristeći HTTP metode. Prije implementacije krajnjeg rješenja potrebno definirane putanje koje će se koristiti za pristup resursima potrebnim za rad web-aplikacije. Krajnje putanje resursa prikazane su tablicom.

Tablica 1 - Krajnje putanje resursa

|  |  |  |
| --- | --- | --- |
| **URL** | **HTTP metoda** | **Akcija** |
| Resurs: Predmet na aukciji | | |
| **http://localhost:3000/** **all-predmet** | **GET** | **Dohvat svih predmeta na aukciji** |
| **http://localhost:3000/** **get-predmet/:id** | **GET** | **Dohvat jednog predmeta na aukciji** |
| **http://localhost:3000/** **get-predmet/:id** | **GET** | **Dohvat predmeta prema ID-u kategorije** |
| **http://localhost:3000/** **unosPredmeta** | **POST** | **Unos novog predmeta** |
| Resurs: Korisnik | | |
| **http://localhost:3000/** **all-korisnik** | **GET** | **Dohvat svih korisnika** |
| Resurs: Kategorija predmeta na aukciji | | |
| **http://localhost:3000/** **all-kategorija** | **GET** | **Dohvat svih kategorija** |
| **http://localhost:3000/getxx/** **get-kategorija-predmet/:id** | **GET** | **Dohvat predmeta prema ID-u kategorije** |
| Resurs: Ponuda | | |
| **http://localhost:3000/** **get-ponuda/:id** | **GET** | **Dohvat ponuda za određeni predmet** |
| **http://localhost:3000/** **/unostrenutnaponuda** | **POST** | **Unos trenutne ponude za predmet** |
|  |  |  |
| Resurs:Slika | | |
| **http://localhost:3000/** **unos-slike** | **POST** | **Unos slike predmeta** |
|  |  |  |
| Resurs:Trenutna cijena | | |
| **http://localhost:3000/get-predmet-trenutna-cijena/:id** | **GET** | **Dohvat trenutne cijene za određenu za ponuđeni predmet** |
| **http://localhost:3000/** **all-predmet-with-current-price** | **GET** | **Dohvat svih trenutni cijena za sve predmete** |

# IMPLEMENTACIJA SUSTAVA

## Prikaz svih predmeta na aukciji

Kako bi prikazali sve premete na aukciji koristimo „mounted“ metodu koja popuni lokalnu varijablu „items“ sa rezultatima sql upita. „Mounted“ metode se izvršavaju prilikom svakog kreiranja vue komponente, te na taj način dobijamo osvježene podatke prilikom svakog posjeta toj stranici.

SQL upita (ovaj upit jos dodatno filtrira na način da prikazuje samo trenutne odnosno aktivne aukcije):

SELECT

p.sifra\_predmeta,

p.naziv\_predmeta,

p.slika,

p.pocetna\_cijena,

p.vrijeme\_zavrsetka,

TIME\_FORMAT(SEC\_TO\_TIME(TIMESTAMPDIFF(SECOND, NOW(), p.vrijeme\_zavrsetka)), '%H:%i:%s') AS preostalo\_vrijeme,

COALESCE(MAX(po.vrijednost\_ponude), p.pocetna\_cijena) AS trenutna\_cijena

FROM

predmet p

LEFT JOIN

ponuda po ON p.sifra\_predmeta = po.sifra\_predmeta

WHERE

p.vrijeme\_zavrsetka > NOW()

GROUP BY

p.sifra\_predmeta

ORDER BY

preostalo\_vrijeme DESC

Dio koda koji je zaslužan za slanje upita na backend i spremanje rezultata u lokalnu varijablu:

const baseUrl = "http://localhost:3000/api/";

export default {

setup() {

return {};

},

data() {

return {

items: [],

kategorija: [],

};

},

mounted() {

axios.get(baseUrl + "all-predmet-with-current-price").then((response) => {

this.items = response.data;

});

},

Realizacija endpointa na backendu:

app.get("/api/all-predmet-with-current-price", (req, res) => {

connection.query(

"SELECT p.sifra\_predmeta, p.naziv\_predmeta, p.slika, p.pocetna\_cijena, p.vrijeme\_zavrsetka, TIME\_FORMAT( SEC\_TO\_TIME(TIMESTAMPDIFF(SECOND, NOW(), p.vrijeme\_zavrsetka)), '%H:%i:%s' ) AS preostalo\_vrijeme, COALESCE(MAX(po.vrijednost\_ponude), p.pocetna\_cijena) AS trenutna\_cijena FROM predmet p LEFT JOIN ponuda po ON p.sifra\_predmeta = po.sifra\_predmeta WHERE p.vrijeme\_zavrsetka > NOW() GROUP BY p.sifra\_predmeta ORDER BY preostalo\_vrijeme DESC",

(error, results) => {

if (error) throw error;

res.send(results);

}

);

});

## Prikaz jednog predmeta na aukciji

**SQL Upit za dohvat jednog predmeta pkreko Id-a.**

Sa SELECT-om oizbnačimo koje sve podatke trebamo dohvatiti

TIME\_FORMAT služi za formatiranje dohvaćenog vremena

TIMESTAMPDIFF se koristi za izračunavanje razlike između dvije vremenske jedinice

COALESCE ukoliko je neki član null coalesce vraća sljedeći

FROM koristimo kako bi znali na kojoj tablici se izvršava upit

LEFT\_JOIN neophodan je za dohvaćanje podataka iz više različitih tablica

WERE uvjet, odnosno u našem slučaju usporedimo id koji šaljemo sa frontenda sa onima iz tablice

**SELECT** p.naziv\_predmeta,

p.sifra\_predmeta,

p.slika,

p.pocetna\_cijena,

p.vrijeme\_pocetka,

p.vrijeme\_zavrsetka,

TIME\_FORMAT(SEC\_TO\_TIME(TIMESTAMPDIFF(**SECOND**, p.vrijeme\_pocetka, p.vrijeme\_zavrsetka)), '%H:%i:%s') **AS** preostalo\_vrijeme,

p.svrha\_donacije,

p.opis\_predmeta,

COALESCE(**MAX**(po.vrijednost\_ponude), p.pocetna\_cijena) **AS** vrijednost\_ponude

**FROM** predmet p

**LEFT** **JOIN** ponuda po **ON** p.sifra\_predmeta = po.sifra\_predmeta

**WHERE** p.sifra\_predmeta = ?

Nakon što na backend stigne GET zahtjev na toj rut *(/api/get-predmet/:id*), kod izvršava SQL upit na bazi podataka (phpMyAdmin) koristeći primarni ključ predmeta (*id*) koji je dobiven kao parametar. Upit dohvaća informacije o nazivu predmeta, šifri predmeta, slici, početnoj cijeni, vremenu početka, vremenu završetka, preostalom vremenu, svrsi donacije, opisu predmeta i vrijednosti najviše ponude za taj predmet.

Ako se prilikom izvršavanja SQL upita javi greška, bacit će exeption (grešku). Ako nema greške, rezultati SQL upita će se poslati na frontend kao odgovor na zahtjev.

app.get("/api/get-predmet/:id", (req, res) => {

const { id } = req.params;

connection.query(

"SELECT p.naziv\_predmeta, p.sifra\_predmeta, p.slika, p.pocetna\_cijena, p.vrijeme\_pocetka, p.vrijeme\_zavrsetka, TIME\_FORMAT( SEC\_TO\_TIME(TIMESTAMPDIFF(SECOND, p.vrijeme\_pocetka, p.vrijeme\_zavrsetka)), '%H:%i:%s' ) AS preostalo\_vrijeme, p.svrha\_donacije, p.opis\_predmeta, COALESCE(MAX(po.vrijednost\_ponude), p.pocetna\_cijena) AS vrijednost\_ponude FROM predmet p LEFT JOIN ponuda po ON p.sifra\_predmeta = po.sifra\_predmeta WHERE p.sifra\_predmeta = ?",

[id],

(error, results) => {

if (error) throw error;

res.send(results);

}

);

});

*baseUrl* definira osnovni URL na kojem se nalazi API (u našem slučaju *http://localhost:3000/api*).

*data()* je funkcija koja vraća početno stanje podataka komponente. U ovom slučaju, item je prazan niz.

*mounted()* je hook (metoda) koja se poziva kod samog renderiranja. Ovaj hook koristi Axios za izvršavanje HTTP GET zahtjeva prema određenom URL-u koji se sastoji od *baseUrl* i "*get-predmet/*" + *this.sifra\_predmeta*.

*this.sifra\_predmeta* je nepohodno slati u zahtjevu kako bi SQL upit znao koji nam točno podatci trebaju odnosno za koji predmet it baze.

const baseUrl = "http://localhost:3000/api/";

export default {

setup() {

return {};

},

data() {

return {

item: [],

};

},

mounted() {

axios

.get(baseUrl + "get-predmet/" + this.sifra\_predmeta, {})

.then((response) => {

this.item = response.data[0];

});

},

## Davanje ponude za predmet na aukciji

API (Application Programming Interface) je skup pravila i protokola koji omogućavaju komunikaciju između različitih softverskih komponenti. U kontekstu web razvoja, API se obično odnosi na sučelje koje omogućava komunikaciju između klijentske strane (frontend) i poslužiteljske strane (backend).

Komunikacija između frontend aplikacije (Vue.js) i API-ja (Express.js) obično se odvija putem HTTP zahtjeva. U našem primjeru HTTP GET zahtjev šalje se s frontend aplikacije za dohvaćanje podataka, a zatim se dobiveni rezultati prikazuju u sučelju korisniku. Također postoji HTTP POST zahtjev koji se šalje kada korisnik potvrdi ponudu.

Uzorak komunikacije između frontend i API-ja u našem primjeru uključuje sljedeće korake:

1. Na frontendu se koristi **axios** biblioteka za izvršavanje HTTP GET ili POST zahtjeva prema odgovarajućim API endpointima.
2. Kada se izvrši HTTP GET zahtjev, frontend prima odgovor s podacima iz API-ja, a ti podaci se mogu koristiti za ažuriranje korisničkog sučelja ili manipulaciju podacima.
3. Kada korisnik potvrdi ponudu, frontend šalje HTTP POST zahtjev s odgovarajućim podacima ponude na API endpoint **/unostrenutnaponuda**. API obrađuje taj zahtjev i vrši umetanje ponude u bazu podataka.

**Frontend kod**

<div class="q-pa-md flex flex-center">

<q-btn label="Ponuda" color="primary" @click="showDialog = true" />

<q-dialog v-model="showDialog">

<q-card style="width: 300px">

<q-card-section>

<div class="text-h6">Ponudi</div>

</q-card-section>

<q-card-section class="q-pt-none">

<q-select

rounded

outlined

v-model="odabranaCijena"

:options="prices"

label="Odaberi cijenu"

/>

</q-card-section>

<q-card-actions align="right" class="bg-white text-teal">

<q-btn flat label="Potvrdi ponudu" @click="potvrdiPonudu" />

</q-card-actions>

</q-card>

</q-dialog>

</div>

</template>

<script>

Navedeni kod predstavlja HTML i JavaScript kod koji implementira dijalog za ponudu u okviru korisničkog sučelja. Ovaj kod koristi Quasar Framework za izradu responzivnog korisničkog interfejsa. Evo kratak opis dijela koda:

1. Događajni handler "onClick" na gumbu "Ponuda" otvara dijalog kada se klikne na njega.
2. Dijalog je definiran pomoću **<q-dialog>** komponente. Kada se postavi v-model atribut na true, dijalog će biti prikazan, a kada se postavi na false, dijalog će biti skriven.
3. **<q-card>** komponenta koristi se za definiranje kartice koja sadrži sadržaj dijaloga.
4. **<q-card-section>** komponenta se koristi za definiranje sekcija unutar kartice. Prva sekcija sadrži naslov "Ponudi".
5. **<q-select>** komponenta koristi se za prikazivanje padajućeg izbornika cijena. Vrijednost odabrane cijene bind-uje se na v-model atribut "odabranaCijena".
6. **<q-card-actions>** komponenta koristi se za definiranje akcija koje se nalaze na dnu kartice. U ovom slučaju, prikazuje se samo gumb "Potvrdi ponudu".
7. Događajni handler "onClick" na gumbu "Potvrdi ponudu" poziva funkciju "potvrdiPonudu".
8. JavaScript kod koji se nalazi ispod HTML dijela koristi Vue.js sintaksu za definiranje funkcija i varijabli koje se koriste u dijalogu.

import { ref } from "vue";

import axios from "axios";

const baseUrl = "http://localhost:3000/api/";

export default {

computed: {

sifra\_predmeta() {

return this.$route.query.sifra\_predmeta;

},

},

data() {

return {

item: [],

showDialog: false,

odabranaCijena: null,

prices: [

{ label: "100 €" },

{ label: "200 €" },

{ label: "300 €" },

{ label: "400 €" },

{ label: "500 €" },

{ label: "1000 €" },

],

potvrdjenaCijena: null,

};

},

mounted() {

axios

.get(baseUrl + "get-predmet-trenutna-cijena/" + this.sifra\_predmeta, {})

.then((response) => {

this.item = response.data[0];

});

},

methods: {

potvrdiPonudu() {

if (this.odabranaCijena) {

this.potvrdjenaCijena = this.odabranaCijena;

this.showDialog = false;

}

},

},

Gore navedeni dio koda predstavlja Vue.js komponentu s definiranim podacima i metodama. Kratak opis dodatnog dijela koda:

1. **data()** metoda: Ova metoda vraća objekt koji sadrži podatke koji se koriste u komponenti. Ovdje su definirane sljedeće varijable:
   * **item**: Prazan niz koji će se koristiti za pohranjivanje podataka dobivenih iz API poziva.
   * **showDialog**: Boolean varijabla koja je inicijalno postavljena na **false** i koristi se za kontrolu prikaza dijaloga.
   * **odabranaCijena**: Varijabla koja je inicijalno postavljena na **null** i koristi se za praćenje odabrane cijene iz padajućeg izbornika.
   * **prices**: Polje koje sadrži opcije cijena prikazane u padajućem izborniku.
   * **potvrdjenaCijena**: Varijabla koja je inicijalno postavljena na **null** i koristi se za prikaz potvrđene cijene.
2. **mounted()** metoda: Ova metoda se izvršava nakon što se komponenta uspješno montira na stranicu. U ovoj metodi se koristi Axios za izvršavanje HTTP GET zahtjeva prema određenom URL-u kako bi se dobili podaci o predmetu na temelju njegovog ID-a (**sifra\_predmeta**). Dobiveni podaci se pohranjuju u varijablu **item** komponente.
3. **potvrdiPonudu()** metoda: Ova metoda se poziva kada korisnik klikne na gumb "Potvrdi ponudu". Provjerava se je li **odabranaCijena** postavljena (nije **null**). Ako je odabrana cijena postavljena, njena vrijednost se dodjeljuje varijabli **potvrdjenaCijena**, a zatim se postavlja **showDialog** na **false** kako bi se zatvorio dijalog.

Ovaj dio koda proširuje prethodni primjer s funkcionalnostima za dohvaćanje podataka putem Axiosa i prikazivanje odabrane cijene iz padajućeg izbornika.

**Backend kod**

const express = require("express");

const cors = require("cors");

const bodyParser = require("body-parser");

const mysql = require("mysql");

const { join } = require("path");

const path = require("path");

const app = express();

const port = 3000;

// Parser za JSON podatke

app.use(bodyParser.json());

// Parser za podatke iz formi

app.use(bodyParser.urlencoded({ extended: true }));

// Postavke direktorija za statičke datoteke

app.use(express.static(path.join(\_\_dirname, "public")));

app.use(cors({ origin: "\*" }));

const connection = mysql.createConnection({

host: "student.veleri.hr",

user: "iooa-aukcije",

password: "11",

database: "iooa-aukcije",

});

connection.connect();

app.post('/unostrenutnaponuda', function (request, response) {

const data = request.body;

predmet = [[ data.id\_ponude, data.vrijednost\_ponude, data.vrijeme\_ponude, data.id\_korisnika, data.sifra\_predmeta]]

connection.query('INSERT INTO ponuda (id\_ponude, vrijednost\_ponude, vrijeme\_ponude, id\_korisnika, sifra\_predmeta) VALUES ?',

[predmet], function (error, results, fields) {

if (error) throw error;

return response.send({ error: false, data: results, message: 'Dodali se trenutnu ponudu.' });

});

});

app.get('/api/get-predmet-trenutna-cijena/:id', (req, res) => {

const { id } = req.params;

connection.query(

"SELECT p.sifra\_predmeta, p.naziv\_predmeta, p.opis\_predmeta, p.svrha\_donacije, p.slika, p.pocetna\_cijena, p.vrijeme\_pocetka, p.vrijeme\_zavrsetka, TIME\_FORMAT(SEC\_TO\_TIME(TIMESTAMPDIFF(SECOND, NOW(), p.vrijeme\_zavrsetka)), '%H:%i:%s') AS preostalo\_vrijeme, COALESCE(po.trenutna\_cijena, p.pocetna\_cijena) AS trenutna\_cijena FROM predmet p LEFT JOIN (SELECT sifra\_predmeta, MAX(vrijednost\_ponude) AS trenutna\_cijena FROM ponuda GROUP BY sifra\_predmeta) po ON p.sifra\_predmeta = po.sifra\_predmeta WHERE p.sifra\_predmeta = ?",

[id],

(error, results) => {

if (error) throw error;

res.send(results);

}

);

});

app.listen(port, () => {

console.log('Server running at port: ' + port);

});

Navedeni kod predstavlja backend implementaciju aplikacije koristeći Express.js, CORS, Body Parser i MySQL biblioteke. Backend je odgovoran za obradu zahtjeva klijenta, komunikaciju s bazom podataka i slanje odgovora natrag klijentu.

Kratak opis navedenog koda:

1. Uvoz potrebnih biblioteka: `express`, `cors`, `body-parser`, `mysql` i `path`.

2. Inicijalizacija Express aplikacije i postavljanje porta na 3000.

3. Konfiguracija parsera za JSON podatke i podatke iz formi koristeći `body-parser` biblioteku.

4. Postavljanje direktorija za statičke datoteke pomoću `express.static` kako bi se poslužile javne datoteke.

5. Postavljanje CORS konfiguracije pomoću `cors` biblioteke kako bi se omogućili zahtjevi s različitih domena.

6. Stvaranje MySQL konekcije prema određenoj bazi podataka s postavkama za host, korisnika, lozinku i bazu podataka.

7. Pozivanje metode `connect()` kako bi se uspostavila veza s bazom podataka.

8. Definiranje POST endpointa "/unostrenutnaponuda" koji obrađuje zahtjeve za unos trenutne ponude. Podaci se dohvaćaju iz zahtjeva i koriste se za izvršavanje SQL upita putem `connection.query()` metode.

9. Definiranje GET endpointa "/api/all-predmet-with-current-price" koji dohvaća sve predmete s trenutnom cijenom. Izvršava se SQL upit koji kombinira podatke iz tablica "predmet" i "ponuda" koristeći lijevu vezu. Rezultati se šalju kao odgovor.

10. Definiranje GET endpointa "/api/get-predmet-trenutna-cijena/:id" koji dohvaća određeni predmet i njegovu trenutnu cijenu na temelju ID-a predmeta. Izvršava se SQL upit koji kombinira podatke iz tablice "predmet" i "ponuda" koristeći lijevu vezu. Rezultati se šalju kao odgovor.

11. Pokretanje servera na određenom portu i ispis poruke u konzoli.

Ovaj backend kod omogućava obradu zahtjeva klijenta, komunikaciju s bazom podataka i slanje odgovora natrag klijentu. Važno je napomenuti da se u kodu koriste postavke kao što su host, korisnik, lozinka i baza podataka koje treba prilagoditi specifičnostima projekta. Također, treba provjeriti i osigurati ispravnu konfiguraciju MySQL konekcije.

Opisani kod prikazuje implementaciju dijela korisničkog sučelja i backend logike za komunikaciju s bazom podataka. Ovdje je kratak opis komunikacije između frontenda i API-ja:

1. Frontend komponenta:
   * Događajni handler "onClick" na gumbu "Ponuda" otvara dijalog kada se klikne na njega.
   * Korisnik odabire cijenu iz padajućeg izbornika.
   * Događajni handler "onClick" na gumbu "Potvrdi ponudu" poziva funkciju "potvrdiPonudu".
2. API endpointi:
   * GET "/api/get-predmet-trenutna-cijena/:id" - Dohvaća podatke o određenom predmetu i njegovu trenutnu cijenu na temelju ID-a predmeta.
   * POST "/unostrenutnaponuda" - Unosi trenutnu ponudu u bazu podataka.
3. Frontend komunikacija s API-jem:
   * Kada se komponenta učita (mounted metoda), frontend šalje GET zahtjev prema "/api/get-predmet-trenutna-cijena/:id" endpointu kako bi dobio podatke o odabranom predmetu.
   * Kada korisnik potvrdi ponudu (klikne na gumb "Potvrdi ponudu"), frontend šalje POST zahtjev prema "/unostrenutnaponuda" endpointu s podacima o ponudi (id\_ponude, vrijednost\_ponude, vrijeme\_ponude, id\_korisnika, sifra\_predmeta).
4. Backend komunikacija s bazom podataka:
   * Backend prima GET zahtjev na "/api/get-predmet-trenutna-cijena/:id" endpointu i izvršava SQL upit koji dohvaća podatke o predmetu i trenutnoj cijeni iz baze podataka. Rezultati se šalju kao odgovor.
   * Backend prima POST zahtjev na "/unostrenutnaponuda" endpointu i izvršava SQL upit koji unosi trenutnu ponudu u bazu podataka. Ovisno o rezultatu upita, šalje se odgovor natrag frontendu.

## Unos predmeta za aukciju

API za **postavljanje aukcije** je realiziran putem POST metode na **/unosPredmeta** endpointu.

POST metodu koristimo kada nešto želimo postaviti i/ili ažurirati u bazi podataka. U ovome je to slučaju postavljanje predmeta na aukciju, što znači kako se određeni podaci postavljaju u bazu podataka tj. Nova aukcija, odnosno predmet.

API Endpoint:

* POST /unosPredmeta

**API za unos predmeta na aukciju je implementiran kroz POST rutu koja je definirana u ovom dijelu koda (PostaviAukciju.vue – middleware)**

app.post('/unosPredmeta', function (request, response) {

const data = request.body;

predmet = [[data.sifra\_predmeta, data.naziv\_predmeta, data.opis\_predmeta, data.slika, data.vrijeme\_pocetka, data.vrijeme\_zavrsetka, data.pocetna\_cijena, data.svrha\_donacije, data.id\_korisnika, data.sifra\_kategorije]]

connection.query('INSERT INTO predmet (sifra\_predmeta, naziv\_predmeta, opis\_predmeta, slika, vrijeme\_pocetka, vrijeme\_zavrsetka, pocetna\_cijena, svrha\_donacije, id\_korisnika, sifra\_kategorije) VALUES ?',

[predmet], function (error, results, fields) {

if (error) throw error;

return response.send({ error: false, data: results, message: 'Predmet je dodan.' });

});

});

* 1. **OPIS POST RUTE/API-ja**
  2. ***'App.post'*** – nova HTTP POST ruta
  3. ***'const data'*** = request.body;' - : Ovdje se podaci iz tijela POST zahtjeva (koji su poslani sa klijenta) spremaju u konstantu "data".
  4. ***'predmet = [[data.sifra\_predmeta, ...]]' –*** linija koda koja pokreće SQL upit za unos u bazu podataka
  5. ***'*** ***if (error) throw error;' -*** Ako se prilikom izvršavanja SQL upita dogodi greška.
  6. ***'*** ***return response.send({ error: false, data: results, message: 'Predmet je dodan.' });' -*** Kada SQL upit uspješno izvrši, server će odgovoriti na klijentov zahtjev slanjem JSON objekta koji sadrži rezultate upita i poruku koja potvrđuje da je predmet uspješno dodan. U slučaju greške, ovaj dio linije koda neće biti uopće dosegnut.

**SQL upit koji se koristi za unos novog predmeta je:**

*INSERT INTO predmet (sifra\_predmeta, naziv\_predmeta, opis\_predmeta, slika, vrijeme\_pocetka, vrijeme\_zavrsetka, pocetna\_cijena, svrha\_donacije, id\_korisnika, sifra\_kategorije) VALUES ?*

**2.1. OPIS SQL UPITA**

Ovaj SQL upit koristi **INSERT INTO** naredbu, koja se koristi za dodavanje novog zapisa u tablicu **predmet**. Evo što točno svaka komponenta upita znači:

* ***'INSERT INTO predmet'***: Ovaj dio govori SQL-u da ćemo dodavati novi zapis u tablicu nazvanu **predmet**.
* ***'(sifra\_predmeta, naziv\_predmeta, opis\_predmeta, slika, vrijeme\_pocetka, vrijeme\_zavrsetka, pocetna\_cijena, svrha\_donacije, id\_korisnika, sifra\_kategorije)'***: Ovaj dio navodi imena stupaca u tablici **predmet** u koje će se unijeti podaci.
* **'VALUES ?'**: Ovo je parametrizirani dio upita koji predstavlja vrijednosti koje će biti unesene u svaki od gore navedenih stupaca (*sifra\_predmeta, naziv\_predmeta,..*.). Umjesto konkretnih vrijednosti, postavljen je upitnik **?**. Taj upitnik omogućuje dodavanje vrijednosti iz varijabli ili iz drugih izvora podataka, odnosno upitnik će biti zamijenjen nizom vrijednosti koji su dobiveni iz POST zahtjeva, odnosno unosom podataka na frontendu, onim redostljedom kako je definirano u SQL upitu.

**Komunikacija frontenda s API-jem:**

Kada korisnik želi postaviti aukciju, korisnik popunjava formu na frontend-u (PostaviAukciju.vue). Nakon što se forma ispunjena, aplikacija šalje POST zahtjev na **/unosPredmeta** endpoint s potrebnim podacima za kreiranje aukcije. Ovi podaci uključuju **sifra\_predmeta**, **naziv\_predmeta**, **opis\_predmeta**, **slika**, **vrijeme\_pocetka**, **vrijeme\_zavrsetka**, **pocetna\_cijena**, **svrha\_donacije**, **id\_korisnika** i **sifra\_kategorije**.

Na backendu, nakon što se primi POST zahtjev, podaci se izvlače iz tijela zahtjeva. Zatim se podaci koriste za izvršavanje SQL upita koji unosi novi redak u tablicu **predmet**.

Nakon uspješnog izvršavanja upita, server šalje odgovor natrag frontendu s informacijama o rezultatima operacije. Frontend zatim koristi te informacije za obavijest korisnika o uspješnosti operacije.

1. u bazi podataka na phpmyadminu postoje također tablice "humanitarna\_svrha" i "transakcija" koje nisam uključio u model podataka (EVA) iz razloga što se u njima ništa ne nalazi, nikakvi podaci tj. nisu uključeni u aplikaciju, pa iz tog razloga ih nisam nacrtao na dijagramu. [↑](#footnote-ref-1)