Web platforma za korisnike TCG zajednice

Sertić, Ivan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij

WEB PLATFORMA ZA KORISNIKE TCG ZAJEDNICE

Diplomski rad

Ivan Sertić

Osijek, 2021. godina.

SADRŽAJ

1. UVOD

Igre razmjene karata (engl. *Trading Card Games*) pojavile su se 1993. godine. Prva takva igra bila je *Magic: The Gathering*. U počecima razmjena karata odvijala se neposredno između prijatelja i mali lokalnih trgovina koje su prodavale takve igre, no porastom popularnosti dolazi do pojave kompetitivnih turnira za koje je bilo potrebno nabaviti određene karte te su se karte počele prodavati. Razvojem tehnologije, a naročito interneta i web aplikacija, i porastom potražnje za kartama, web aplikacije preuzimaju jednu od glavnih uloga prodaje ovakvih igara te su dosadašnje načine razmjene i prodaje proširile sa lokalnih zajednica na cijeli svijet. Danas se karte preko postojećih platformi mogu kupovati diljem svijeta, no to zahtijeva puno truda jer se svaka karta mora unositi posebno. Kada su u pitanju aukcije one se odvijaju većinom preko društvene mreže *Facebook,* preko odjeljka za komentare. Ovakav pristup često uzrokuje nepoželjne notifikacije koje nužno ne obavještavaju sudionika aukcije o novoj ponudi već i ostalim komentarima koji ne moraju imati veze s aukcijom. Uočavanjem grešaka i nedostataka postojećih platformi nastala je web platforma koja nudi univerzalno rješenje. Web platforma za korisnike TCG zajednice omogućuje jednostavan način postavljanja slika svih karata koje se prodaju, traže ili su za razmjenu te omogućuje pošten i brz način postavljanja ponuda na aukcijama kako bi korisnici bez nepotrebnih komentara mogli znati tko je postavio najveću ponudu.

Drugo poglavlje ovog rada sadrži pregled postojećih rješenja te su ukratko opisani načini korištenja istih. Treće poglavlje ukratko opisuje primijenjene tehnologije korištene u izradi platforme. U četvrtom poglavlju opisana je arhitektura platforme, načini pohrane podataka te struktura baze podataka. Peto poglavlje opisuje načine korištenja platforme. Zaključak prikazuje osvrt na izradu platforme i diplomskog rada

1.2. Tekst zadatka

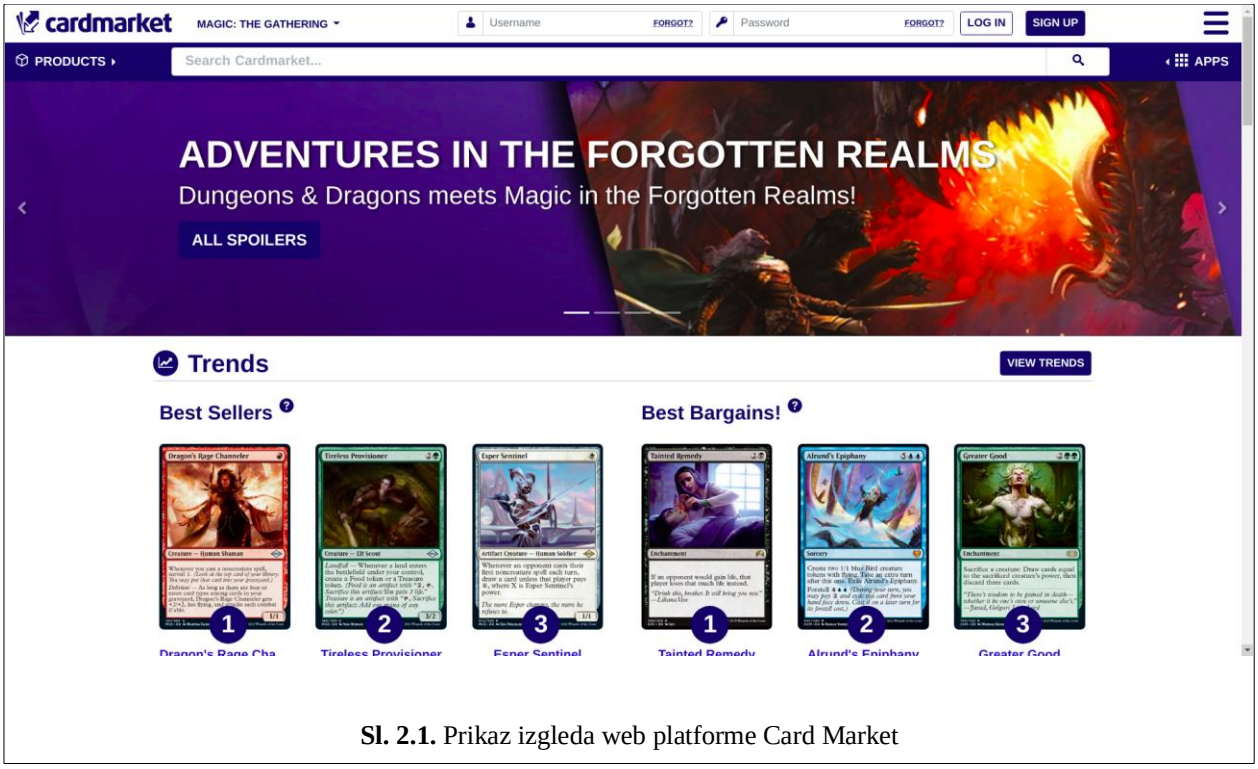
Objasniti pojam TCG. Navesti funkcionalnosti koje bi trebala imati web platforma za korisnike TCG zajednice. Izraditi web platformu za korisnike TCG zajednice. Predvidjeti različite uloge na platformi: administrator i korisnik (funkcije korisnika su prodavatelj ili kupac artikala, i ocjenjivanje prodavatelja predmeta). Korisnik u ulozi prodavatelja treba imati mogućnost postavljanja predmeta na prodaju, ali i mogućnost sudjelovanja u kupovini artikala koje ne nudi. Kupac treba imati mogućnost postavljanja ponuda za pojedine predmete. Administratorske ovlasti trebaju uključivati akcije privremene, polutrajne ili trajne zabrane pristupa platformi.

2. POSTOJEĆA RJEŠENJA

Danas postoje mnoga rješenja za kupovanja, razmjene i aukcije karata, no većinom se ta rješenja međusobno razlikuju. Neka imaju izgled i funkcionalnosti klasičnih web trgovina gdje korisnik željene karte stavlja u košaricu te ih nakon toga kupuje. Druga imaju izgled starih foruma na kojima se preko komentara vrše aukcije i prodaja ili su pak dio grupe na društvenim mrežama gdje se preko objave sadržaja i komentiranja vrše spomenute akcije. U idućim potpoglavljima opisana su i prikazana neka od postojećih rješenja.

2.1. Card Market

Card Market, Sl. 2.1, je web aplikacija koja registriranim korisnicima omogućuje pretragu, sortiranje i kupnju karata.[1] Isto tako nudi mogućnost postavljanja karata na prodaju. Karte se postavljaju pojedinačno te im se dodaje količina dostupna za prodaju. Web aplikacija nudi i mogućnost ocjene i prijave prodavača u slučaju neprimjerenih poruka ili slanja krivih pošiljki te nudi povrat novca.

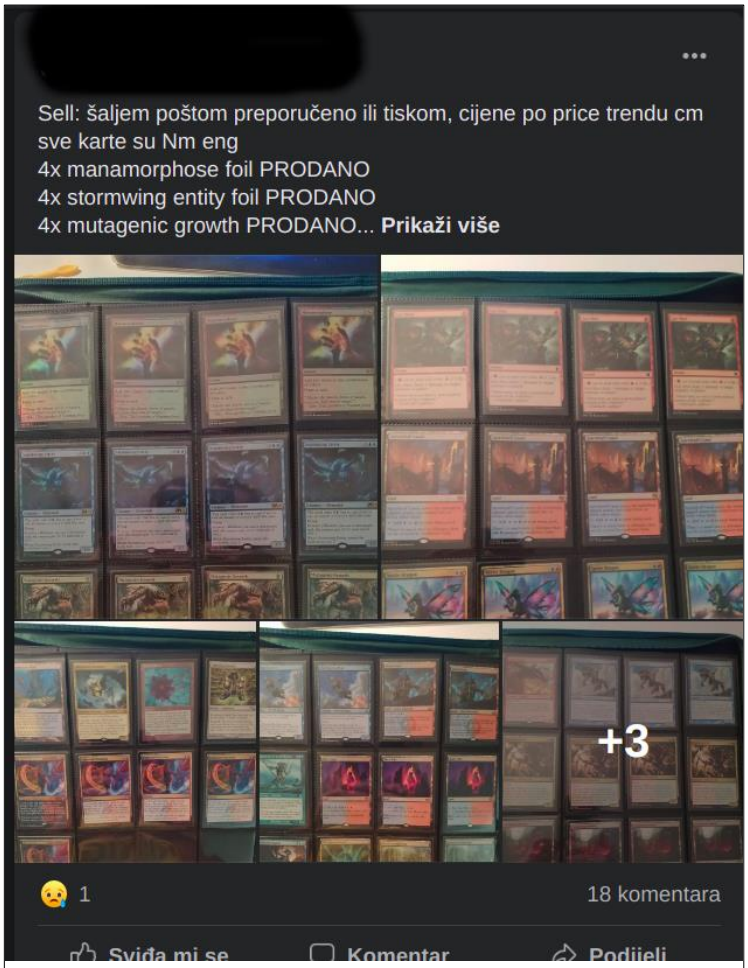


Sl. 2.1. Prikaz izgleda web platforme Card Market

2.2. Facebook Grupe

Facebook, kao društvena mreža nudi stvaranje grupa s obzirom na to kakav će se sadržaj objavljivati.[2] Jedna takva grupa nudi svojim članovima stvaranje objava o prodaji, potražnji ili

aukciji što je prikazano na Sl 4.2. Svaki član grupe može komentirati objave bez obzira zanima li ga sadržaj ili ne.



Sl. 2.2. Prikaz objave prodaje na Facebooku

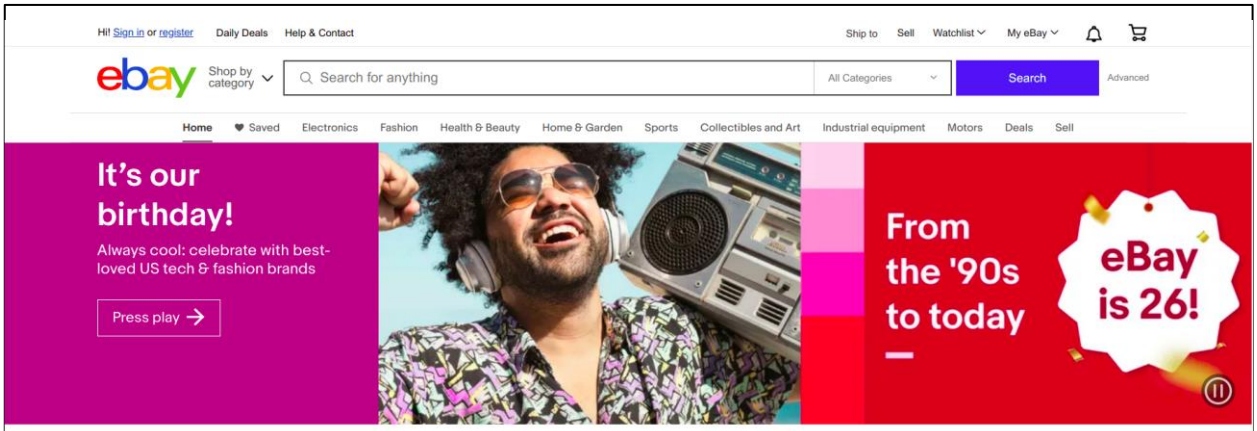
2.3. CroMTG Forum

Na ovom forumu dostupne su objave vezane za kartašku igru *Magic: The Gathering.*[3] Uz objave o turnirima nalaze se i objave o prodaji, kupnji, aukciji i razmjeni na kojima korisnici u obliku odgovora na objavu stavljaju svoje ponude, komentare i slično. Sl. 2.3. prikazuje izgled stranice CROMTG forum.

Sl. 2.3. Prikaz sekcije trgovine na CROMTG forumu

2.4. Ebay

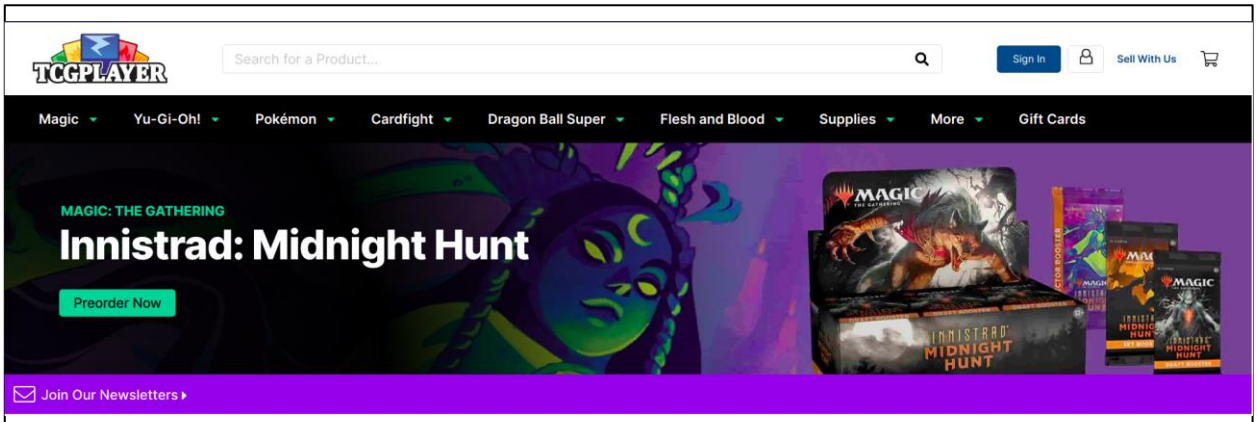
Ebay kao web trgovina nudi raznolike proizvode među kojima se nalaze i karte za kartaške igre.[4] Ova web trgovina omogućuje pretraživanje i kupnju proizvoda te sudjelovanje u aukcijama koje registrirani korisnici mogu stvoriti za proizvod koji prodaju. Izgled početne stranice prikazan je na Sl. 2.4.



Sl. 2.4. Prikaz početne stranice web trgovine ebay

2.5. TCGplayer

TCGplayer je web platforma koja registriranim korisnicima nudi pretragu i kupnju proizvoda vezanih za kartaške igre. Isto tako nudi opciju kupnje pojedinačnih karata.[5] Ova stranica je primarno namijenjena za američko tržište, no korisnici iz drugih zemalja mogu kupovati proizvode uz nešto skuplje cijene dostave. Izgled početne stranice prikazan je na Sl. 2.5.



Sl. 2.5. Prikaz početne stranice web trgovine TCGplayer

3. KORIŠTENE TEHNOLOGIJE PRI RAZVOJU WEB PLATFORME

Kako bi web platforma bila funkcionalna potrebno je izabrati pravilne alate, odnosno tehnologije, kojima će se izraditi. Za izradu pojedinih dijelova web platforme odabrani su određene tehnologije koje pružaju lakšu i bržu izradu iste. U idućim potpoglavljima opisane su odabrane tehnologije.

3.1. Node.js

Node.js je okruženje otvorenog koda koje se koriste za pokretanje JavaScript kôd izvan internetskih pretraživača.[6] Ovim je omogućeno pisanje poslužiteljskog dijela web platforme u skriptnom programskom jeziku JavaScript, ali isto tako klijentske strane čiji se sadržaj stvara na poslužiteljskoj strani.

Node.js danas nudi velik broj radnih okvira za izradu poslužiteljskih i klijentskih dijelova web platformi. Za izradu ovog rada korištena su radna okruženja Adonis.js, za izradu poslužiteljskog dijela, te Nuxt.js za izradu klijentskog dijela.

3.2. Adonis.js

Adonis.js je Node.js radni okvir koji je inspiriran Laravelom te koristi MVC (engl. *Model View Controller*) arhitekturu.[7][8] Adonis.js koristi se za izradu poslužiteljskog dijela web aplikacija te korisnicima omogućuje lako sučelje za stvaranje datoteka modela i kontrolera. Isto tako Adonis.js pruža jednostavnu izradu migracijskih datoteka koja služe za brže i lakše stvaranje tablica baze podataka. Za izradu ovog rada korištena je verzija 5 Adonis.js koja koristi TypeScript kao programski jezik. TypeScript je nad set nad programskim jezikom JavaScript te omogućuje lakšu validaciju podataka jer za razliku od JavaScripta koristi predefinirane tipove podataka.

3.3. Nuxt.js

Nuxt.js je radni okvir korišten pri izradi klijentskog dijela web aplikacija čiji se sadržaj stvara na poslužiteljskoj strani.[9] Baziran je na Vue.js.[10] Omogućuje korisnicima laku konfiguraciju te svojom izvedbom pojednostavljuje obradu asinkronih zahtjeva, stvaranje među sloja za provjeru podatak te brzo i lako usmjeravanje po komponentama pomoću ugrađenog usmjerivača (engl. *router*). Kako je baziran na Vue.js prikaz elemenata korisničkog sučelja pisan je u HTML-u, dok se za izgled i stiliziranje elementa koristi CSS. Kako bi komunikacija sa poslužiteljem bila moguća koristi se JavaScript.

3.4. Vuetify

Vueitfy je radni okvir radni okvir za izradu komponenti korisničkog sučelja.[11] Pruža velik broj komponenti spremnih za korištenje te se može poistovjetiti sa *Boostrap*-om napravljenim za korištenje unutar radnih okvira koji su izgrađeni na Vue.jsu.

3.5. Sokcket.IO

Socket.IO je biblioteka pisana u JavaScriptu koja omogućuje komunikacija između klijenta i poslužitelja u stvarnom vremenu.[12] Komunikacija između klijenta i poslužitelja odvija se dvosmjerno što znači da se za svaku promjenu na poslužiteljskoj strani obavještava klijentska strana i obratno. Sastoji se od klijentskog dijela, odnosno od dijela koji se pokreće unutar pretraživača te od poslužiteljskog dijela koji se poziva unutar poslužitelja nastalih na Node.js radnom okviru. Iako se koriste dva dijela, poslužiteljski i klijentski, biblioteke pružaju gotovo identičan aplikacijski sloj te je na taj način omogućena laka implementacija oba dijela.

3.6. PostgreSQL

PostgreSQL je sustav otvorenog kôda za stvaranje i upravljanje relacijskim bazama podataka.[13] Baziran je na SQLu te mu je zbog toga sintaksa jezika za definiciju podataka i jezika za manipulaciju podatcima slična jezicima kao što su MySQL i MariaDB.

3.7. Postman

Postman je klijent za aplikacijsko programsko sučelje koji omogućuje lakše testiranje poslužiteljske strane web aplikacija i platformi.[14] Omogućuje lako slanje zahtijeva za kreiranje, izmjenu, brisanje i dohvaćanje, odnosno CRUD zahtijeva (engl. *Create Read Update Delete*). Za prijenos podataka primarno koristi JSON format, ali isto tako omogućuje slanje zahtjeva u obliku formi, teksta, JavaScripta i slično.

4. RAZVOJ RJEŠENJA

U ovom poglavlju opisan je način korištenja tehnologija spomenutih u prijašnjem poglavlju te njihova interakcija u razvoju rješenja web platforme. Prvo potpoglavlje opisuje razvoju poslužiteljskog dijela platforme te interakciju s bazom podataka. U drugom potpoglavlju opisan je razvoj klijentskog dijela platforme te komunikaciju sa poslužiteljskim dijelom. U trećem potpoglavlju opisan je način implementacije socketa[1] na poslužiteljskoj i klijentskoj strani aplikacije. Četvrto potpoglavlje opisuje način testiranja poslužiteljskog i klijentskog dijela aplikacije.

4.1. Poslužiteljski Dio Rješenja

Kako bi bio moguć razvoj poslužiteljskog dijela rješenja prvo je potrebno stvoriti projekt. Kako bi se projekt mogao stvoriti potrebno je imati instaliran Node.js. Nakon instalacije Node.js-a te unosom komandne linije *npm init adonis-ts-app@latest <ime-aplikcije>* stvara se početni Adonis.js projekt.

4.1.1. Migracije

Jedan od najbitnijih dijelova web platforme je pohrana potaka. Kako bi se podatci mogli pohraniti i biti trajno spremljeni potrebno je stvoriti migracijske datoteke kako bi definirali izgled tablica. Samo stvaranje migracijske datoteke neće automatski omogućiti spajanje poslužitelja sa bazom te je prije pokretanja potrebno instalirati biblioteku koja se brine o spajanju na bazu te ju inicijalizirati, a rad sa PostgreSQL bazom podataka ostvaruje se na način da se unutar komandne linije projekta pozovu naredbe *npm i @adonisjs/lucid* i *node ace configure @adonisjs/lucid.* Nakon konfiguracije poslužitelja za rad sa PostgreSQL bazom podataka stvara se nova migracijska datoteka unosom nardbe *node ace migration:run <ime-migracije>*. Stvara se datoteka unutar koje se piše kôd kojim se definira izgled tablice, kao što je prikazano na isječku kôda 4.1. Izgled podataka spremljenih u PostgreSQL bazu prikazan je na Sl. 4.1.

1 Socket je jedna krajnja točka u dvosmjernoj komunikaciji između dva programa koji se izvode na mreži.

```
import BaseSchema from '@ioc:Adonis/Lucid/Schema'

export default class Users extends BaseSchema {
  protected tableName = 'users'

  public async up () {
    this.schema.createTable(this.tableName, callback: (table : CreateTableBuilder ) => {
      table.increments( columnName: 'id')

      /**
       * Uses timestampz for PostgreSQL and DATETIME2 for MSSQL
       */
      table.string( columnName: 'username', length: 80).unique().notNullable()
      table.string( columnName: 'email').unique().notNullable()
      table.string( columnName: 'first_name').notNullable()
      table.string( columnName: 'last_name').notNullable()
      table.string( columnName: 'password').notNullable()
      table.timestamp( columnName: 'created_at', options: { useTz: true })
      table.timestamp( columnName: 'updated_at', options: { useTz: true })
    })
  }

  public async down () {
    this.schema.dropTable(this.tableName)
  }
}
```

Isječak kôda 4.1 Prikaz sadržaja migracijske datoteke

	...	1 username	email	first_name	last_name	password	created_at
1		1 msTheMan	abc@abc.com	Ivan	Sertic	$argon2id$v=19$t=3,m=4096,p=1$wE2s2e5cBxOftnbtUuDJ…	2021-06-06 16
2		3 usain	abc1@abc.com	Marko	Sertic	$argon2id$v=19$t=3,m=4096,p=1$mGWawedHFhQl1KgaEtCm…	2021-06-06 16
3		4 isTheMan	is@gmail.com	Ivan	Sertić	$argon2id$v=19$t=3,m=4096,p=1$zhz5tDglmcXBtzeqgSBY…	2021-07-04 19
4		5 isTheMan2	is2@gmail.com	Ivan	Sertić	$argon2id$v=19$t=3,m=4096,p=1$xemlPlEpEZ8HVH4/H3q5…	2021-07-04 19

Sl. 4.1. Prikaz tablice PostgreSQLa

4.1.2. Modeli

Kako Adonis.js koristi već spomenutu MVC arhitekturu, nakon stvaranja migracijskih datoteka za tablice koje će spremati podatke za korisnike, objave, medijski sadržaj, ponude na aukcijama te zabrane za pojedine korisnike potrebno je stvoriti modele po uzoru na stvorene tablice. Modeli su stvoreni unosom naredbe *node ace make:model <ime-modela>*. Nakon stvaranja datoteke modela potrebno je definirati sve atribute koji predstavljaju stupce unutar tablice te ukoliko su neke tablice međusobno povezane potrebno je stvoriti relacije između modela

koje odgovaraju relacijama unutar tablica, kao što je prikazano na isječku kôda 4.2. Za označavanje relacija i atributa kao predstavnike stupaca, Adonis.js koristi dekoratere *@column() i @<vrsta-relacije>()*.

```
export default class Ban extends BaseModel {
  @column( options: { isPrimary: true })
  public id: number

  @column( options: {columnName:'user_id',serializeAs:'userId'})
  public userId: number

  @column.dateTime( options: {columnName:'banned_until',serializeAs:'bannedUntil'})
  public bannedUntil: DateTime

  @column( options: {columnName:'is_perma_banned',serializeAs:'isPermaBanned'})
  public isPermaBanned: boolean

  @column.dateTime( options: { autoCreate: true })
  public createdAt: DateTime

  @column.dateTime( options: { autoCreate: true, autoUpdate: true })
  public updatedAt: DateTime

  @belongsTo( model: ()=>User, options: {
    localKey:'id',
    foreignKey:'userId'
  })
  public user: BelongsTo<typeof User>
```

Isječak kôda 4.2. Prikaz sadržaja datoteke modela

4.1.3. Kontroleri i rute

Poslovni dio logike web platforme pisan je unutar datoteka kontrolera. Kontroleri su stvoreni unosom naredbe *node ace make:controller <ime-kontrolera>*. Unutar kontrolera pisana je logika za stvaranje, izmjenu, brisanje i dohvaćanje podataka. Primjer je pokazan isječkom kôda 4.3. Za svaki model stvoren je odgovarajući kontroler. Kako kontroleri sami po sebi ne bi imali smisla jer ih klijentska strana ne može samo tako pozvati potrebno je za svaki kontroler definirati rutu te na tu rutu kao rukovoditelja postaviti željenu metodu kontrolera, kao što je prikazano na isječku kôda 4.4.

```
export default class AdminsController {
  async getReports({request,response}){

    const queryParams: any = request.only(['limit','page'])

    const paginationHelper: Pagination = new Pagination()
    const paginationData:any = await paginationHelper.validateData(queryParams)

    if(paginationData.hasOwnProperty( v: 'messages')){
      return response.badRequest(paginationData.messages)
    }

    const reports: ModelPaginatorContract<Report> = await Report.query()
      .whereDoesntHave( relation: 'reportedUser', callback: (query : ... )=>{
        query.whereHas('ban',(_query)=>{})
      })
      .preload( relation: 'reportedUser')
      .preload( relation: 'reportMaker')
      .preload( relation: 'post')
      .paginate( page: paginationData.page ? paginationData.page : 1 , perPage: paginationData.limit ? paginationData.limit: 10)
    reports.namingStrategy = Report.getNamingStrategy()

    return response.ok(reports)
  }
```

Isječak kôda 4.3. Prikaz sadržaja datoteke kontrolera

```
import Route from "@ioc:Adonis/Core/Route";

Route.group( callback: ()=>{
  Route.get( pattern: 'reports', handler: 'AdminsController.getReports')
  Route.post( pattern: 'ban/user/:userId', handler: 'AdminsController.banUser')
  Route.patch( pattern: 'ban/:banId/user/:userId', handler: 'AdminsController.editBan')
  Route.delete( pattern: 'ban/:banId', handler: 'AdminsController.deleteBan')
  Route.get( pattern: 'ban/:banId', handler: 'AdminsController.getSingleBan')
  Route.get( pattern: 'ban', handler: 'AdminsController.getBans')
}).prefix( prefix: 'api/admin/').middleware( middleware: ['isAdmin','getUser'])
```

Isječak kôda 4.4. Prikaz sadržaja datoteke rute

4.1.4. Verifikacija korisnika i među sloj

Kako bi neregistriranim korisnicima pristup web platformi bio onemogućen korišten je JWT (eng. *Json Web Token*).[10] Prilikom prijave korisnika pomoću para privatnog i javnog ključa generira se JWT pomoću kojeg se može autentificirati korisnik. JWT postavlja u zaglavlje zahtjeva koji se postavlja na poslužitelja, kako bi se izbjegao redundantan kod provjere valjanosti tokena, stvara se među sloj koji je postavljen na rute koje je potrebno zaštiti. Među sloj verificira i čita podatke predane preko tokena te dohvaća korisnika iz baze podataka ukoliko korisnik sa predanim podatcima postoji, što je prikazano isječkom kôda 4.5. Ukoliko je token nevaljan ili korisnik ne postoji zahtjev se odbija te korisnik dobije poruku o grešci, kako je prikazano na isječku kôda 4.6.

```
class Jwt {

  private static instance: Jwt

  private constructor() {
  }

  public static init() {
    if (!this.instance) {
      this.instance = new Jwt()
    }
    return this.instance
  }

  public async getUserData(authHeader: string): Promise<any> {
    try {
      const token: string = authHeader.split( separator: ' ')[1]
      const jwtData = await jwt.verify(token,Env.get( key: 'APP_KEY'))

      const user: User = await User.findOrFail(jwtData.uid)

      return user
    } catch (e) {
      return {
        message: "Invalid token",
        code: 'error.invalidToken'
      }
    }
  }
}
```

Isječak kôda 4.5. Prikaz sadržaja datoteke za provjeru valjanosti tokena

```
import JwtHelper from 'App/Helpers/Jwt'
import User from "App/Models/User";
export default class GetUser {
  public async handle (ctx, next: () => Promise<void>) {
    // code for middleware goes here. ABOVE THE NEXT CALL
    const data:any = await JwtHelper.getUserData(ctx.request.headers().authorization)

    if(data instanceof User){
      ctx.user = data
    }else{
      return ctx.response.badRequest(data)
    }

    await next()

  }
}
```

Isječak kôda 4.6. Prikaz sadržaja datoteke među sloja

4.2. Klijentski dio rješenja

Za klijentski dio rješenja stvoren je projekt unosom naredbe *npm create nuxt-app <ime-projekta>* unutar komandne linije. Tijekom stvaranja interaktivnim izbornikom dodana je biblioteka za korištenje Vue.js-a. Klijentska strana sastoji se od stranica za prikaz ukupnog sadržaja te čistog i predefiniranog rasporeda (engl. *layout*).

Nuxt.js koristi dinamički usmjerivač kojim generira putanje do određenih stranica preko strukture direktorija unutar samog projekta. Isto tako nudi mogućnost podešavanja osnovnog dijela ruta za komunikaciju sa poslužiteljem pomoću *nuxt.js* datoteke.

4.2.1. Izrada rasporeda

Raspored se koristi kako bi prikazali komponente koje će određene stranice dijeliti. Tako se u datoteci predefiniranog rasporeda čiji je kôd prikazan na isječku kôda 4.7., stvaraju elementi koji će se nalaziti na većini stvorenih stranica. Isto tako čisti raspored koristi se prilikom registracije i prijave jer na tim stranicama navigacijska traka ne bi trebala biti vidljiva.

```
<template>
  <v-app dark>
    <v-navigation-drawer
      v-model="drawer"
      :mini-variant="miniVariant"
      :clipped="clipped"
      fixed
      clipped
    >
      <v-list>
        <v-list-item
          v-for="(item, i) in items"
          :key="i"
          :to="item.to"
          router
          exact
          active-class="navigationActiveItem"
          @click="changeAlreadyClicked(item)"
          v-show="item.visible"
        >
          <v-list-item-action>
            <v-icon :disabled="item.alreadyClicked">{{ item.icon }}</v-icon>
          </v-list-item-action>
```

Isječak koda 4.7. Prikaz sadržaja socket.ts datoteke

4.2.2. Izrada izgleda i funkcionalnosti stranica

Svaka stranica zadužena je za određene funkcionalnosti platforme, prikaz, stvaranje, izmjena ili brisanje. Svaka stranica koristi različite elemente stvorene pomoću Vuetify-a te sadrži dio skripte koji je zadužen za pozivanje poslužiteljske strane i manipulaciju podatcima za prikazivanje. Unutar *template* oznaka pisan je kod koji je zadužen za prikaz elemenata te popunjavanje elemenata podatcima. Kako bi elementi mogli pristupiti i prikazati podatke unutar oznaka *script* pisane su metode, isječak kôda 4.8., koje manipuliraju podatcima ili pozivaju poslužitelja za dohvaćanje ili stvaranje novih podataka.

```
methods:{
  async login(){
    try{
      let res = await this.$axios.post( url: 'auth/login',this.user)
      jsCookie.set('token',res.data.data.token)
      jsCookie.set('check',res.data.data.isAdmin)
      jsCookie.set('id',res.data.data.id)
      await this.$router.push("posts")
    }catch (err) {
      console.log(err.response)
      if(err.response.data.data.hasOwnProperty( v: 'message')){
        this.$toast.error(err.response.data.data.message)
      }else{
        this.$toast.error("Incorrect username or password!")
      }
    }
  },
```

Isječak kôda 4.8. Prikaz sadržaja unutar script oznaka

Kako bi svaka stranica mogla ispravno komunicirati sa poslužiteljem koristi se usmjerivač *axios* koji je zadužen za pozivanje prethodno definiranih ruta na poslužitelju. Isto tako koristi predefinirani usmjerivač kojim se stranice međusobno povezuju, odnosno omogućuje se navigacija između stranica.

4.3. Implementacija socketa na klijentskoj i poslužiteljskoj strani

Kako je već ranije spomenutu za komunikaciju u stvarnom vremenu između poslužiteljske i klijentske strane koristi se biblioteka Socket.IO. Primarna uloga ove biblioteke na ovoj web platformi je omogućiti korisnicima postavljanje ponuda te dobivanje obavijesti u stvarnom vremenu.

Na poslužiteljskoj i na klijentskoj strani postavljaju se događaji koji osluškuju ili primaju određene događaje koje jedna ili druga strana odašilje. Klijentska i poslužiteljska strana koriste istu sintaksu za postizanje ove funkcionalnosti stoga se implementacija značajno ne razlikuje već dolazi do razlike samo u tome što se na poslužiteljskoj strani događaji definiraju unutar *socket.ts* datoteke, a na klijentskoj unutar oznaka *script.* Na isječku kôda 4.9. prikazan je način implementacije socketa na poslužiteljskoj strani.

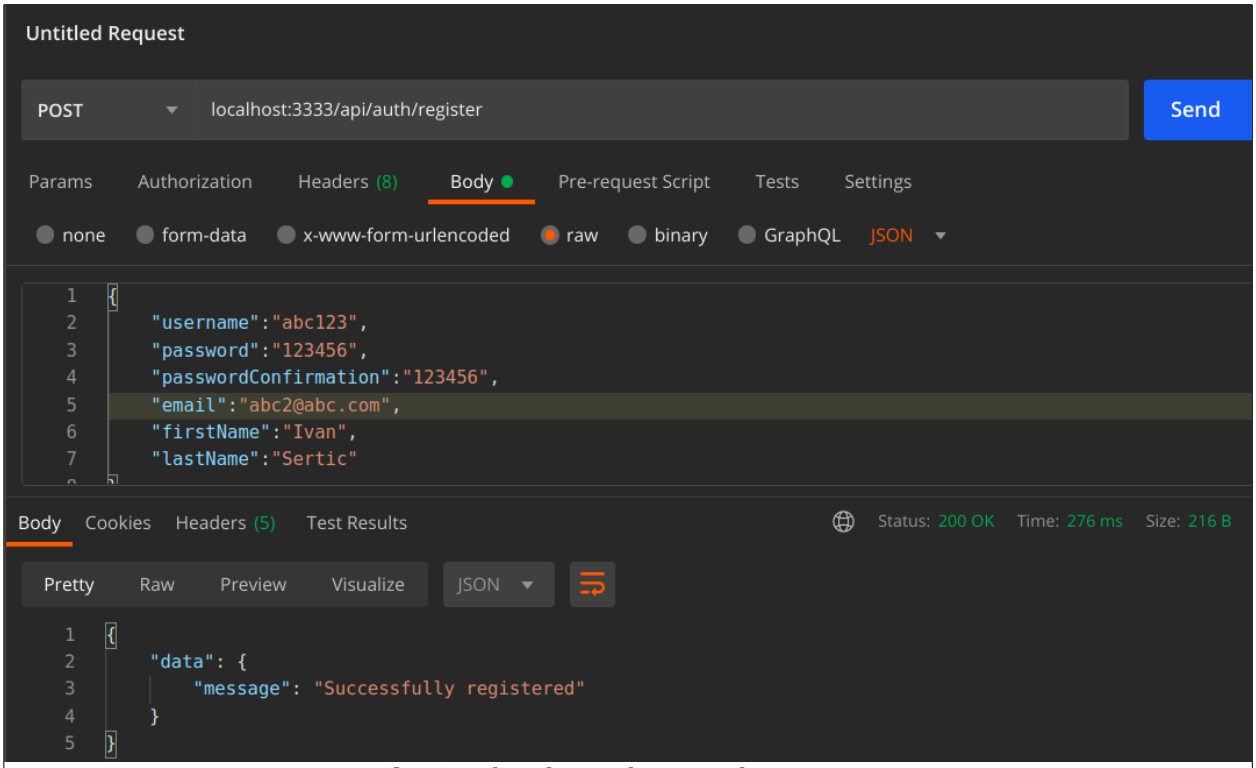
```
Ws.io.on( ev: 'connection',  listener: (socket) => {
  socket.join(`user:${socket.data.userId}`)

  socket.on('bid',async (data,callback)=>{

    const post: Post|null = await Post
      .query()
      .where('postType','auction')
      .where('id',data.postId)
      .preload( relation: 'highestBid')
      .first()

    if(!post){
      return
    }

    const user: User = await User.findOrFail(socket.data.userId)

    let bid: Bid
    if(!post.highestBid){
      bid = await Bid.create( values: {
        amount:post.startingBid,
        userId: socket.data.userId,
        postId: post.id
      })
    }else{
```

Isječak kôda 4.9. Prikaz sadržaja *socket.ts* datoteke

4.4. Testiranje

Kako bi se potvrdila ispravnost rada platforme potrebno je testirati poslužiteljski i klijentski dio. Klijentski dio testiran je metodom bijele kutije. Pošto su promjene na klijentskoj strani odmah vidljive testiranje je bilo moguće bez dodatnih alata. Najvažniji dijelovi klijentske strane su usmjeravanje sa stranice na stranicu i prikaz podataka. Kako je prikaz podataka usko vezna sa poslužiteljskom stranom prije izrade samog klijentskog dijela potrebno je pobrinuti se da poslužiteljska strana radi ispravno.

Poslužiteljska strana prva je rađena bilo je potrebno pobrinuti se da sve rute i njihovi rukovoditelji ispravno rade, kako bi bilo lakše izraditi klijentsku stranu i spojiti je sa poslužiteljskom stranom, korišten je alat *Postman* čije je sučelje prikazano na Sl. 4.2. Ovaj alat nudi lako i brzo testiranje aplikacijskog programskog sučelja na način da se unutar okvira za rad unese ruta definirana na poslužiteljskoj strani, tip rute koji može biti *Post, Delete, Put, Patch* ili *Get,* te tijelo zahtijeva ako je potrebno za obradu zahtijeva. Ukoliko je rad rute ispravan poslužitelj će vratiti podatke ili poruku u JSON formatu. Ukoliko je došlo do greške na poslužiteljskoj strani Adonis.js vraća rezultat pripadajući status greške te detaljan uvid gdje je točno greška nastala kako bi otklanjanje greške bili brže i jednostavnije.



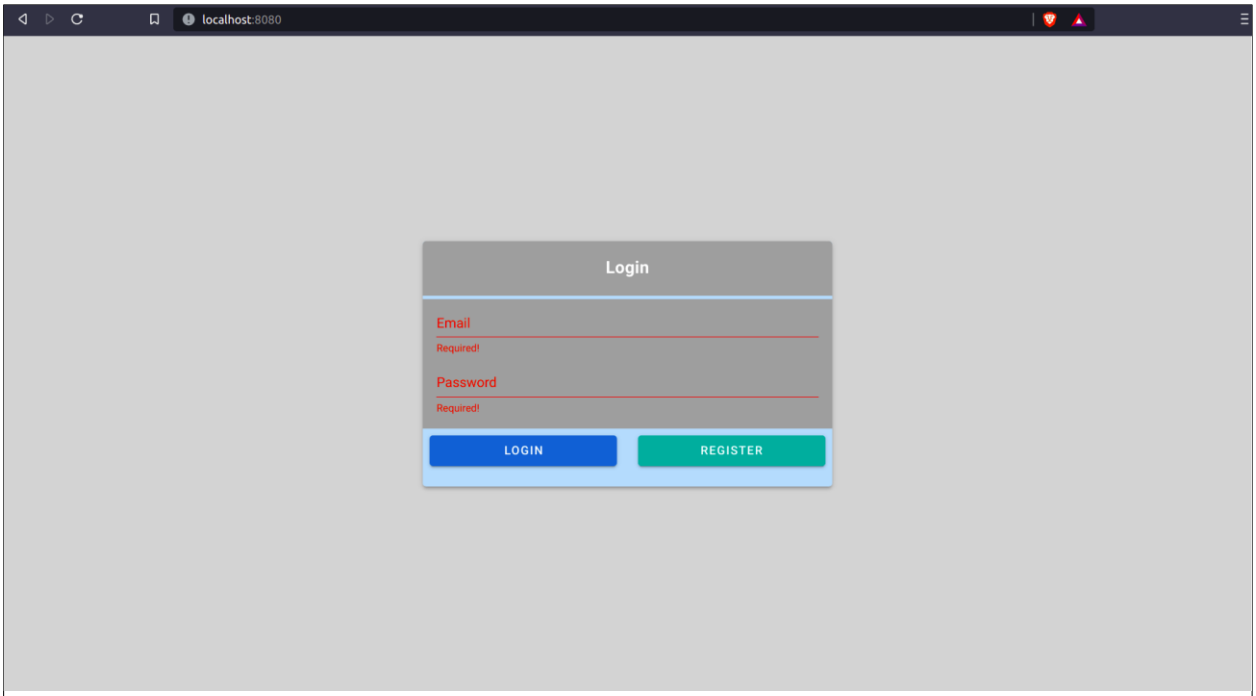
Sl. 4.2. Prikaz slanja zahtijeva u alatu Postman

5. PRIKAZ I KORIŠTENJE WEB PLATFORME

U ovom poglavlju prikazan je izgleda web platforme spremne za korištenje te način korištenja iste. Prvo potpoglavlje opisuje način prijave i registracije u sustav te kako je validacija izvedena sa klijentske strane. Drugo potpoglavlje prikazuje izgled i funkcionalnosti web platforme sa stajališta običnog korisnika. Treće potpoglavlje opisuje funkcionalnosti koje web platforma nudi svojim administratorima.

5.1. Prijava i registracija

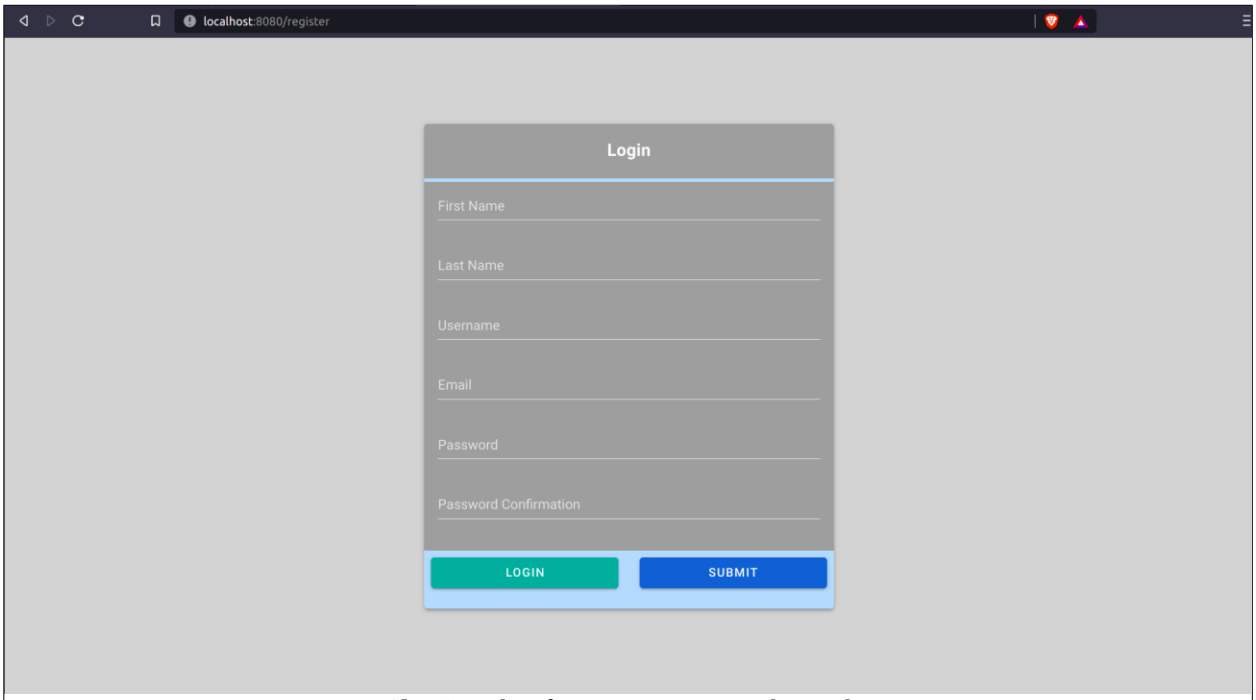
Otvaranjem web platforme korisniku se prikazuje forma za prijavu u sustav. Izgled forme za prijavu prikazan je na Sl. 5.1. Ukoliko korisnik ima već stvoren račun unosom ispravne e-mail adrese i svoje zaporke te pritiskom na „*Login*" gumb prijavljuje se u sustav.



Sl. 5.1. Prikaz forme za prjavu

Ukoliko korisnik ne unese potrebne podatke ili e-mail nije ispravnog oblika prikazuje mu se poruka o grešci na onom polju gdje je greška nastala. Validacija podataka formi tako radi na cijeloj platformi.

Za korisnike koji nemaju stvorene račune, klikom na gumb „*Register*" otvara se stranica sa formom za stvaranje računa prikazana na Sl. 5.2. Nakon unosa podataka i klikom na gumb „*Submit*", ukoliko su podatci zadovoljili validaciju, stvara se korisnički račun i korisnik se preusmjerava na stranicu na kojoj se nalazi forma za prijavu u sustav.

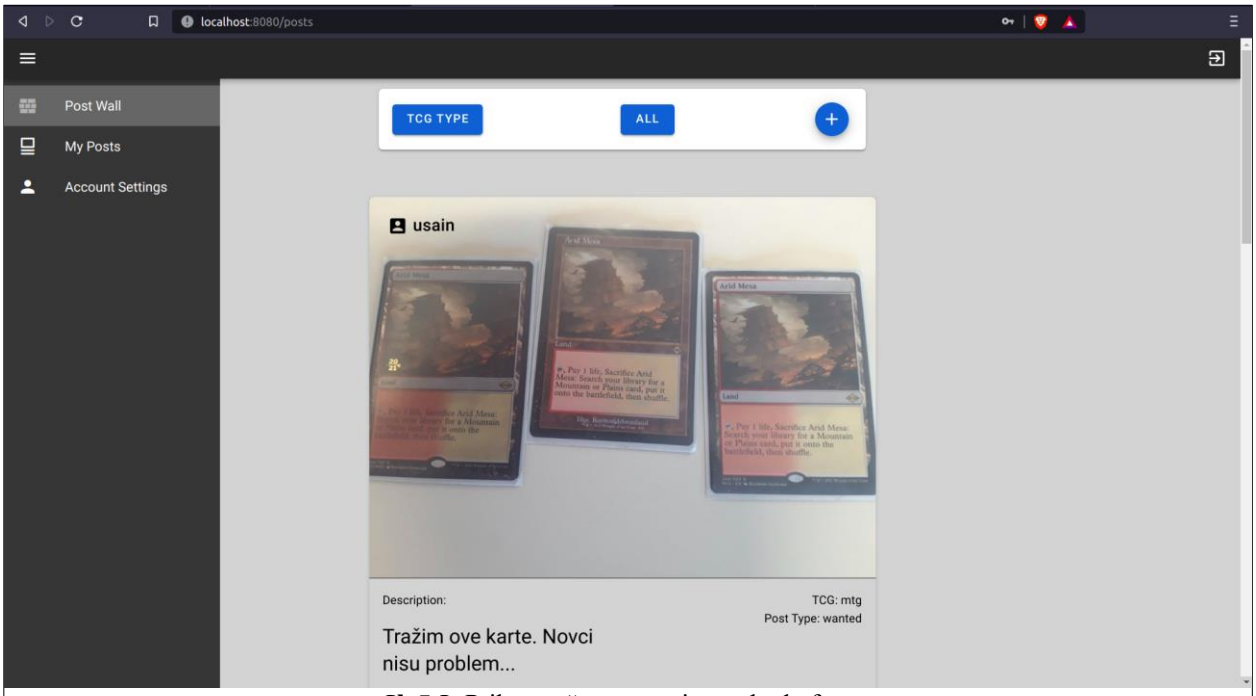


Sl. 5.2. Prikaz forme za registraciju korisnika

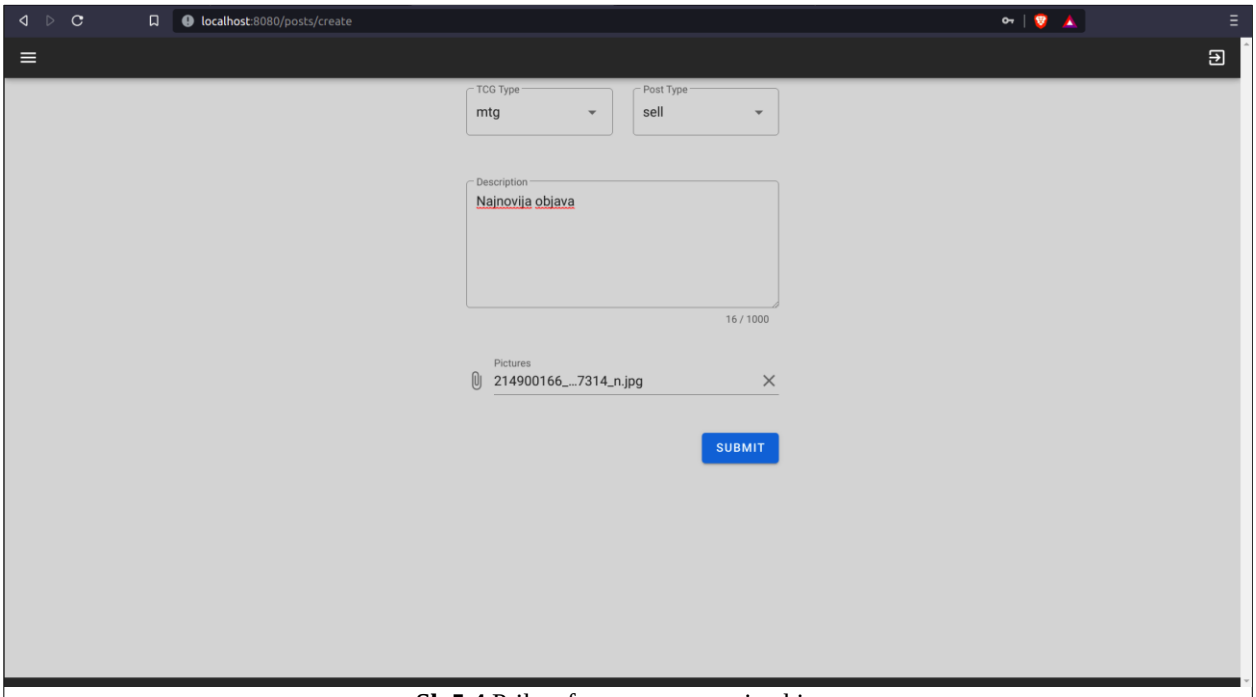
5.2. Glavne funkcionalnosti web platforme za korisnike

Prijavom u sustav korisnik se preusmjerava na stranicu na kojoj se nalaze sve objave svih korisnika. Izgled početne stranice prikazan je na Sl. 5.3. Stvara se predefinirani izgled stranice. Na aplikacijskoj traci nalazi se gumb na kojemu korisnik može otvoriti izbornik sa lijeve strane pomoću kojeg se može navigirati kroz aplikaciju, te gumb za izlaz. Iznad svih učitanih objava nalazi se alatna traka na kojoj korisnici mogu filtrirati objave po tipu kartaške igri, tipu objave ili mogu dodati svoju objavu.

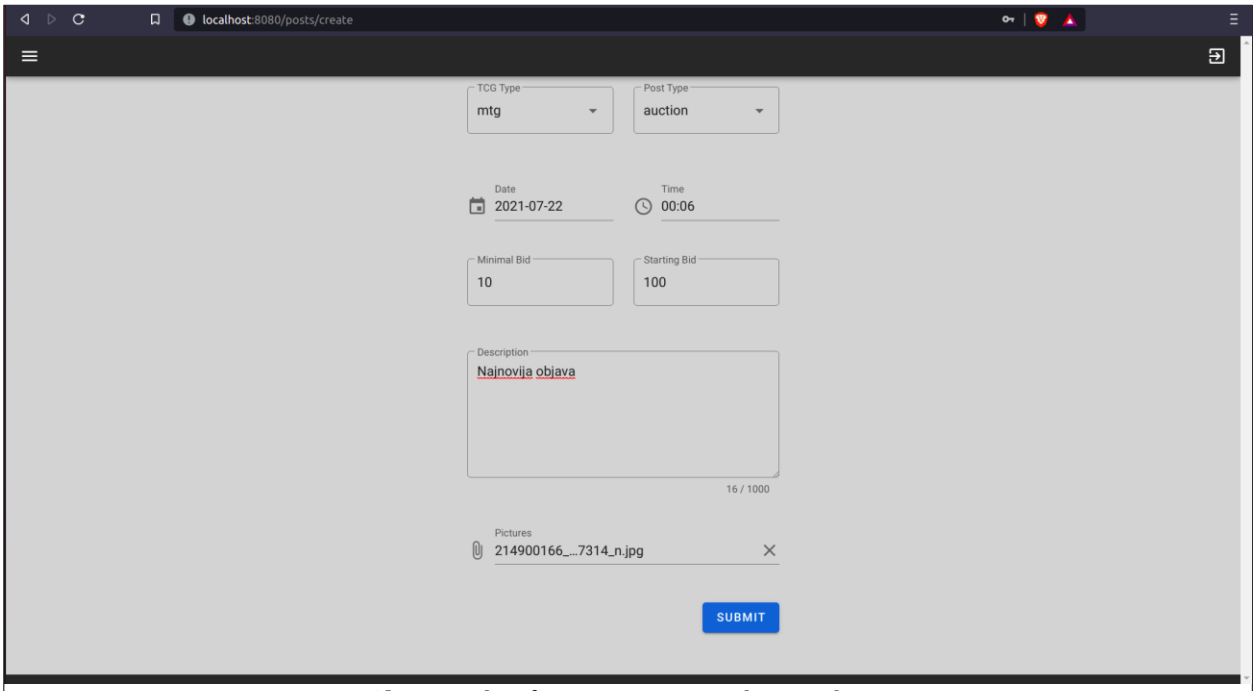
Pritiskom na gumb sa znakom plusa korisniku se otvara forma za stvaranje novih objava prikazana na Sl. 5.4. Podatci forme ovise o tome je li objava aukcija ili ne. Ukoliko je objava aukcija potrebno je unijeti dodatne informacije kao što su kraj aukcije, datum i vrijeme, početnu ponudu i minimalnu ponudu što je prikazano Sl. 5.5. Za stvaranje objave u oba slučaja korisnik je obavezan postaviti minimalno jednu sliku.



Sl. 5.3. Prikaz početne stranice web platforme

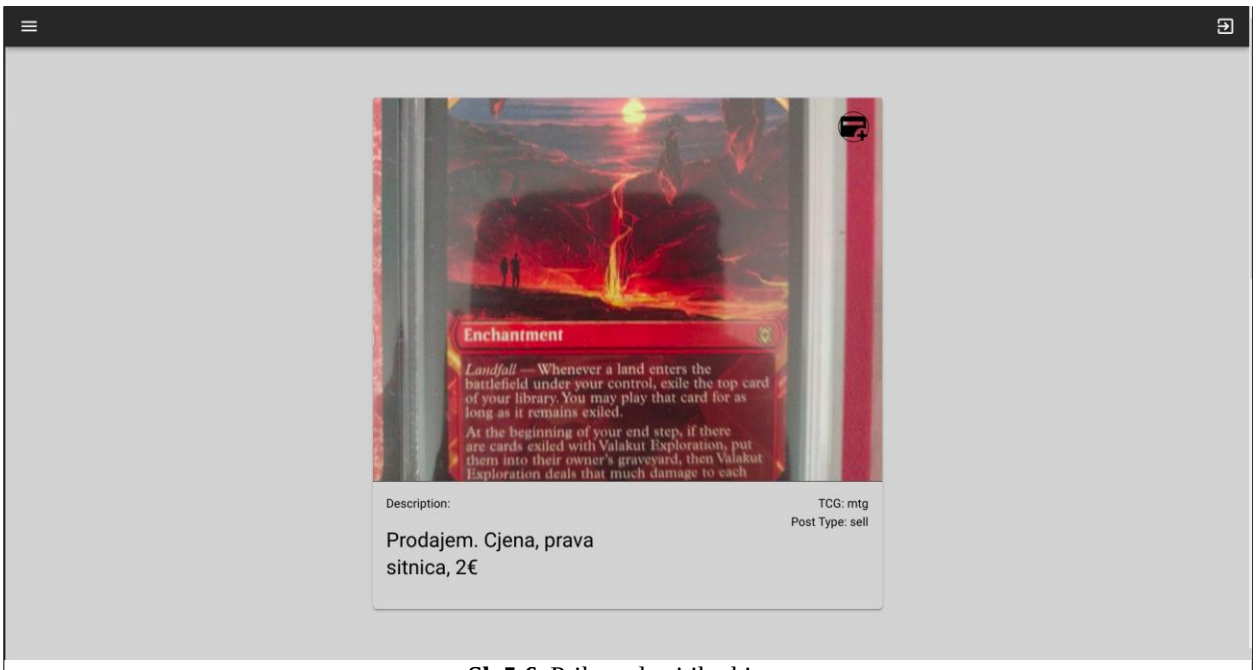


Sl. 5.4 Prikaz forme za stvaranje objave

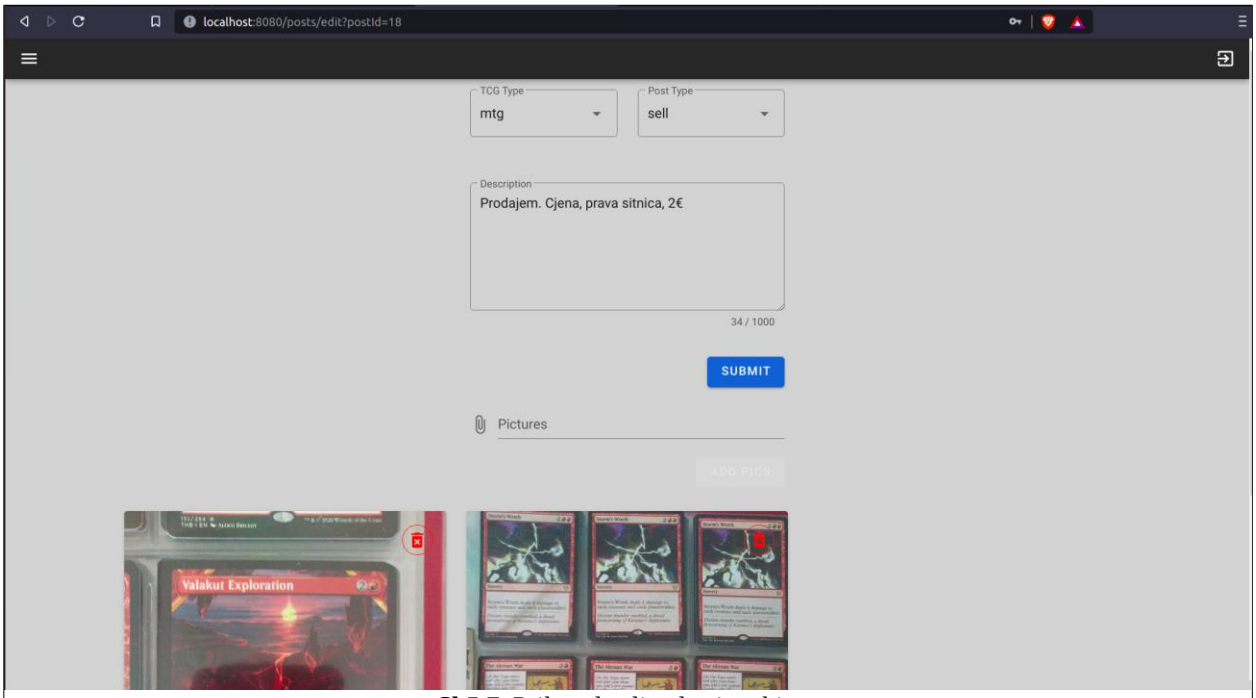


Sl. 5.5. Prikaz forme za stvaranje objave aukcije

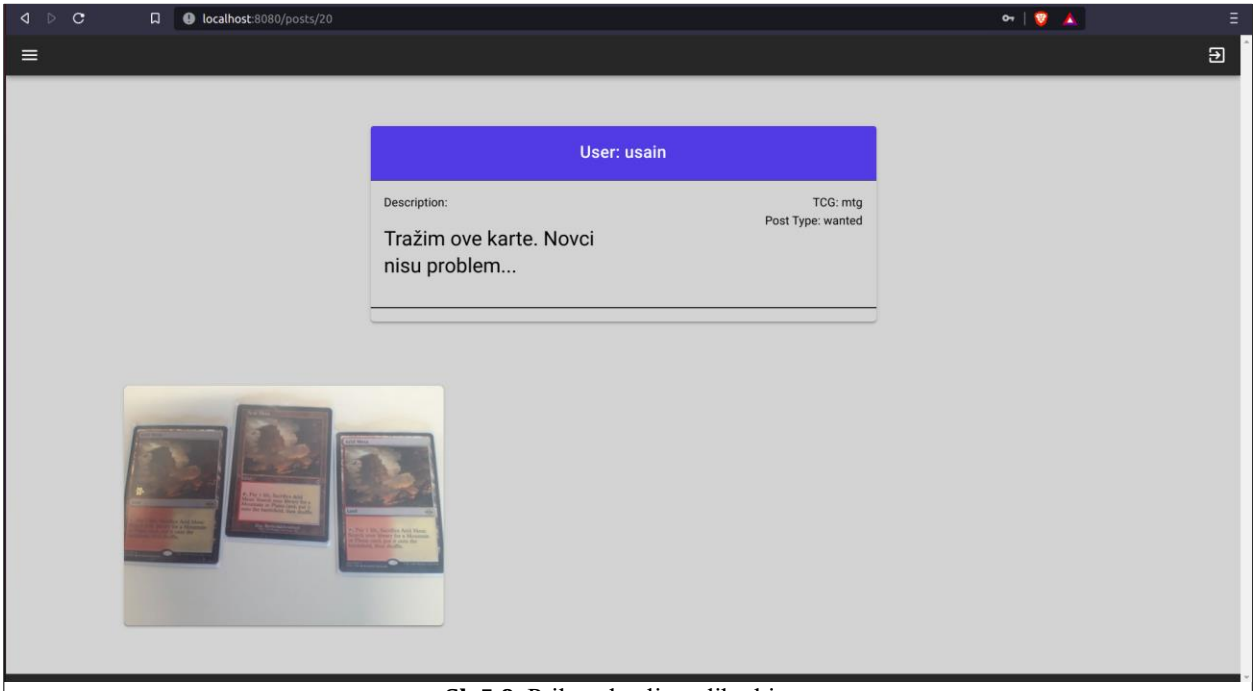
Ukoliko objava ima više slika samo će jedna biti postavljena za prikaz na glavnoj stranici, Sl. 5.6., dok se ostale slike mogu naći na detaljima objave, Sl. 5.8., koji se otvaraju klikom na sliku objave na naslovnoj stranici. Isto tako ukoliko se pritiskom na sliku otvori vlastita objava korisnik se preusmjerava na stranicu za uređivanje objave koja je prikazana slikom 5.7. Zbog lakšeg uređivanja objava korisnik do svojih objava može doći odabirom „*My Posts*" iz izbornika prikazanog na Sl. 5.3. te na gumb sa znakom plusa otvoriti istu stranicu za uređivanje objava.



Sl. 5.6. Prikaz vlastitih objava

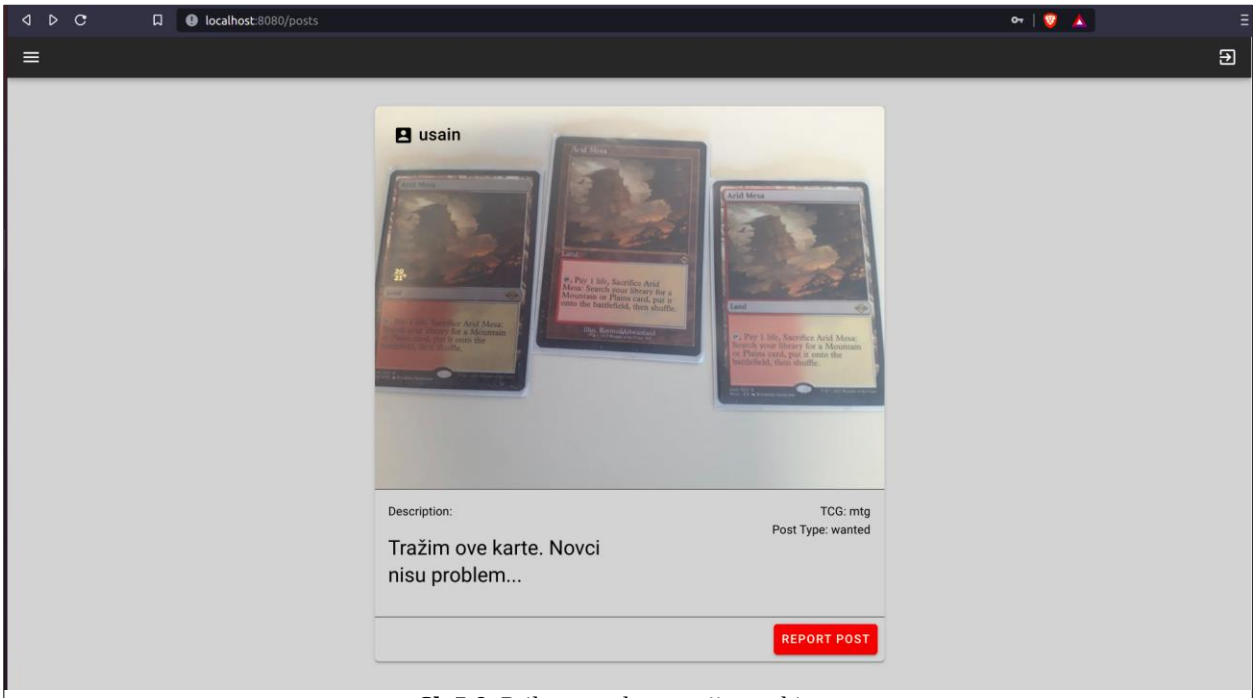


Sl 5.7. Prikaz detalja vlastite objave

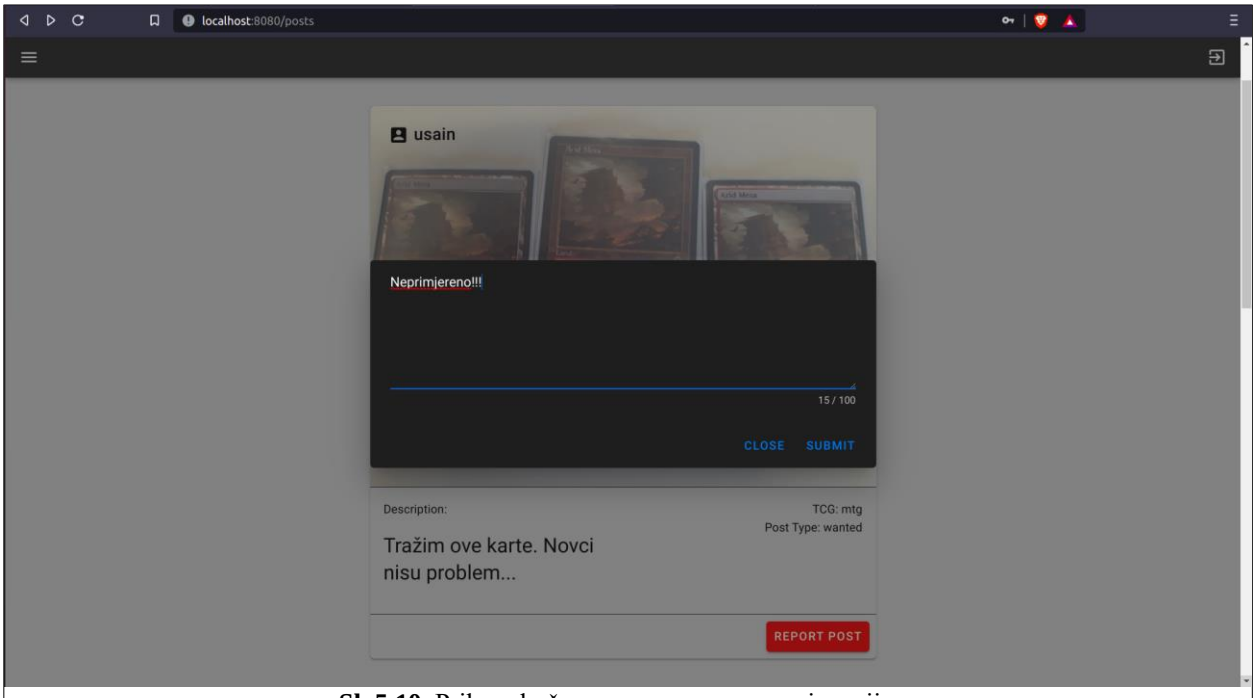


Sl. 5.8. Prikaz detalja tuđih objava

Na naslovnoj strani na svim objavama osim vlastitih, nalazi se gumb za prijavu objave kao što je prikazano Sl. 5.9. Pritiskom na gumb „*Report Post*" otvara se skočni prozor, Sl. 5.10., u kojemu se unosi opis zašto korisnik želi prijaviti objavu.

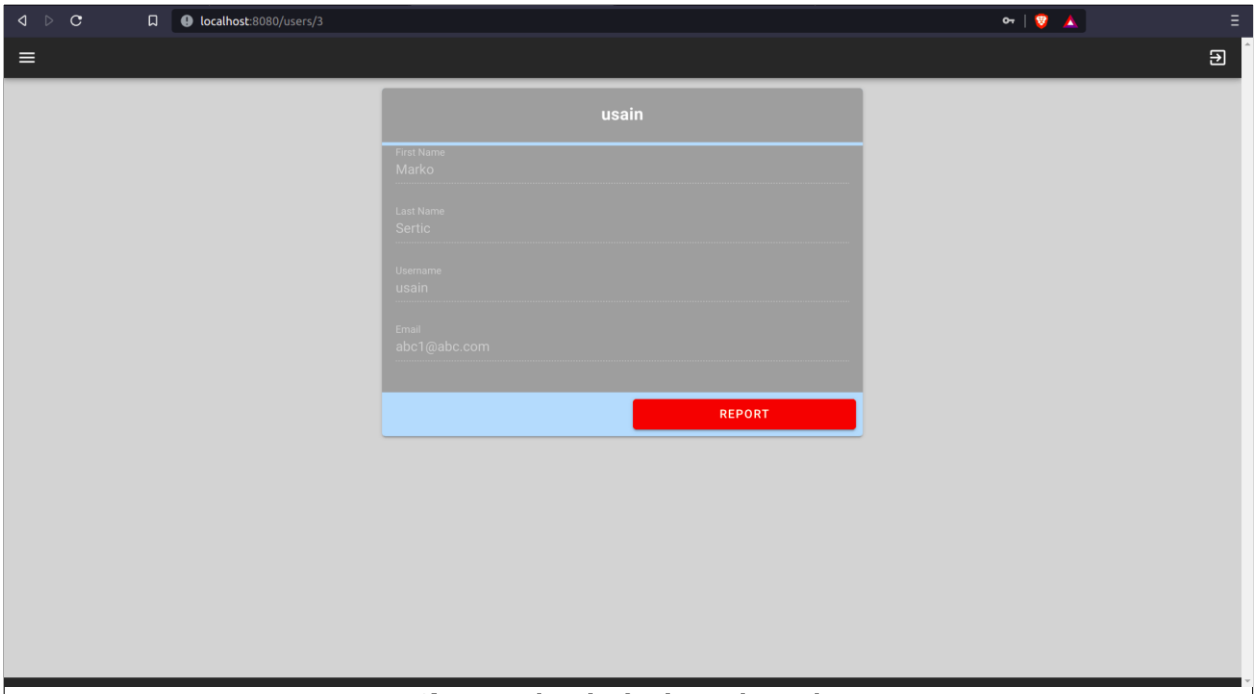


Sl. 5.9. Prikaz gumba za prijavu objave



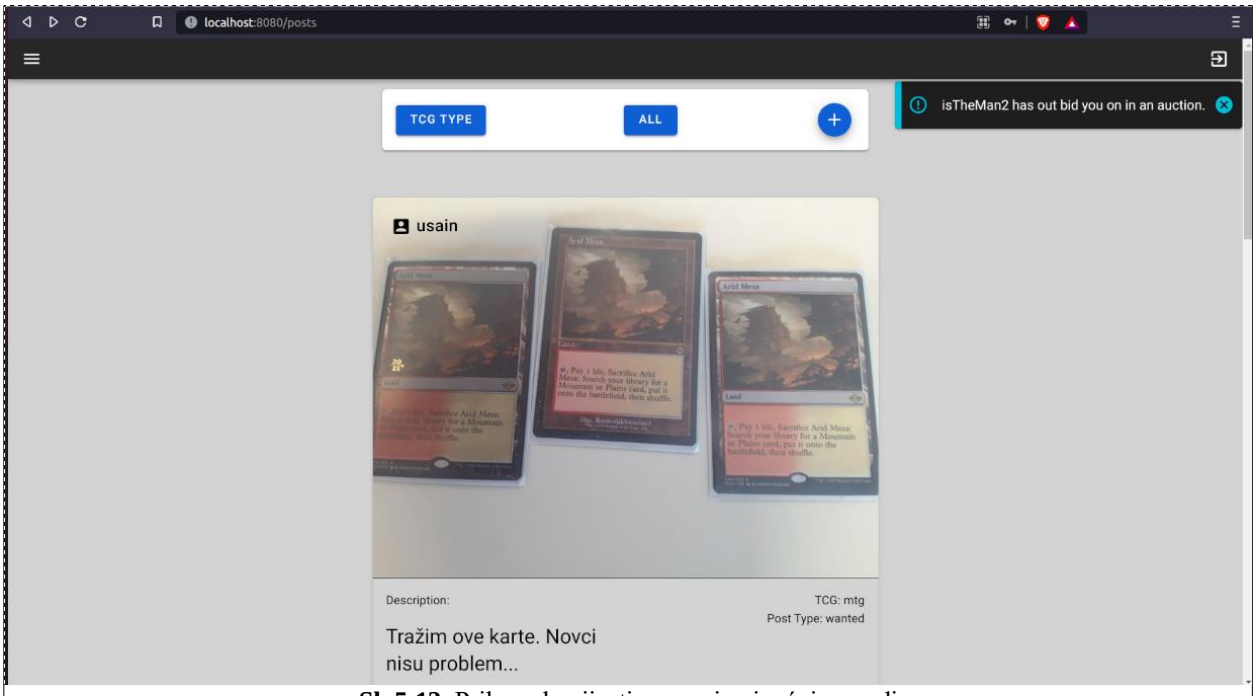
Sl. 5.10. Prikaz skočnog prozora za unos opisa prijave

Na svim objavama u gornjem desnom kutu nalazi se korisničko ime korisnika koji je stvorio objavu. Lijevo od korisničkog imena nalazi se ikona koja vodi na detalje korisnika. Na detaljima su vidljivi osobni podatci korisnike te se nudi prijava korisnika kao na Sl. 5.11. Klikom na gumb „*Report*" pojavljuje se skočni prozor kao na Sl.5.10.



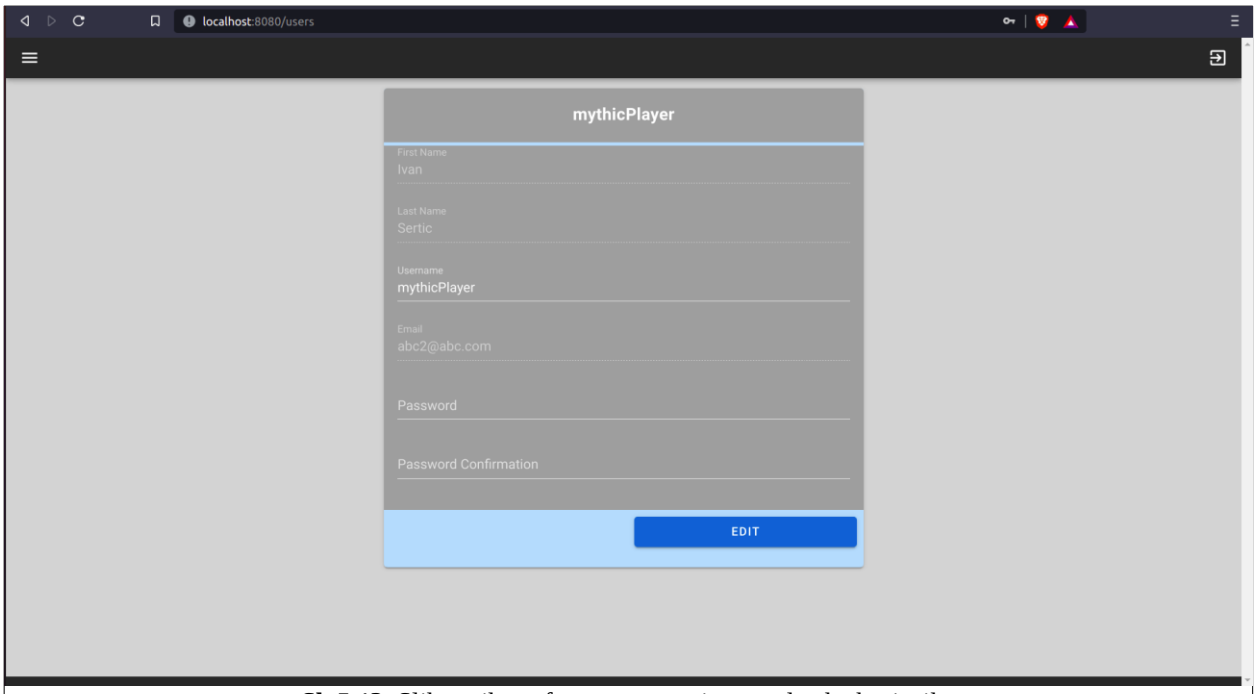
Sl 5.11. Prikaz detalja drugog korisnika

Ukoliko je objava nekog korisnika tipa aukcija drugi korisnici imaju pravo postavljanja ponuda. Kad korisnik postavi novu podatci objave se automatski ažuriraju te je sada njegova ponuda i njegovo korisničko ime predstavljeni kao korisnika sa najvećom ponudom. Kako bi aukcije bila poštena svim se korisnicima, koji su do sad postavili barem jednu ponudu, preko socketa šalje obavijest o tome tko je novi korisnik koji je ponudio najveću ponudu kao na Sl. 5.12. Klikom na obavijest korisnik se preusmjerava na detalje posta na kojemu je postavio ponudu.



Sl. 5.12. Prikaz obavijesti o novoj najvećoj ponudi

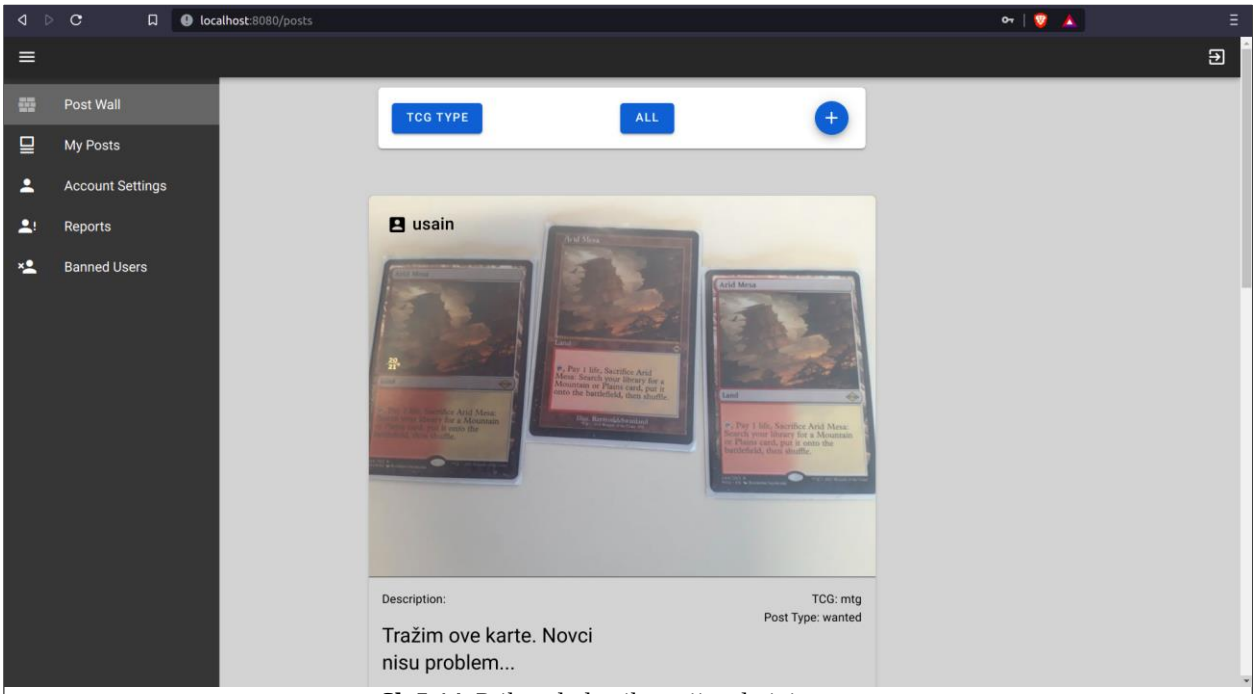
Svaki korisnik može promijeniti svoje podatke klikom na „*Account Setting*”. Otvara se stranica na kojoj se nalazi forma, Sl. 5.13., te nakon unosa klikom na gumb „*Submit*” mijenjaju se podatci korisnika.



Sl. 5.13. Slika prikaza forme za promjenu podataka korisnika

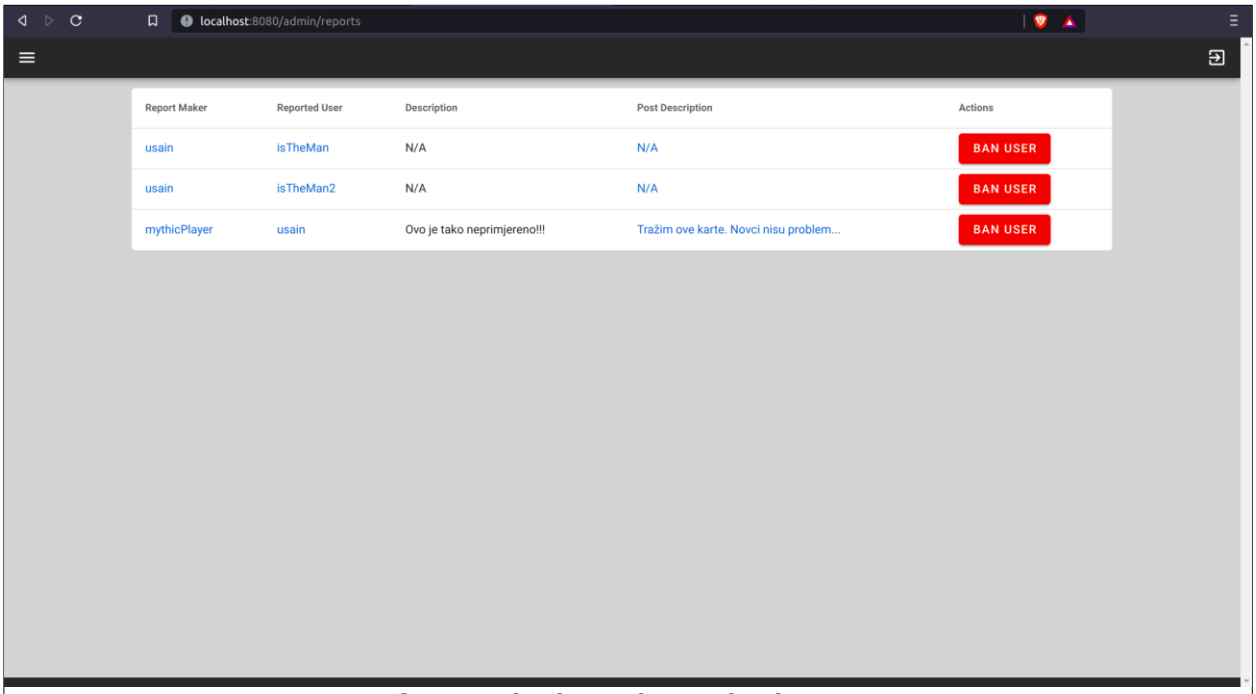
5.3. Prikaz funkcionalnosti administratora platforme

Uz sve mogućnosti korisnika administratori imaju dvije važne mogućnosti. Mogu pregledati sve prijave koje su korisnici postavili te isto tako mogu dodijeliti zabrane određenim korisnicima za koje smatraju da su postavljali ne primjeren sadržaj ili se nisu pridržavali željenog ponašanja na platformi. U izborniku s lijeve strane pojavljuju se dodatne dvije opcije „*Reports*” i „*Banned Users*” kao na Sl. 5.14.

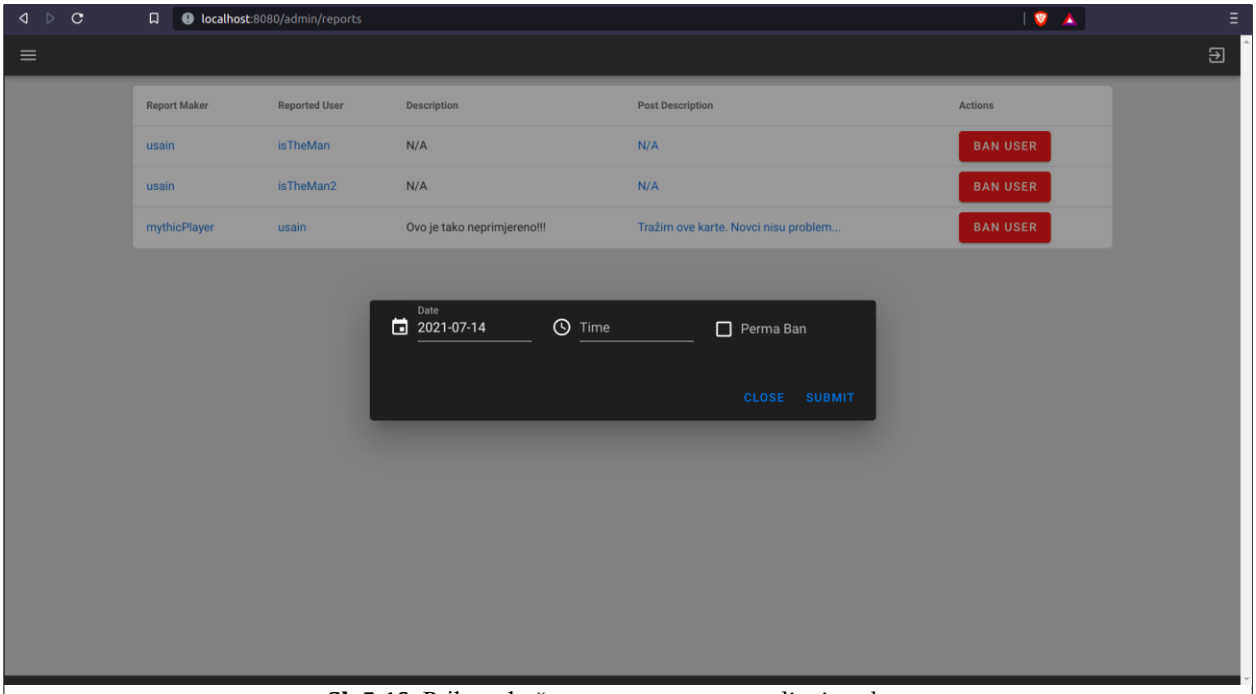


Sl. 5.14. Prikaz dodatnih opcija administratora

Klikom na „*Reports*" pojavljuje se lista svih postavljenih pritužbi s njihovim opisima prikazano na Sl. 5.15. Klikom na imena korisnika otvaraju se detalji profila tih korisnika. Ukoliko je pritužba postavljena na objavu korisnika klikom na opis objave otvara se prijavljena objava. Ovdje administratori imaju mogućnost postaviti zabrane prijavljenim korisnicima. Klikom na gumb „*Ban User*" otvara se skočni prozor u kojemu se definira vrijeme zabrane ili se korisniku pristup zabranjuje do daljnjeg, prikazano na Sl. 5.16., odnosno dok administratori ne odluče vratiti mu pristup platformi.

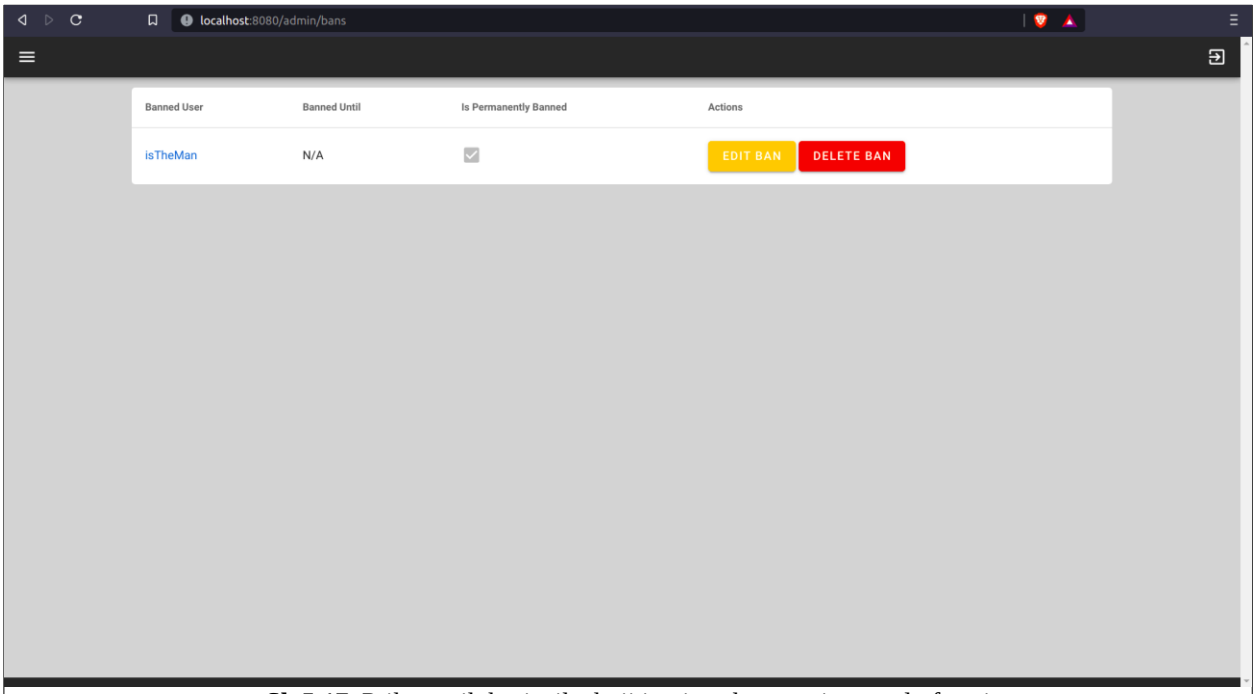


Sl. 5.15. Prikaz liste svih postavljenih prijava



Sl. 5.16. Prikaz skočnog prozora za postavljanje zabrane

Nakon postavljanja zabrane korisniku klikom na „*Baned Users*” unutar izbornika administrator vidi sve korisnike kojima je postavljena zabrana kao što je prikazano na Sl. 5.17. Ovdje administrator može ili ukloniti zabranu u potpunosti ili može urediti zabranu te promijeniti trajanje zabrane.



Sl. 5.17. Prikaz svih korisnika koji imaju zabrane pristupa platformi

6. ZAKLJUČAK

Izradom ovog diplomskog rada pokazalo se kako se uz pravilan odabir tehnologija za izradu web platformi mnogi aspekti naših života, bili oni za razonodu ili posao, mogu poboljšati i ubrzati. Na ovoj web platformi omogućeno je jednostavnije postavljanje objava za prodaju, potražnju ili razmjenu karata te se poboljšalo iskustvo aukcija koje su nudile dosadašnje aplikacije i platforme.

Za izradu ovog diplomskog rada bilo je, ponajprije, potrebno uložiti mnogo vremena na proučavanje i traženje tehnologija za izradu koje bi zajedno omogućile brzu i jednostavnu izradu ovakve web platforme. Izradom web platforme stečena su mnoga teoretska znanje i praktična iskustva te se primjenom jednih i drugih pokazalo lako i efikasno izgraditi ovakvu platformu.

Testiranjem je osiguran isprivan način rada ove web platforme te se pokazalo kako je potrebno uložiti gotovo podjednako vremena u testiranje kao i u razvoj jer je svaku funkcionalnost bilo potrebno ponajprije testirate prije nego su se nove funkcionalnosti razvijale.

LITERATURA

[1] Card Market, dostupno na: https://www.cardmarket.com/en [datum posjete: srpanj 2021.]

[2] Facebook, dostupno na: https://www.facebook.com/ [datum posjete: srpanj 2021.]

[3] CroMTG forum, dostupno na: https://cromtg.forumcroatian.com/ [datum posjete: srpanj 2021.]

[4] Ebay, dostupno na: https://www.ebay.com/ [datum posjete: srpanj 2021.]

[5] TCGplayer, dostupno na: https://www.tcgplayer.com/ [datum posjete: srpanj 2021.]

[6] Node.js, dostupno na: https://nodejs.org/en/ [datum posjete: srpanj 2021.]

[7] Adonis.js, dostupno na: https://adonisjs.com/ [datum posjete: srpanj 2021.]

[8] MVC, dostupno na: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm#:~:text=The Model-View-Controller (,development aspects of an application. [datum posjete: srpanj 2021.]

[9] Nuxt.js, dostupno na: https://nuxtjs.org/ [datum posjete: srpanj 2021.]

[10] Vue.js, dostupno na: https://vuejs.org/ [datum posjete: srpanj 2021.]

[11] Vuetify, dostupno na: https://vuetifyjs.com/en/ [datum posjete: srpanj 2021.]

[12] Socket.IO, dostupno na: https://socket.io/ [datum posjete: srpanj 2021.]

[13] PostgreSQL, dostupno na: https://www.postgresql.org/ [datum posjete: srpanj 2021.]

[14] Postman, dostupno na: https://www.postman.com/ [datum posjete: srpanj 2021.]

[15] JSON Web Token: dostupno na: https://jwt.io/ [datum posjete: srpanj 2021.]

SAŽETAK

U ovom diplomskom radu izrađena je web platforma za korisnike TCG zajednice. Korisnicima je omogućeno postavljanje objava za prodaju, razmjenu, potražnju i aukciju karata. Isto tako opisane su tehnologije koje su korištene pri izradi ove web platforme.

Web platforma je izrađena kako bi poboljšala korisničko iskustvo dosadašnjih platformi koje nude slične mogućnosti. Za kontrolu korisnika implementiran je sustav administratora.

Ključne riječi: Adonis.js, Nuxt.js, platforma, Socket.IO, TCG, web platforma

ABSTRACT

Web Platform for TCG Community

In this master thesis a web platform for users of a TCG community was developed. Users can create posts for selling, buying, trading and auctioning cards. Also, the technologies which are used for creating this web platform are described.

The web platform was developed to enhance user experience of using similar platforms with similar features. User control functionality is implemented in administrator panel.

Keywords: Adonis.js, Nuxt.js, platform, Socket.IO, TCG, web platform