TEHNIČKA ŠKOLA DARUVAR

ZAVRŠNI RAD

TSD TV – izrada web aplikacije na Linux poslužitelju

Alexander Gustovarac

RAZRED:

4.RT

ZANIMANJE:

Tehničar za računalstvo

MENTORICA

Ivana Milić, mag. inf.

DARUVAR, svibanj 2024.

ZAHVALE

Zahvaljujem se ***Tehničkoj školi Daruvar*** na pruženoj mogućnosti i prilici da za svoj završni rad razvijem softver za školski televizor.

SADRŽAJ

1 Uvod

TSD TV je web aplikacija izgrađena na *Node.js* sustavu, s namjerom da omogući jednostavniju projekciju vizualnog sadržaja na televizorima Tehničke škole Daruvar. Svrha ove aplikacije je pružiti njenim korisnicima to jest školskom osoblju brz i jednostavan pristup sadržaju vezanom uz školu i školska događanja, uz jednostavnu administraciju istih. Web aplikacija tako omogućuje brzo i jednostavno postavljanje i uređivanje vizualnog sadržaja pružajući jedinstveno i jednostavno sučelje za upravljanje. Nova aplikacija ima za glavni cilj riješiti probleme koji su postojali u dosadašnjoj verziji aplikacije, a to se uglavnom odnosi na kompleksno postavljanje vizualnog sadržaja na televizor. Nova verzija aplikacije pruža cjelovito i jedinstveno rješenje za sve dosadašnje probleme, a uzevši u obzir sveobuhvatnost ovakvog rješenja, imao sam priliku u ovom zadatku okušati se kao *full-stack* web developer. Web aplikacija u svom *back-end* dijelu sadrži aplikacijsko programsko sučelje (engl. API) za komunikaciju s bazom podataka koju ima, dok s druge strane pruža korisniku preko računalne mreže i grafičko sučelje za upravljanje tj. *front-end* dio. Web aplikacija namijenjena je školskom osoblju Tehničke škole Daruvar sa svrhom prikazivanja raznih vizualnih sadržaja na televizoru i davanja informacija o trenutnim događanjima u školi kao što su promjene u rasporedu, trenutno vrijeme i sl. Uz sve to web aplikacija dizajnirana je pristup s bilo kojeg računala spojenog na mrežu škole, bez preuzimanja dodatnog softvera na računalu kojim želimo pristupiti web aplikaciji. Drugim riječima, Internet preglednik nam je dovoljan da bismo imali pristup radu s ovom aplikaciji. Web aplikacija zahtijeva redovito ažuriranje novim slikama i novim informacijama.

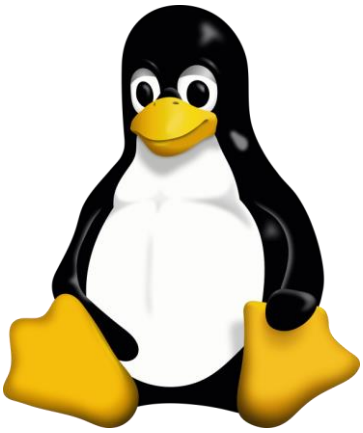
2 Teoretska podloga rada

2.1 Raspberry Pi 4

Raspberry Pi je mikroračunalo namijenjeno da bude lako dostupno svima. Malih je dimenzija, gotovo veličine bankovne kartice, a cjenovno vrlo pristupačno. Stvorila ga je Raspberry Pi fondacija s ciljem poboljšanja i lakšeg učenja početnicima u računalstvu, no svakako se može i primijeniti u profesionalne svrhe. Svakom novom verzijom hardver postaje sve snažniji, a trenutni procesor je Broadcomov SoC koji sadrži ARM-ove jezgre procesora koje također možemo pronaći i u pametnih telefona. Najvažniji dio njegovog hardvera su svakako GPIO iglice koje mu donose i praktičnu važnost, jer pomoću njih može kontrolirati različite uređaje. Kako je Raspberry Pi zapravo računalo, potreban mu je i operacijski sustav. Posebno u njegovom slučaju, potreban mu je operacijski sustav na čijem se izvornom kodu mogu raditi modifikacije i dodavati nove stvari. Operacijski sustav takvog tipa je Linux i njegove distribucije, a najpopularnija Linuxova distribucija za Raspberry Pi je Raspberry Pi OS koji dolazi u paketu s mnoštvom dodatnih mogućnosti. Iako je prvobitno stvoren za podučavanje računalstva u školama, ubrzo dobiva novu funkciju u robotici i sličnim područjima. Raspberry Pi također dobiva funkciju malog računalnog servera za pružanje jednostavnih, ali i složenih aplikacija poput ove te može biti savršeno rješenje kao web poslužitelj.

Slika 1. Raspberry Pi hardver

2.2 Raspberry Pi OS i Linux



Slika 2. Linux logo

Raspberry Pi kao mikroračunalo, poput svakog drugog računala, treba operacijski sustav. Već pri dizajniranju, ideja je bila da se na Raspberry Pi-ju pokreće sustav otvorenog koda (engl. *open-source*), tj. sustav kod kojega je moguće mijenjati izvorni kod i prilagođavati ga vlastitim potrebama. Operacijski sustav otvorenog tipa koji je najviše prihvaćen od strane korisnika Raspberry Pi-ja je Linux. To ne začuđuje iz razloga što je besplatan, ali najveći razlog leži u tome što ARM-ova jezgra procesora podržava Linux, za razliku od većine ostalih operacijskih sustava koji su razvijani isključivo za Intelovu procesorsku arhitekturu. Međutim, treba napomenuti da ARM-ova jezgra procesora ne podražava baš svaku Linux distribuciju.

Linux je prvenstveno naziv za jezgru operacijskog sustava koju je izmislio Linus Torvalds za svoje osobno računalo koje je pokretao procesor sa Intelovom x86 arhitekturom. Nedugo nakon toga odlučio je programski kod jezgre objaviti na internetu i pozvati ljude da zajedno s njim sudjeluju u nastajanju operacijskog sustava. Puno ime onoga što danas zovemo Linux ili Linux distribucije zapravo je *GNU/Linux* i to je ustvari ime za jednu obitelj operacijskih sustava, a svaki član te obitelji ima svoje posebne značajke, poput mnoštva biblioteka koje su dodane na Linux jezgru. Dakle, ono što Linux čini različitim od standardnih operacijskih sustava je to što je nastao kao kreacija ljudi diljem svijeta te se i danas ljudi bave njegovim nadograđivanjem, a izvorni kod mu je u potpunosti slobodan i otvoren za izmjene.

Tako je i Raspberry Pi OS samo jedan od Linux distribucija koji se često nalazi na Raspberry Pi hardveru. Raspberry Pi OS je baziran na *Debian*-u koji je besplatan i pruža nešto više od običnog operacijskog sustava. Kreirala ga je mala skupina programera koji su obožavatelji Raspberry Pi hardvera i naravno *Debian* projekta.

Slika 3. Raspberry Pi OS logo

2.3 PHP: Hypertext Preprocessor (PHP)

PHP je interpretacijski programski jezik koji se obrađuje na poslužitelju. Sintaksa PHP programskog jezika orijentirana je prema sintaksi programskih jezika *C* i *Perl*. PHP je namijenjen isključivo za izradu dinamičnih web stranica i aplikacija. Ne koristi se za programiranje desktop aplikacija namijenjenih jednom korisniku, već se PHP kod obrađuje na poslužitelju, a korisnici poslužitelju pristupaju putem web preglednika. Takav način komunikacije nazivamo klijentsko-poslužiteljska arhitektura (engl. *Client-server architecture*). PHP se ističe širokom podrškom raznih baza podataka i internet protokola kao i raspoloživošću brojnih programerskih biblioteka. Važno je naglasiti da se radi o proizvodu otvorenog koda (engl. *Open-source*), što znači da postoji pristup izvornom kodu kojeg je moguće koristiti, mijenjati i dalje distribuirati potpuno besplatno. U praktičnom dijelu rada PHP će omogućiti upravljanje bazom podataka preko *phpMyAdmin* sučelja.

Slika 4. PHP logo

2.4 MySQL

MySQL je poslužitelj baza podataka (engl. *Database server*). Drugim riječima, radi se o softveru kojem se može pristupiti preko mreže na sličan način kao i web (HTTP) poslužiteljima. Razlika je u tome da se *MySQL*-u obično pristupa pomoću korisničkog imena i lozinke. *MySQL* ima dobro dokumentirane module i ekstenzije te podršku brojnih programskih jezika: PHP, *Java*, *Perl*, *Python*, *Node.js* itd. *MySQL* baze su relacijskog tipa, koji se pokazao kao najbolji način skladištenja i pretraživanja velikih količina podataka i u suštini predstavljaju osnovu svakog informacijskog sustava, tj. temelj svakog poslovnog subjekta koji svoje poslovanje bazira na dostupnosti kvalitetnih i brzih informacija. Na serveru može postojati veći broj baza podataka koje su potpuno samostalne, no unutar jednog projekta mogu se upotrebljavati podaci iz više baza podataka na serveru. Svakom korisničkom računu na serveru moguće je dodijeliti razna administracijska prava nad cijelim serverom ili pojedinom bazom podataka. Neka od

prava mogu biti: stvaranje novih baza, pravo pristupa postojećim bazama, pravo uređivanja (unosa ili izmjena podataka) postojećih baza itd. Pri instalaciji *MySQL*-a se stvara tzv. super administrator (obično se zove *root*) koji ima sva administracijska prava. Jedna od velikih prednosti *MySQL*-a je što postoje verzije za sve važnije operacijske sustave te ih se distribuira pod GPL licencom (engl. *General Public Licence*). Jedan od poslužitelja *MySQL* baze podataka je *MariaDB* server koji je korišten u ovome radu.

Slika 5. MySQL logo

2.5 JavaScript (JS)

Slika 6. JavaScript logo

JavaScript je programski jezik koji nam omogućuje manipulaciju HTML elemenata. Proširuje se na HTML i omogućuje interaktivnost. On manipulira HTML-om preko DOM-a. Objektni model dokumenta (engl. DOM – *Document Object Model*) je aplikacijsko programsko sučelje (engl. API) za strukturirane dokumente kao što je HTML. Dokument i elemente dokumenta DOM uvijek definira kao objekte, a programsko sučelje tad čine određena svojstva samog objekta i metode tih objekata. Takvim objektima se preko programskoga sučelja može dinamično pristupati, ažurirati sadržaj i izgled, mijenjati strukturu kompletnog dokumenta neovisno o programskom jeziku. Osim DOM-a postoji i BOM (engl. *Browser Object Model*). BOM omogućuje komunikaciju *JavaScript*-a i preglednika. Osim razvoja web aplikacija i stranica, *JavaScript* se danas koristi i za izradu mobilnih aplikacija, igara i mrežnih aplikacija.

U pregledniku, *JavaScript* se pokreće preko tzv. *JavaScript Engine*-a. Na primjer, u *Mozilli* se taj engine zove *SpiderMonkey*, a u *Google Chrome*-u *v8*. Također, *JavaScript* se koristi u *Node.js* koji je *v8* engine izdvojio iz *Chromium*-a te ga pokreće kao skriptni jezik na poslužitelju. *JavaScript* posjeduje i brojne programske biblioteke. Najpoznatiji su *jQuery*, *Angular*, *React* i *Vue.js*. *JavaScript* omogućuje i rukovanje podacima u realnom vremenu, bez osvježavanja stranice, a za to se koristi AJAX. U radu se *JavaScript* koristi za dinamičko upravljanje elementima. Uz to, *JavaScript* je korišten i kao dio sučelja *Fetch* API. *Fetch* API je sučelje za dohvat resursa (engl. *Resources*). Koristi se za dohvat podataka iz vanjskog API-ja te za generiranje predloška s popisom proizvoda.

2.5.1 jQuery

jQuery je *JavaScript* biblioteka koja je dizajnirana za pojednostavljenje skriptiranja HTML-a na strani klijenta. To je brza i sažeta biblioteka koju je John Resig stvorio 2006. *jQuery* je otvorenog koda i slobodno je dostupan s glavnim motom: „*manje piši, više radi*“. Jedna je od najčešće korištenih i rasprostranjenih *JavaScript* biblioteka za internetske svrhe. Podržava više preglednika i više platformi. *jQuery* pojednostavljuje proces skriptiranja *front-end* dijela aplikacije. Pojednostavljuje upravljanje događajima i animacijama. Također pomaže u pozivima AJAX-a za brzi razvoj aplikacija. Skup značajki *jQuery*, uključujući odabir DOM elemenata, prelazak i manipulaciju, koji omogućuje mehanizam odabira, stvorio je stil programiranja, kombinirajući algoritme i strukture podataka vezane za DOM. Biblioteka *jQuery* slijedi modularni pristup koji omogućuje stvaranje dinamičkih web stranica, a time i web aplikacija. Ova je biblioteka tako znatno olakšala izradu ove web aplikacije.

Slika 7. *jQuery* logo

2.6 Node.js (Node)

Node.js je *JavaScript* okvir, dakle može se reći da je *Node.js* zapravo drugačija inačica *JavaScript*-a koja omogućava da se *JavaScript* kod pokreće na poslužitelju, odnosno bilo kojem računalu izvan web preglednika. *Node.js* je asinkron i većinom baziran na događajima (engl. *events*). *Node.js* je baziran na *Google*-ovom *Chrome* V8 *JavaScript Engine*-u. *V8* nije verzija već samo ime engine-a koji je originalno namijenjen za korištenje u browseru, no kasnije je prešao na stranu otvorenog koda i na taj način postao podloga za *Node.js*. Glavna karakteristika *Node.js*-a je to što je ne-blokirajući, što znači da se svi događaji definirani u *Node.js*-u dodaju u tzv. skup događaja (engl. *event pool*) te se registriraju i pokreću samo onda kada je to potrebno bez blokiranja daljnjeg izvršavanja koda. Može se reći da su ti događaji glavni elementi *Node.js*-a kao programskog jezika te se rad s *Node.js*-om temelji na njima. Ti događaji mogu imati povratne funkcije (engl. *callbacks*) koje se pozivaju nakon izvršavanja događaja. Takav način rada vrlo je pogodan za izradu pravovremenih brzih i jednostavnih aplikacija zato što izvršavanje manje bitnog koda ne bi blokiralo izvršavanje koda za npr. prikaz same aplikacije. *Node.js* dodaje nove mogućnosti *JavaScript*-u koje nisu dostupne unutar web preglednika, primjerice, pristupanje i modificiranje lokalnih datoteka. Također, bez obzira je li aplikacija izrađena u *Node.js*-u, na klijentskoj strani se i dalje pokreće samo *JavaScript*, dok je *Node.js* ograničen na pokretanje na poslužiteljskoj strani. Još jedna jaka strana *Node.js*-a je mogućnost proširivanja pomoću biblioteka. Jednostavno rečeno, biblioteke su dodaci *Node.js* okviru u obliku *JavaScript* biblioteka koje se mogu uključiti u izvedbu bilo koje skripte. Te biblioteke se mogu dodavati u *Node.js* okruženje poput *Node Package Manager*-a (*npm*).

Slika 8. Node.js logo

2.6.1 Express.js

Express.js, ili samo *Express* efikasna je biblioteka koja rješava problem oko pisanja HTTP zahtjeva, što uključuje rukovanja sa zaglavljima (engl. *header*) i podacima iz tijela (engl. *body*) HTML dokumenata uz brojna druga poboljšanja. Neke od prednosti *Express*-a su što se jednostavno postavlja i koristi. Pojednostavljuje rukovanje zahtjevima uvođenjem tzv. *middleware*-a, funkcija koje se pokreću tijekom zahtjeva. Također daje mogućnost lakog serviranja statičnih dokumenata, poput slika. Jednostavno se integrira sa poznatim bazama podataka kao što su *MongoDb* ili *MySQL*. To je sve moguće i bez *Express*-a, no kako bi kod ostao pregledan i kako bi se olakšala izrada same web aplikacije, preporučljivo je koristiti *Express*. *Express* je iz istog razloga izrazito korišten u izradi praktičnog dijela ove aplikacije.

Slika 9. Express.js logo

2.7 Tehnologije korištene za označavanje sadržaja aplikacije

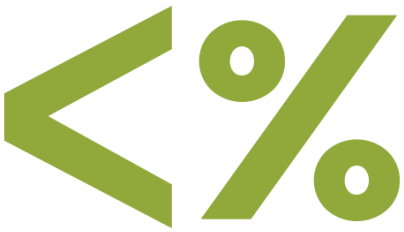
2.7.1 HyperText Markup Language (HTML)

HTML (engl. *HyperText Markup Language*) je prezentacijski jezik za izradu web aplikacija. HTML se ne smatra programskim jezikom jer se za njegovo korištenje ne primjenjuje nikakva logika. Služi za oblikovanje web stranica i aplikacija. Putem HTML navodimo računalo gdje se koji element mora nalaziti na zaslonu. Budući da HTML ne omogućuje dinamičnost prikaza strukture web stranice u ovoj aplikaciji se zato koristi EJS koji proširuje HTML dodajući mu programsku logiku.

Slika 10. HTML logo

2.7.2 Embedded JavaScript (EJS)

Embedded JavaScript (EJS) jedan je od dostupnih *templating engine*-a koji se mogu koristiti kako bi se lakše i dinamično generirao HTML sadržaj. EJS uvodi dodatnu sintaksu kojom omogućuje da se unutar HTML datoteka piše *JavaScript* i koriste varijable koje se proslijede putem *Node.js*-a. EJS podržava i zadavanje vlastitih sintaksi kod prosljeđivanja HTML sadržaja. Postoje i drugi *templating engine*-i, primjerice, *Pug* i *Handlebars*, no EJS je odabran za korištenje u projektu iz razloga što se koristi HTML i *JavaScript*, dok se primjerice kod *Pug*-a, koristi prilagođen HTML jezik i sintaksa. EJS je vrlo jednostavno razumjeti te je pogodan za kretanje kroz petlju i ispisivanje dinamičnog broja sadržaja.



Slika 11. EJS logo

2.8 Tehnologije korištene za stiliziranje sadržaja aplikacije

2.8.1 Cascading Style Sheets (CSS)

Slika 12. CSS logo

CSS (engl. *Cascading Style Sheets*) ili jezik kaskadnog oblikovanja služi za dizajniranje i opisivanje prezentacije HTML-a. Također, omogućuje prilagodljiv dizajn kako bi korisničko iskustvo bilo dobro i na manjim ekranima. Primjerice, preko CSS-a možemo mijenjati font, boje ili odabrati veće razmake između elemenata. Danas je sve popularniji i modularni pristup CSS dizajniranju pa se i same CSS datoteke razdvajaju u zasebne dijelove. CSS ima i svoje okvire koji se koriste za kreiranje dizajna. Neki od najpoznatijih su *Bootstrap*, *Skeleton* i *Semantic*. Najkorišteniji je *Bootstrap*. *Bootstrap* je vrlo popularan jer uvelike olakšava programeru implementaciju prilagodljivog dizajna, ali i olakšava dizajniranje web aplikacija jer programer ne mora pisati svoj CSS.

2.8.2 Bootstrap i Bootstrap Icons

Slika 13. Bootstrap logo

Bootstrap je skup alata kreiranih u CSS-u i *JavaScript*-u koji omogućava brzu, kvalitetnu i konzistentnu izradu web stranica. To je okvir za razvijanje vidljivog dijela web aplikacija i web stranica, a dostupan je svima jer je besplatan i otvorenog koda (engl. *open-source*). Pruža skup komponenata za izradu kvalitetnog grafičkog sučelja (engl. *User Interface*) te mrežni sustav bitan za prilagodljive web stranice koje se trebaju ispravno prikazivati na svim uređajima uključujući mobilni, tablet, prijenosno računalo i stolno. Kako bi korisniku olakšali korištenje web aplikacije uključio sam i *Bootstrap* ikone koje će korisniku olakšati snalaženje na stranici. Kako bi se pridonijelo osobnome stilu web aplikacije, iskorištene su *Sass* mogućnosti *Bootstrap*-a kako bi se dizajn prilagodio već postojećoj stranici škole.

2.8.3 Syntactically Awesome Style Sheets (*Sass*)

SCSS (engl. *Sassy Cascading Style Sheets*) naprednija je verzija CSS jezika koja dodaje dodatnu funkcionalnost CSS jeziku te daje web programerima veću fleksibilnost i kontrolu prilikom izrade web dizajna. Koristi istu sintaksu kao i CSS koji zahtijeva zagrade i točku-zarez kako bi odredio blokove i završetke redaka. Većina preglednika ne može razumjeti SCSS i SASS (engl. *Syntactically Awesome Style Sheets*) te ih se mora kompilirati u CSS prije nego što se mogu koristiti u web pregledniku. Jednako kao u programskom jeziku u SCSS-u dostupne su varijable. Varijable se često spremaju u zasebne liste kako bi ih korisnik mogao pozivati u projektu na više mjesta. Prilikom promjene jedne varijable ažurira se sadržaj na više mjesta u stilskoj listi te na takav način programer može uštedjeti vrijeme.

Slika 14. Sass logo

2.9 Python

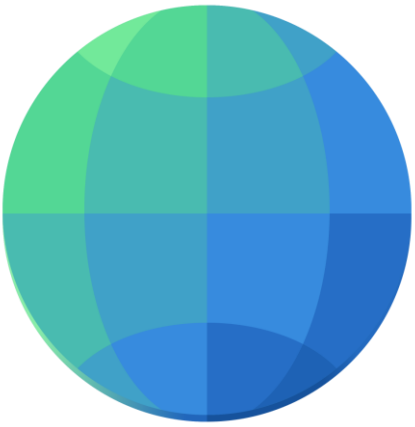
Python je interaktivni i objektno orijentirani programski jezik nastao 1991. godine, a razvio ga je Guido van Rossum. Python svoju snagu i brzinu kombinira s vrlo jasnom i jednostavnom sintaksom što ga svrstava u sam vrh najkorištenijih programskih jezika. On spada u interpretacijsku skupinu programskih jezika, što znači da prevodi naredbe u trenutku izvođenja samog programa (prevodi liniju po liniju). Također, Python spada i u skriptne jezike. Skriptni

jezici su svi oni jezici koji se mogu ugraditi u HTML. Popularnost Pythona toliko je velika zbog toga što se koristi u razne svrhe, poput: upravljanja podacima i bazama podataka, strojnog učenja (engl. *Machine Learning*), izrade web stranica i aplikacija, programiranja informacijskih sustava, matematičkih (3D modeliranje, grafovi, projekcije) i znanstvenih istraživanja. U ovoj aplikaciji koristi se kao pokretačka skripta koja postavlja početno stanje Raspberry Pi računala.

Slika 15. Python logo

2.10 GNOME Web

GNOME Web internetski je preglednik za GNOME radno okruženje, drugim riječima web preglednik za Linux. Ima jednostavno i intuitivno sučelje koje omogućuje njegovu laku i brzu upotrebu. Zbog svojeg jednostavnog sučelja izabran je kao sučelje koje će u ovoj web aplikaciji pokazivati vizualni sadržaj. *GNOME Web* često se naziva po svojem kodnom nazivu, *Epiphany*.



Slika 16. GNOME Web logo

3 Tehnički opis i izvedba rada

3.1 Konfiguracija Raspberry Pi-ja

Na samom početku rada, započeo sam s idejom da koristeći Raspberry Pi proširim ograničenja softvera televizora. U početku, Raspberry Pi korišten je kao softver koji je imao aplikaciju za prikaz vizualnih sadržaja u obliku prezentacije. Na taj način proširene su softverske mogućnosti televizora Raspberry Pi-jem. Takva konfiguracija bila je funkcionalna, ali ne i praktična, jer je prijenos novih slika bio kompleksan proces koji nije mogao obavljati svaki zaposlenik škole. Tako sam došao do ideje napraviti posebnu aplikaciju koja bi spremala sve željene postavke i slike za vizualni prikaz na televizoru i slala to svakom televizoru u školi. Odlučio sam se za web aplikaciju jer joj je lakše pristupiti s bilo kojeg mjesta u školi. Pojavila se potreba za serverom na kojem će se web aplikacija nalaziti. Pošto je za televizor u holu škole i dalje bio potreban Raspberry Pi, donesena je odluka da Raspberry Pi pretvorim u server i njime pokazujem vizualni sadržaj na televizoru. Dodavanjem novog televizora bilo bi potrebno samo taj televizor postaviti na kiosk način web pregleda određene web stranice ove web aplikacije.

S obzirom da škola trenutno upotrebljava samo jedan televizor, imamo i samo jedan Raspberry Pi koji je u ovom slučaju korišten kao server i kao korisnik aplikacije. Kako bi Raspberry Pi kao i sam televizor bio dostupan za rad, potrebno mu je napajanje koje dobiva iz produžnog kabla koji se proteže do televizora. Raspberry Pi-ju još je potrebna i računalna mreža kako bi se moglo pristupiti web aplikaciji, tako da je Raspberry Pi spojen na školsku mrežu preko *ethernet* kabla. Na kraju, kako bi televizoru bilo omogućeno prikazivati vizualni sadržaj Raspberry Pi-ja, Raspberry Pi spojen je pomoću HDMI kabla na televizor. Televizor nije potrebno držati uključenim cijelo vrijeme kao i Raspberry Pi, no preporučeno je da se Raspberry Pi ne isključuje radi serverskih podataka i boljeg rada same web aplikacije.

Za rad aplikacije i televizora nije potrebno dodatno održavanje. Dovoljno je televizor po potrebi uključivati i isključivati. Naglašavam da bi bilo dobro Raspberry Pi čistiti barem svakih godinu dana, a najbolje svakih šest mjeseci zbog nakupljanja prašine u samome uređaju kao i njegove okoline.

3.1.1 Konfiguracija operacijskog sustava

Raspberry Pi kao operacijski sustav ima Raspberry Pi OS čiji su način rada i funkcija objašnjeni u teorijskom dijelu rada. Kao i svaki drugi operacijski sustav, mora imati korisnika za pristup i rad u operacijskom sustavu. Za običnog korisnika postoji korisnički račun pod korisničkim imenom *„pi“.* Prijava na taj korisnički račun nalazi se u nadolazećem poglavlju „*Podaci za prijavu*“. Kako bi se na televizoru mogao prikazati vizualni sadržaj, Raspberry Pi konfiguriran je tako da se navedeni korisnik *„pi“* automatski prijavi na uređaj bez provjere identiteta. Potrebno je napomenuti da je uobičajeno pristupati Raspberry Pi-ju isključivo preko web aplikacije. Međutim, na Raspberry Pi-ju inicijalno je uključen FTP server kojemu se može pristupiti podacima za prijavu kao i za web aplikaciju. U nastavku teksta, sve podatke za prijavu možete potražiti u već spomenutom poglavlju „*Podaci za prijavu*“.

Kako bi svi servisi bili aktivni, o njima se brine skripta koja je napisana u Python programskom jeziku. Navedena skripta, koja se nalazi na sljedećoj lokaciji na Rasperry Pi-ju: */home/pi/TSD-TV/python/start-script.py* pokreće sve bitne dijelove za rad Raspberry Pi-ja. Skripta najprije pokreće novi terminal koji čini korisnički miš nevidljivim radi boljeg prikaza vizualnih sadržaja na televizoru te zadržavanja pažnje na samoj web aplikaciji. Nadalje, web aplikacija pokreće FTP server, PHP server s web aplikacijom *phpMyAdmin* i ovu web aplikaciju koja je ključni dio ovoga rada. Kao završni korak skripta pokreće internetski preglednik naziva GNOME Web u kiosk načinu rada. Pokretanjem preglednika, koji je opisan na početku rada, u kiosk načinu rada osigurali smo skrivanje svog neželjenog sadržaja i pokazivanje samo bitnog sadržaja. Sve skripte navedene za rad ove aplikacije većinom se nalaze na sljedećoj putanji: */home/pi/TSD-TV/.*

Na kraju bih naglasio kako SSH server na ovome uređaju nije inicijalno pokrenut, no moguće ga je pokrenuti putem sučelja web aplikacije u slučaju nužde ili rada na Raspberry Pi-ju. SSH inicijalno je isključen zbog slabe postavljene lozinke za prijavu u sustav koja je tako postavljena zbog jednostavnosti pristupa.

3.1.2 Mrežna konfiguracija

Kao što je već spomenuto, Raspberry Pi spojen je žičanom vezom na školsku računalnu mrežu preko sučelja „eth0“. Potrebno je napomenuti kako Raspberry Pi-ju ne možemo pristupiti

izvan računalne mreže škole. Drugim riječima, nije dostupan na internetu, što zapravo predstavlja značajnu razinu informacijske sigurnosti. Kako bi se mogao postaviti DNS ili uopće pristupiti uređaju, Raspberry Pi-ju potrebna je statička IPv4 adresa. Statička IPv4 adresa identifikator je uređaja na računalnoj mreži te se ne mijenja vremenom. Za razliku od statičke, dinamička IPv4 adresa mijenja se u određenim vremenskim intervalima. Budući da smo izabrali statičku IPv4 adresu, moramo računalu tj. Raspberry Pi-ju samo dodijeliti jednu slobodnu IPv4 adresu. Nadalje, Raspberry Pi nalazi se u 1*72.25.26.0* računalnoj mreži *subnet* maske *255.255.255.0* (*/24*). Navedene postavke nalaze se u datoteci *dhcpcd.conf* koja se može pronaći na sljedećoj lokaciji: */etc/dhcpcd.conf*. Detaljniju konfiguraciju moguće je pronaći u sljedećoj tablici ili u sadržaju datoteke *dhcpcd.conf.*

Tablica 1. Mrežna konfiguracija Raspberry Pi na interfejsu: eth0

Mrežna konfiguracija Raspberry Pi na interfejsu: eth0	
IPv4 adresa	172.25.26.65
Mreža	*172.25.26.0/24*
Subnet maska	*255.255.255.0* (*/24*)
Usmjernik (engl. *router*)	*172.25.26.5*
DNS	*172.25.26.190*, *8.8.8.8*, *8.8.4.4*, *1.1.1.1*

```
interface eth0
static ip_address=172.25.26.65/24
static routers=172.25.65.5
static domain_name_servers=172.25.26.190 8.8.8.8 8.8.4.4 1.1.1.1
noipv6
inform 172.25.26.65
```

Slika 17. Datoteka: /etc/dhcpcd.conf

3.1.3 Pokretačke skripte

Kao što je već navedeno, pokretačka skripta pisana je u Python programskom jeziku i zadužena je za pokretanje servera i postavljanje Raspberry Pi-ja u početno stanje. Skriptu je

moguće pronaći na sljedećoj lokaciji: */home/pi/TSD-TV/python/start-script.py.* Kako bi se skripta pokrenula zajedno s Raspberry Pi-jem, potrebno je stvoriti datoteku *start-script.desktop* na sljedećoj lokaciji: */home/pi/.config/autostart/start-script.desktop.* Stvaranjem navedene datoteke Linux sustav će pri automatskoj prijavi korisnika *„pi“* pročitati i analizirati navedenu datoteku koja će operacijskom sustavu reći što mora činiti pri pokretanju.

```
[Desktop Entry]
Type=Application
Name=Start Script
NoDisplay=false
Exec=gnome-terminal --command="python /home/pi/TSD-TV/python/start-script.py"
Terminal=true
```

Slika 18. Datoteka: /home/pi/.config/autostart/start-script.desktop

Na slici 18. uviđamo kako datoteka govori operacijskom sustavu da pokrene naredbu koja pomoću *Python*-a pokreće pokretačku skriptu. Ova datoteka bit će izvršena tek onda kada se korisnik *„pi“* prijavi u računalo.

Pokretanjem skripte, Python prvo izvršava naredbu koja otvara novi prozor terminala na radnoj površini, sa svrhom da učini miš operacijskog sustava nevidljivim pomoću biblioteke *„unclutter“.* Sakrivajući miš radi ljepšeg prikaza, pokretačka skripta zatim poziva u novom terminalu skriptu koja upravlja PHP serverom, koji će biti detaljnije opisan u daljnjem tekstu. Zatim pokretačka skripta otvara internetski preglednik u kiosk načinu rada za prikaz vizualnog sadržaja. Internet preglednik kojeg koristim naziva se GNOME Web i pomoću njega otvaram sljedeću web stranicu: *http://127.0.0.1/tv?id=main-tv*. Pomoću ove poveznice Internetski preglednik spaja se na web aplikaciju, govoreći joj koji je od televizora napravio zahtjev. Mijenjajući svojstvo *id* također možemo doći i do drugih sučelja za druge televizore po želji. Nadalje, pokretačka skripta ponovno otvara novi terminal koji poziva Python skriptu, ovog puta za FTP server. FTP skripta kao i PHP skripta će biti objašnjene u daljnjem tekstu. Napokon, skripta ulazi u beskonačnu petlju koja pokreće web aplikaciju. U slučaju pada web aplikacije, beskonačna petlja ponovno je pokreće te web aplikacija nastavlja s dosadašnjim radom.

Rad pokretačke skripte moguće je vidjeti na sljedećoj slici.

```
import os

os.system('gnome-terminal -- bash -c "unclutter -idle 1"')
os.system('gnome-terminal -- bash -c "sudo python3 /home/pi/TSD-TV/python/php.py"')
os.system('gnome-terminal -- bash -c "/usr/bin/chromium-browser --kiosk --start-fullscreen http://127.0.0.1/tv?id=main-tv"')
os.system('gnome-terminal -- bash -c "sudo python3 /home/pi/TSD-TV/python/ftp.py"')
os.system('cd /home/pi/TSD-TV/node.js/')
os.system('gnome-terminal -- bash -c "sleep 10s;midori http://127.0.0.1/tv?id=main-tv --display=:0 -e Fullscreen"')

#os.system('cd /home/pi/TSD-TV/node.js/;sudo /usr/bin/npm install')
while True:
        os.system('sudo node /home/pi/TSD-TV/node.js/server.js')
```

Slika 19. Datoteka: /home/pi/TSD-TV/python/start-script.py

3.2 FTP server

FTP server pokreće se i pisan je u Python programskom jeziku. Glavni dio skripte je biblioteka *„pyftpdlib“* koja nam pruža mogućnosti konfiguracije i prilagodbe rada FTP servera. Ukratko, skripta na početku očitava konfiguracijsku datoteku servera u kojoj se nalaze korisnički podaci za prijavu administratora te ga dodaje u autorizaciju, tako omogućujući da se samo administrator može prijaviti na FTP server. Na kraju, skripta pokreće FTP server na uobičajenom portu 21. Prijava na FTP server može se i mora vršiti pomoću softvera koji podržava FTP protokol. Najjednostavniji je način preko *Eksplorera za datoteke* Microsoft Windowsa pristupiti FTP serveru, no to je moguće učiniti i s FTP softverima od kojih je možda najpoznatiji tzv. *„FileZilla“.* Navedeni opis i način rada skripte može se uočiti na slici 20., gdje se može vidjeti i mnogo više detalja.

Glavna primjena ovoga FTP servera je prijenos podataka između nekog računala na mreži i Raspberry Pi-ja. Najčešća svrha FTP servera je omogućiti pristup podatcima na Raspberry Pi-ju preko mreže te u slučaju ažuriranja aplikacije obnoviti kod za web aplikaciju, ili sl.

```python
import os
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer
import json

config = json.loads('{"admin":{"username":"admin", "password":"admin2"}}')

try:
    f = open('/home/pi/TSD-TV/node.js/config/config.json')
    config = json.load(f)
    f.close()
except:
    config['admin']['username'] = 'admin'
    config['admin']['password'] = 'admin'

def main():
    authorizer = DummyAuthorizer()
    authorizer.add_user(config['admin']['username'], config['admin']['password'], '/', perm='elradfmwMT')

    handler = FTPHandler
    handler.authorizer = authorizer
    handler.banner = 'pyftpdlib based ftpd ready'

    address = ('', 21)
    server = FTPServer(address, handler)

    server.max_cons = 5

    server.serve_forever()

if __name__ == '__main__':
    main()
```

Slika 20. Datoteka: /home/pi/TSD-TV/python/ftp.py

3.3 phpMyAdmin server

PHP server pokreće se u Python programskom jeziku. Za razliku od FTP servera, nije pisan u Pythonu i dosta je jednostavnije strukture. Glavni dio skripte je pozvati operacijski sustav da pokrene unaprijed postavljeni PHP server. Ukratko rečeno, pokretanjem PHP servera pokreće se i *phpMyAdmin* web aplikacija koja je instalirana na njemu. *PhpMyAdmin* je web aplikacija poput ove, a pisana je u programskom jeziku PHP. *PhpMyAdmin* pruža administratoru udaljen pristup bazi podataka preko grafičkog sučelja. Navedeno grafičko sučelje na taj način olakšava i pruža pristup i uvid u bazu podataka za ovu web aplikaciju. PHP server pokreće se na portu 8080, kao i web aplikacija *phpMyAdmin*. Pristupiti web aplikaciji također možemo pomoću grafičkog sučelja. Prijava u *phpMyAdmin* ili bazu podataka može se pronaći u sučelju ove web aplikacije ili u poglavlju „*Podaci za prijavu*“. Kao što je rečeno, ova skripta je kratka i jednostavna, a izvorni kod skripte možete vidjeti na sljedećoj slici.

```
import os

os.chdir('/home/pi/TSD-TV/php/')
os.system('sudo php -S 0.0.0.0:8080 -c /etc/php/7.4/cli/php.ini')
```

Slika 21. Datoteka: /home/pi/TSD-TV/python/php.py

3.4 Node.js server

Node.js, za razliku od PHP i FTP poslužitelja, nema svoju posebnu skriptu nego se nalazi u samoj startnoj skripti. Nakon što startna skripta pokrene sve poslužitelje, ulazi u beskonačnu petlju pokretanja *Node.js* poslužitelja. Ako dođe do bilo kakvoga kvara ili gašenja *Node.js* poslužitelja, beskonačna petlja nam osigurava da se poslužitelj ponovno pokrene. Drugim riječima, *Python* startna skripta pokreće *JavaScript* skriptu ove aplikacije. Glavna skripta ove web aplikacije nalazi se na sljedećoj lokaciji: */home/pi/TSD-TV/node.js/server.js*. Kao što sam već ranije spomenuo, skripta je napisana u programskom jeziku *JavaScript* te je pokreće engine *Node.js*-a. Sam rad navedene skripte i web aplikacije bit će objašnjen u daljnjem tekstu.

3.5 Baza podataka

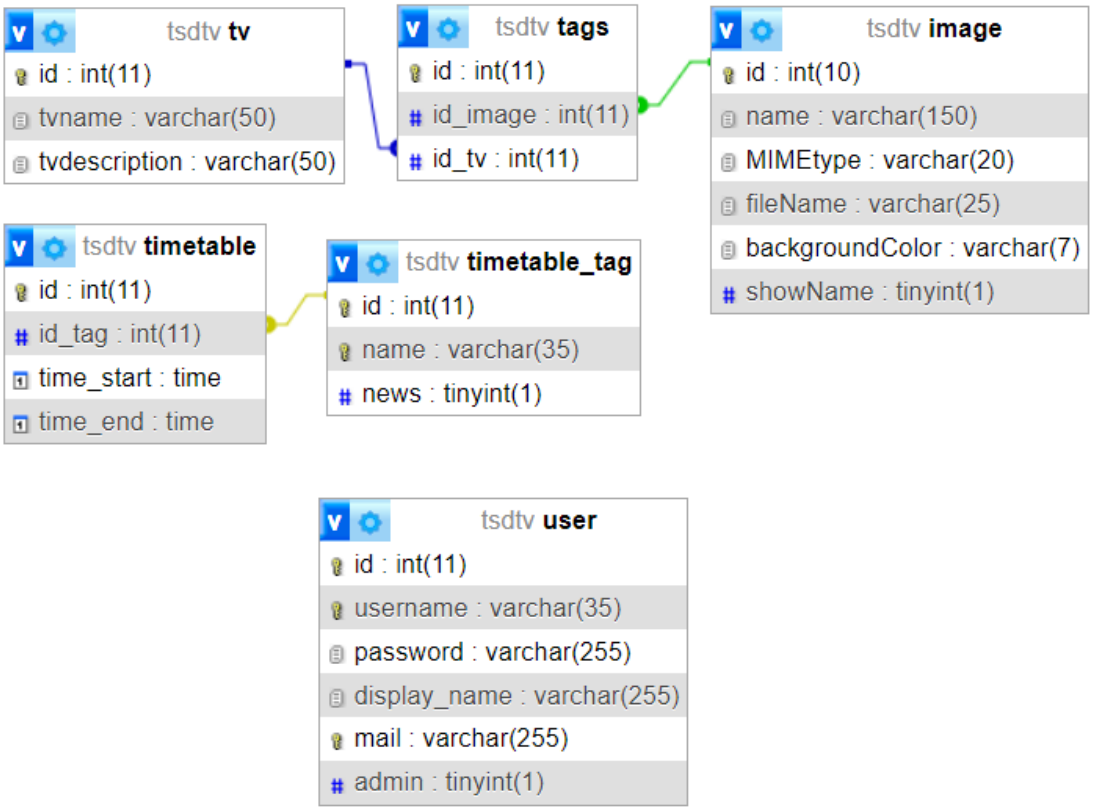
Ova web aplikacija, kao programsko rješenje, koristi relacijsku bazu podataka. To znači da se podaci spremaju u tablice, koje međusobno mogu imati određenu relaciju. Baza podataka na Raspberry Pi-ju inicijalno se pokreće u isto vrijeme kada i računalo kao pozadinski servis. Biblioteka tj. softver za bazu podataka zove se „*MariaDB“* i to je samo softversko rješenje za *MySQL* server baze podataka. Baza podataka web aplikacije nosi naziv „*tsdtv*“ i u njoj su svi podaci koje web aplikacija sakuplja kroz svoj uspješan rad. Baza podataka pokrenuta je preko porta 3306, a prijavu u bazu moguće je pronaći u poglavlju „*Podaci za prijavu*“ ili u web sučelju ove web aplikacije. Naime, bitno je spomenuti kako se bazi podataka može pristupiti i upravljati podacima putem već spomenutog alata *phpMyAdmin*.

3.5.1 Pravo pristupa

Bazi podataka može se pristupiti preko školske računalne mreže. No, spajanje na bazu podataka preko računalne mreže nije potrebno jer aplikacija automatski upravlja bazom podataka. Pristup postoji samo zbog mogućnosti kvara aplikacije. Svakako, mijenjanje baze podataka ne preporučuje se ako niste ovlašteni i sigurni u posljedice mijenjanja podataka u bazi podataka. Također, u bazi podataka postoji glavni korisnik *„tsdtv“,* za čiju prijavu se podaci mogu pronaći u poglavlju *„Podaci za prijavu“*.

3.5.2 Organizacija baze podataka

Kao što je već navedeno, ova web aplikacija koristi relacijsku bazu podataka kao svoje rješenje. To nam govori da su podaci spremljeni u zapise koji se nalaze u relacijskim tablicama. Tako su za rad ove web aplikacije potrebne tablice koje spremaju razne podatke – od podataka o vizualnim sadržajima za prikaz do tablice za popis televizora koji postoje u školi. Svaka tablica ima stupce u kojima se nalaze podaci poput imena slika ili slično. Među tim podacima također se mogu pronaći i posebni podaci kao identifikacijski brojevi zapisa u određenoj tablici. Ti brojevi omogućavaju stvaranje veza između više tablica, tako omogućavajući složene veze između tablica. Svaka tablica i njezina funkcija bit će objašnjena u daljnjem tekstu. Odnos između tablica najbolje je prikazati ER dijagramom, tj. grafičkim prikazom dizajna baze podataka. Sljedeća slika prikazuje dizajn i organizaciju podataka u bazi podataka ove web aplikacije.



Slika 22. ER dijagram baze podataka

3.5.2.1 Tablica „*image*“

Tablica 2. Tablica tsdtv.image u bazi podataka

tsdtv.image
id: int(10)
name: varchar(150)
MIMEtype: varchar(20)
fileName: varchar(25)
backgroundColor: varchar(7)
showName: tinyint(1)

Kako bi tablica bila u mogućnosti spremati vizualan sadržaj na televizoru, potrebno je te vizualne sadržaje tj. slikovne datoteke pohraniti na poslužitelj web aplikacije. Međutim, slikovne datoteke nije potrebno samo spremiti u memoriju, nego ih spremiti i u bazu podataka radi postavki prikazivanja naših vizualnih sadržaja. Sama aplikacija sve slikovne datoteke sprema u mapu na lokaciji: */home/pi/TSD-TV/node.js/data/gallery/,* uz to sprema podatke u tablicu

„image“. Zapis u tablici identificiran je posebnim identifikacijskim brojem koji se zove *id*. Stupci *fileName* i *MIMEtype* povezuju zapise u bazi podataka sa datotekom slike koja se nalazi u već spomenutoj mapi, gdje web aplikacija sprema sve slikovne datoteke. Ova tablica također pohranjuje svojstva slike kao što su: ime slike, postavka prikazivanja imena slike na televizor te kojom će bojom biti ispunjena pozadina televizora ispod same slike. Svojstvo *backgroundColor* sprema boju ispune ispod slike u RGB zapisu boje u formatu #*RRGGBB gdje RR, GG* i *BB* predstavljaju heksadecimalni broj u intervalu od 00 do FF. Svojstvo *showName* je tipa *tinyint* veličine od jednog znaka, moguće vrijednosti su 0 koja predstavlja da se ime ne prikazuje na televizoru, dok vrijednost 1 predstavlja kako je navedeno ime u svojstvu *name* potrebno ispisati na televizor.

3.5.2.2 Tablica „tv“

Tablica 3. Tablica tsdtv.tv u bazi podataka

tsdtv.tv
id: int(11)
tvname: varchar(50)
tvdescription: varchar(50)

Kako bi web aplikacija bila u stanju raspoznavati različite televizore, svakom televizoru moramo dodijeliti posebni zapis u tablici *„tv“*. Svaki zapis u tablici ima svoj jedinstveni identifikacijski broj također nazvan *id* i sadržava dva svojstva *tvname* i *tvdescription*. Svojstvo *tvdescription*, ukratko rečeno, sadržava tekst kojim opisujemo televizor. To svojstvo korisniku pojašnjava na koji se televizor misli. Svojstvo *tvname* važno je jer se njime identificira sučelje televizora. U ovome tekstu spomenuto je kako je internetski preglednik konfiguriran u kiosk načinu rada postavljen na web lokaciju: *http://127.0.0.1/tv?id=main-tv.* Parametar *id*, koji u ovom primjeru nosi vrijednost *main-tv*, određuje web aplikaciji koje sučelje treba poslati određenom televizoru. Na navedenu poveznicu potrebno je dodati parametar *id* s vrijednošću svojstva *tvname* iz tablice tv i tako omogućiti web aplikaciji mogućnost posebnog konfiguriranja svakog televizora.

3.5.2.3 Tablica „*tags*“

Tablica 4. Tablica tsdtv.tags u bazi podataka

tsdtv.tags
id: int(11)
id_image: int(11)
id_tv: int(11)

Tablica „*tags*“ ima ulogu prometne relacijske tablice u ovoj bazi podataka. Tablica pomoću jednog zapisa govori koja se slikovna datoteka treba pokazivati na kojem televizoru. Ovakav rad web aplikacije omogućuje postavljanje različitih slika na različitim televizorima, ali i puno složenije podijele. Prometna tablica bazirana je na način da svaki zapis ima svoj jedinstveni identifikacijski ključ *id* i uz to još dva strana svojstva tzv. strana ključa koja povezuju dva identifikacijska broja različitih tablica u više naprama više vezi ili tzv. *N:M* vezi.

3.5.2.4 Tablica „*timetable_tag*“

Tablica 5. Tablica tsdtv.timetable_tag u bazi podataka

tsdtv.timetable_tag
id: int(11)
name: varchar(35)
news: tinyint(1)

Ova tablica dio je sustava za raspored. Točnije, ova tablica ima popis svih mogućih školskih sati. Dva najdostavnija školska sata koja se koriste su radni sat i odmor. Također, može se podijeliti na radni sat, petominutni odmor i veliki odmor. Sve ovisi o željama škole. Svaki zapis u tablici je jedinstven, a kao i sve ostale tablice ima svoj jedinstveni identifikator *id* te svojstva *name* i *news*. Svojstvo *name* ime je školskog sata za prikaz u sučelju za podešavanje i također će se pokazivati na televizoru, dok je svojstvo *news* je tipa *tinyint* s dvije moguće vrijednosti. Ako je postavljena na 0 u tom školskom satu televizori neće prikazivati obavijesti nego samo vizualan sadržaj. Postavljenjem svojstva na vrijednost 1 televizori će prikazu obavijesti dati veći prioritet te na taj način omogućiti veći broj prikazanih obavijesti u određenom trenutku, ako je potrebno.

3.5.2.5 Tablica „*timetable*“

Tablica 6. Tablica tsdtv.timetable u bazi podataka

tsdtv.timetable
id: int(11)
id_tag: int(11)
time_start time
time_end time

Nakon stvaranja školskog sata prethodnom tablicom, potrebna je tablica koja će školske sate organizirati u raspored. Kao i sve tablice i ova tablica za svaki zapis ima jedinstveni identifikator pod nazivom *id*. Zapis u ovoj tablici govori o trajanju školskog sata. Toj je tablici potrebno svojstvo *id_tag* s tzv. stranim ključem koje će povezivati ovaj zapis sa školskim satom koji se nalazi kao zapis u tablici. Kako sam na ovaj način omogućio povezivanje zapisa i školskog sata, još mi ostaje da pomoću svojstava *time_start* i *time_end* odredim interval trajanja sata. Nakon ispravnog postavljanja, web aplikacija će moći odrediti televizorima koji školski sat je u tijeku i kako se trebaju ponašati u tom periodu.

3.5.2.6 Tablica „*user*“

Tablica 7. Tablica tsdtv.user u bazi podataka

tsdtv.user
id: int(11)
username: varchar(35)
password: varchar(255)
display_name: varchar(255)
mail: varchar(255)
admin : tinyint(1)

Kako bi web aplikaciji mogli pristupiti i ostali djelatnici škole, a ne samo administrator, potrebno je napraviti korisnike u web aplikaciji. U ovoj tablici svaki zapis predstavlja jednog korisnika u sustavu. Kao i kod svih tablica, svaki zapis tj. korisnik ima svoji jedinstveni identifikacijski broj pod nazivom *id*. Kako bi se korisnik mogao prijaviti na stranicu, potrebno mu je korisničko ime koje se sprema u svojstvo *username* te lozinka koja se sprema pod svojstvom password. Lozinke su u bazi podataka pohranjene u *HASH* enkripcijskom obliku radi sigurnosti podataka. Korisnik kao dodatne parametre ima sljedeća svojstva: ime za prikaz u sučelju koje se sprema u svojstvo *display_name*, elektroničku adresu koja se sprema u svojstvo *mail* i svojstvo *admin* tipa *tinyint* veličine jednog znaka. Navedeno svojstvo ima ulogu reći web aplikaciji ima li prijavljeni korisnik administratorske ovlasti u web aplikaciji. Postavljanjem

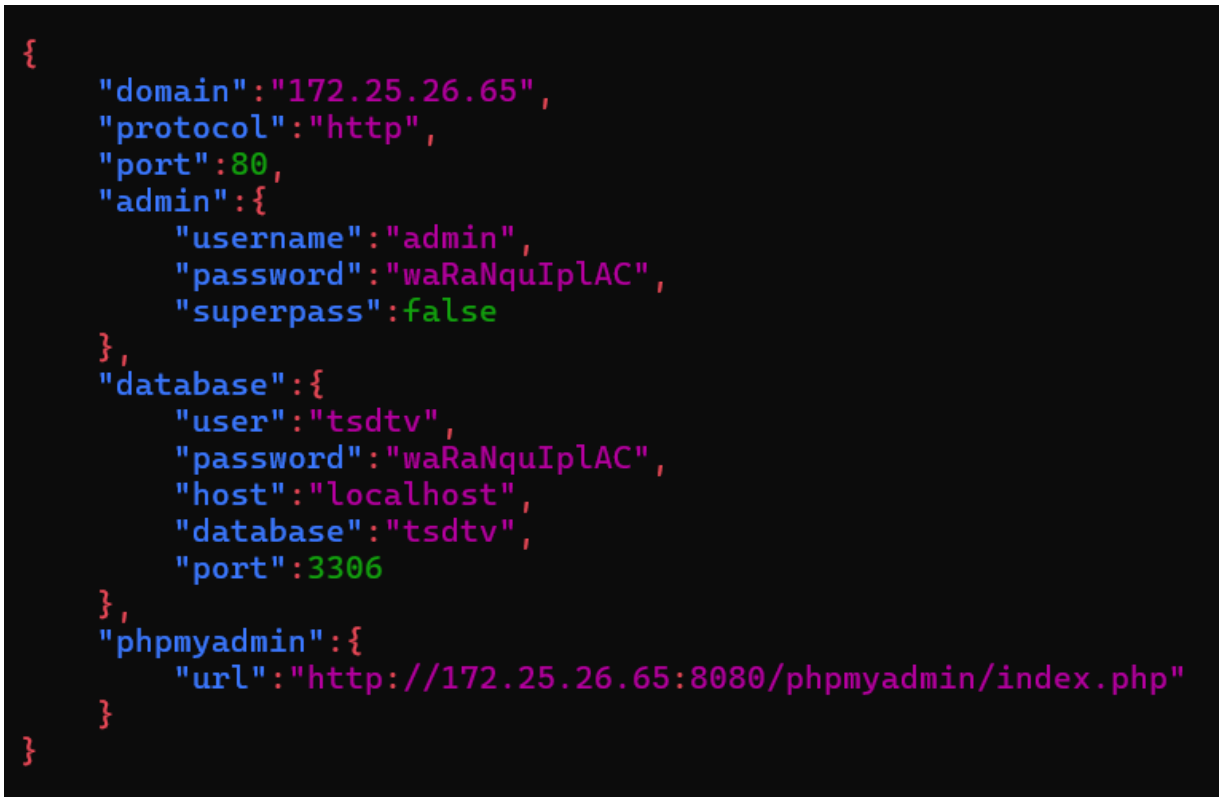
svojstva na vrijednost 1, određeni korisnik imat će punu kontrolu nad web aplikacijom kao i administrator. Postavljanjem svojstva na 0, web aplikacija će korisniku pružiti samo osnovna podešavanja aplikacije.

3.6 Rad web aplikacije

3.6.1 Konfiguracija servera

Ranije sam već spomenuo da se glavna skripta web aplikacije nalazi na sljedećoj lokaciji: */home/pi/TSD-TV/node.js/server.js*. Pisana je u programskom jeziku *JavaScript*, dok je izvodi engine softvera *Node.js*. Na početku skripte navodimo i deklariramo biblioteke za rad web aplikacije čiji se popis može detaljnije vidjeti u sljedećem poglavlju. Nakon toga se definiraju rute tj. definira se koja će skripta odgovarati na koji *HTTP* upit. Na kraju je potrebno napisati kod koji pokreće slušanje *HTTP* zahtjeva na server web aplikacije.

Uza sve to, važno je spomenuti da aplikacija ima i svoji konfiguracijski *JSON* objekt koji se nalazi na sljedećoj lokaciji: */home/pi/TSD-TV/node.js/config/config.json*. U ovome objektu nalaze se svojstva koja govore kako server treba biti konfiguriran. To su svojstva poput *domain, protocol* i *port*, a govore web aplikaciji na koji će se način serveru pristupati. U ovome primjeru vidljivo je da ćemo serveru pristupiti preko linka: *http://172.25.26.65:80/*. Zatim imamo sljedeći objekt koji nosi naziv *admin.* Ovaj objekt nam govori korisničke podatke za prijavu administratora u sustav, točnije njegovo korisničko ime i lozinku. Tu se također nalazi i svojstvo *superpass* koje omogućuje da se bilo kojim korisničkim imenom možemo prijaviti u sustav pomoću lozinke *„superpass“*. Ova postavka mora biti isključena u radnome okruženju te nužno postoji samo u razvojne svrhe. Nadalje, kroz konfiguracijski objekt dolazimo do objekta *database* koji omogućuje aplikaciji povezivanje na bazu podataka. Preciznije rečeno, u ovome objektu nalaze se podaci za povezivanje na bazu podataka. Povezivanje na bazu podataka već smo objasnili u ovome radu. Kao posljednje svojstvo ovog objekta dolazi svojstvo *phpmyadmin* koje sadrži poveznicu na *phpMyAdmin* aplikaciju. Ovakav način rada web aplikacije omogućuje administratoru da u budućnosti, ako bude potrebe, promjeni podatke konfiguracije za drugačiji način rada i upotrebe ove web aplikacije. Primjer ovoga *JSON* objekta može se vidjeti na sljedećoj slici.



Slika 23. Datoteka: /home/pi/TSD-TV/node.js/config/config.json

3.6.2 Biblioteke za Node.js

Kao i svakoj drugoj aplikaciji i ovoj su potrebne vanjske biblioteke. Vanjske biblioteke olakšavaju programeru rad u nekom programskom rješenju. Bez njih bilo bi gotovo nemoguće raditi aplikacije jer bi jedan čovjek tj. programer trebao biti detaljno upućen u hardverski rad računala. Budući da je gotovo nemoguće imati toliko znanja, biblioteke nam omogućuju da se s manje znanja možemo služiti naprednijim mogućnostima računala. Bez njih, kod bi bio suviše komplekasn i dugačak te nerazumljiv i kompliciran za modificiranje. Ova web aplikacija primarno koristi biblioteku pod nazivom *Express.js* za obradu HTTP zahtjeva na web aplikaciju. Međutim, *Express.js* nije jedina biblioteka koja se koristi u ovome projektu te je zato u tablici 8. priložen popis svih korištenih biblioteka.

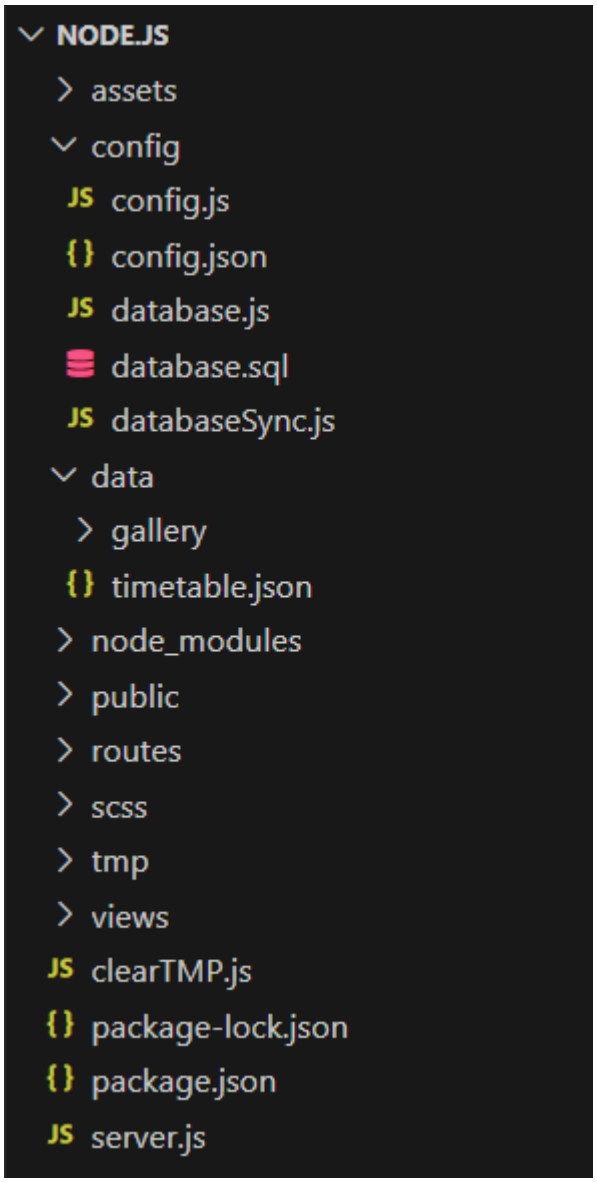
Tablica 8. Popis korištenih biblioteka za rad web aplikacije

Popis korištenih biblioteka za rad web aplikacije		
Naziv biblioteke	***npm* identifikator**	**Opis**
Bcrypt	bcrypt	Olakšava HASH-iranje lozinki
Body parser	body-parser	*Node.js* među program za analizu tijela (engl. *body*)
Bootstrap	bootstrap	Elegantan, intuitivan i moćan *front-end* okvir za brži i lakši razvoj web aplikacija
Check disk space	check-disk-space	Jednostavan višeplatformski alat za provjeru prostora na disku bez softvera treće strane za *Node.js*
Node.js child_process	child_process	Omogućuje komunikaciju s operacijskim sustavom
Embedded JavaScript templates	ejs	Izrada predložaka koji omogućuju generiranje dinamičkoga HTML-*a* pomoću HTML oznaka običnim *JavaScript-om*
express	express	Brz, samostalan, minimalistički web okvir za *Node.js*
Express fileupload	express-fileupload	Jednostavan dodatak *express.js*-u za učitavanje tj. prijenos datoteka

Morgan	morgan	Posrednički softver za zapisnik HTTP zahtjeva za *Node.js*
MySQL	mysql	Povezivanje i izrada upita za bazu podataka asinkronom vezom
Path	path	Jednostavno i brzo rješenje za upravljanje i uređivanje datotečnih putanja
Sass	sass	Čista *JavaScript* implementacija Sass-a
Sync MySQL	sync-mysql	Povezivanje i izrada upita za bazu podataka sinkroniziranom vezom

3.6.3 Organizacija koda

Izborni kod ove web aplikacije nije jednostavan, vrlo je složeno strukturiran i organiziran. Kao što je već napomenuto, centralni je dio i početak ove web aplikacije datoteka *server.js* koja se nalazi na vrhu mape web aplikacije: */home/pi/TSD-TV/node.js/.* Prva podmapa mape ove web aplikacije je mapa *assets*. Mapa *assets* u sebi sadrži *JavaScript* skriptu biblioteke *jQuery* i *Bootstrap,* a također sadrži i *CSS* skripte *Bootstrepa.* Nadalje, dolazi podmapa pod nazivom *config*. Ova mapa u sebi sadržava *JSON* konfiguracijski objekt web aplikacije te *JavaScript* skriptu *condfig.js* koju web aplikacija poziva za pristup konfiguracijskom objektu.



Slika 24. Organizacija koda web aplikacije

Podmapa također sadrži dvije datoteke *database.js* i *databaseSync.js* koje imaju ulogu povezivanja web aplikacije s bazom podataka prema konfiguraciji iz *JSON* objekta u datoteci *config.json*. Sljedeća podmapa naše web aplikacije jest mapa *data*. Ova podmapa ima mapu *gallery* u kojoj su spremljene slikovne datoteke za vizualni sadržaj koji se prikazuje na televizoru. Slijedi podmapa *node_modules* koja sadržava sve biblioteke koje su već navedene, kako bi omogućila rad ove web aplikacije. Nadalje dolazi podmapa *public* koja omogućuje da se datoteke unutar te mape mogu dohvatiti preko HTTP upita na web aplikaciju. Zatim slijedi podmapa *routes.* Ova podmapa sadržava sve skripte u programskom jeziku *JavaScript* koje odgovaraju na određene HTTP upite. Tako, na primjer, kada se napravi HTTP upit *„http://172.25.26.65:80/“* na web aplikaciju, poziva se skripta *GET_index.js* koja odgovara na taj upit. Sljedeća podmapa je mapa pod nazivom *scss*. U ovoj podmapi nalazi se *Scss* kod koji modificira *Bootstrap* tako mijenjajući tematsku boju kako bi odgovarala već postojećem dizajnu stranice škole. Tako dolazim i do predzadnje podmape *tmp* u kojoj se spremaju privremene datoteke. Biblioteci *express-fileupload* potrebna je ova mapa kako bi privremeno tijekom obrade HTTP zahtijeva spremila učitanu datoteku. Posljednja podmapa u web aplikaciji je mapa *views.* U ovoj podmapi nalazi se *HTML* tj. *EJS* stranice tj. *front-end* kod za konstrukciju web sučelja. Na kraju ostaju datoteke *package-local.json* i *package.json* koje su potrebne samo zbog instalacije i rada biblioteka u ovoj web aplikaciji, a koje instalira *npm* modul.

3.6.4 Primjer koda i objašnjenje

3.6.4.1 Primjer koda „Dohvaćanje *index* stranice“

Kako bi u *Node.js*-u dohvatili *index* stranicu preko poveznice *http://172.25.26.65:80/* potrebno je najprije *express.js*-u navesti da sluša na „/“ *HTTP* zahtjeve na *Node.js* poslužitelja. Ova logika primjenjuje se na dohvaćanje bilo koje stranice poslužitelja, a kao primjer slijedi slika 25. na kojoj je prikazan izvorni kod za dohvaćanje *index* stranice (29. linija koda).

```
28
29    tv.get('/', require_path('/routes/GET_index.js'))
30    tv.get('/impressum', require_path('/routes/GET_impressum.js'))
31    tv.get('/tv', require_path('/routes/GET_tv.js'))
32
```

Slika 25. Primjer koda: dohvaćanje *index* stranice web aplikacije

Za obradu zahtjeva poziva se kod iz skripte *GET_index.js*. Kao sve obrade zahtjeva i ova obrada vrši se u posebnoj skripti. Na sljedećoj slici vidljiva je skriptu *GET_index.js*.

```
1  const fs = require('fs')
2
3  module.exports = (req, res)=>{
4      res.render('index.ejs', {config:JSON.parse(fs.readFileSync('config/config.json', {encoding:'utf-8'}))})
5  }
```

Slika 26. Primjer koda: skripta GET_index.js

Navedena skripta vrlo je jednostavna te prema tome idealna kao primjer. Ova skripta bazira se na tome da na zahtjev odgovara sa *index.ejs* predloškom. Predlošku je potreban konfiguracijski objekt kako bi bio ispravno prikazan. Tako navedeni kod predlošku prvobitno dohvaća konfiguracijski objekt spremljen u *config.json* JSON objektu. Nakon obrađenog i postavljenog predloška prema zahtjevu korisnika, poslužitelj korisniku šalje odgovarajući HTTP odgovor.

3.6.4.2 Primjer koda „biblioteka web aplikacije: *database.js*“

Ova aplikacija uz svoje biblioteke od strane *Node.js*-a ima i dodatne biblioteke prilagođene samoj aplikaciji. Trenutno ova web aplikacija ima tri vlastite biblioteke koje se često koriste kao kod i spremljene su u podmapi *config*. Dvije od te tri biblioteke su *database.js* i *databaseSync.js*. Navedene biblioteke omogućuju u kodu web aplikacije mogućnost povezivanja i vršenja upita na bazi podataka. Treća i posljednja biblioteka jest *config.js* i služi za dohvaćanje i proširivanje konfiguracijskog objekta spremljenog u datoteci *config.json,* što je već ranije pojašnjeno u ovom dokumentu. Kao primjer uzet ćemo datoteku *database.js* koja će kodu web aplikacije davati asinkronu vezu s bazom podataka, što se može vidjeti na slici 27.

```
1   const mysql = require('mysql')
2   var link = mysql.createConnection(require('./config.json').database)
3
4   link.connect(function(err) {
5       if (err) throw err;
6       console.log("Connected!");
7   })
8   link.on('error', function(err) {
9       console.log("[mysql error] : ",err);
10      link = mysql.createConnection(require('./databaase.js').webnet)
11      link.connect(function(err) {
12          if (err) throw err;
13          console.log("Connected!");
14      })
15      process.exit()
16  });
17
18  module.exports = link
```

Slika 27. Primjer koda: skripta biblioteke *database.js*

Navedeni kod prvobitno deklarira varijablu *link* koja predstavlja vezu s bazom podataka. Kako bi deklarirali navedenu varijablu, koristimo eksternu biblioteku *mysql* koja je objašnjena u prethodnom poglavlju. Nadalje, izvorni kod pomoću varijable *link* izvršava povezivanje s bazom

podataka te je na kraju izvozi sa svrhom da je drugi kod može pozvati u slučaju potrebe za povezivanjem s bazom podataka.

3.6.4.3 Primjer koda „*front-end* sučelje, galerija“

Sva korisnička sučelja rade na sličan način, no u primjeru je objašnjeno kako radi sučelje za galeriju aplikacije. Nakon što poslužitelj obradi zahtjev korisnika šalje mu HTML s kodom korisničkog *front-end* sučelja. Svaka stranica, kao i ova, ima jednu glavnu ili par glavnih funkcija koje postavljaju stranicu. Jedna od glavnih funkcija koje postavljaju stranicu galerije jest funkcija *printImages()* čiji je izvorni kod prikazan na slici 28.

```
317  function printImages(page = 1) {
318      if (page < 1) page = 1
319      if (page == 1)$('#image-list')[0].innerHTML = ''
320      $('#finishUpload')[0].style.display = 'none'
321      $.ajax({
322          url:'/api/gallery',
323          method:'POST',
324          data:{
325              username:'<%= admin.username %>',
326              password:'<%= admin.password %>',
327              action:'get',
328              page:page,
329          },
330          success: (data) => {
331              data.forEach(e => {
332                  uiid = Math.random().toString(36).substring(2, 10) + Math.random().toString(36).substring(2, 10)
333                  $('#image-list')[0].innerHTML += `<div class="card" style="width: 100%;max-width: 500px;" id="uiid-${uiid}">
334                          <img src="/data/gallery/${e.fileName}" class="card-img-top border border-primary border-2" style="margin: 10px;width:
335                          <div class="card-body">
336                              <button class="btn btn-primary" onclick="editImage(${e.id})" data-bs-toggle="offcanvas" data-bs-target="#editImag
337                              <button class="btn btn-danger" onclick="deleteImage(${e.id}, 'uiid-${uiid}')">Obriši <i class="bi bi-trash3-fill"
338                          </div>
339                      </div>`
340              });
341              if (data.length >= 40) {
342                  $('#loadMoreImate')[0].style.display = 'block'
343                  $('#loadMoreImate')[0].onclick = () => {printImages(page+1)}
344              }else{
345                  $('#loadMoreImate')[0].style.display = 'none'
346                  $('#loadMoreImate')[0].onclick = () => {printImages(page)}
347              }
348          }
349      })
350  }
351
352  function tagImageUpload(index, tag_id, element_id) {
353      if (typeof tags_uploadImage[index] != "object") tags_uploadImage[index] = []
```

Slika 28. Primjer koda: funkcija *printImages()* u datoteci *admin-gallery.ejs*

Glavni dio funkcije jest AJAX zahtjev programskom sučelju (engl. API) aplikacije za dohvaćanje objekata koji će funkciji dohvatiti popis slika u web aplikaciji. Odgovor u obliku objekta sprema se u varijablu *data*. Na kraju funkcija prolazi kroz dohvaćeni objekt te umeće

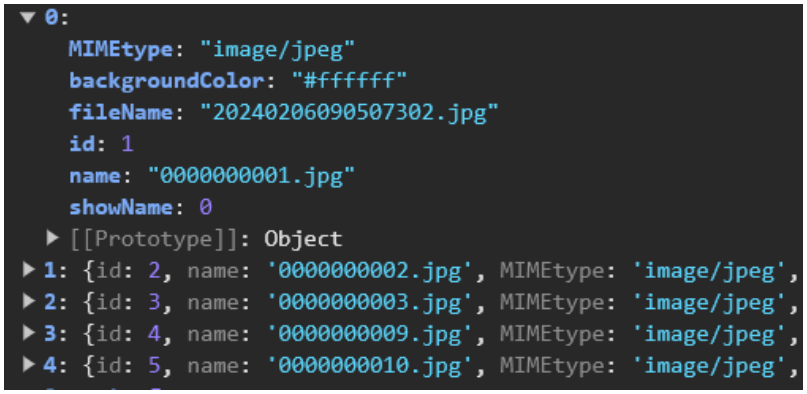
HTML kod u sučelje aplikacije. Nakon izrađene funkcije, na sučelju će se prikazati maksimalno 40. slika te gumb za učitavanje dodatnih slika ako postoje u zapisu baze podataka.

3.6.4.4 Primjer koda „Upit na API, dohvaćanje slika“

Kako bi sučelje moglo komunicirati s aplikacijom, potrebno je imati aplikacijsko sučelje (engl. *API*). U ovoj aplikaciji programsko sučelje radi na sličan način kao i dohvaćanje sučelja web aplikacije. Jedina razlika je u tome što je HTTP zahtjev na poslužitelj složeniji, kao i odgovor servera, koji je zapravo JSON objekt umjesto HTML koda. Na sljedećoj slici vidljiva je obrada zahtjeva koja se odnosi na upit za dohvaćanje slika iz baze podataka. Slika 29. prikazuje izvorni kod dijela skripte, odnosno datoteke: *POST_api-gallery.js*.

```
76  else if (req.body.action == 'get') {
77      let page = req.body.page;
78      database.query(`SELECT * FROM image LIMIT 40 OFFSET ${(page-1)*40}`, (error, data) => {
79          res.json(data)
80      })
81      return
82  }
```

Slika 29. Primjer koda: dio skripte POST_api-gallery.js odgovoran za dohvaćanje slika



Slika 30. Primjer odgovora na upit programskog sučelja web aplikacije

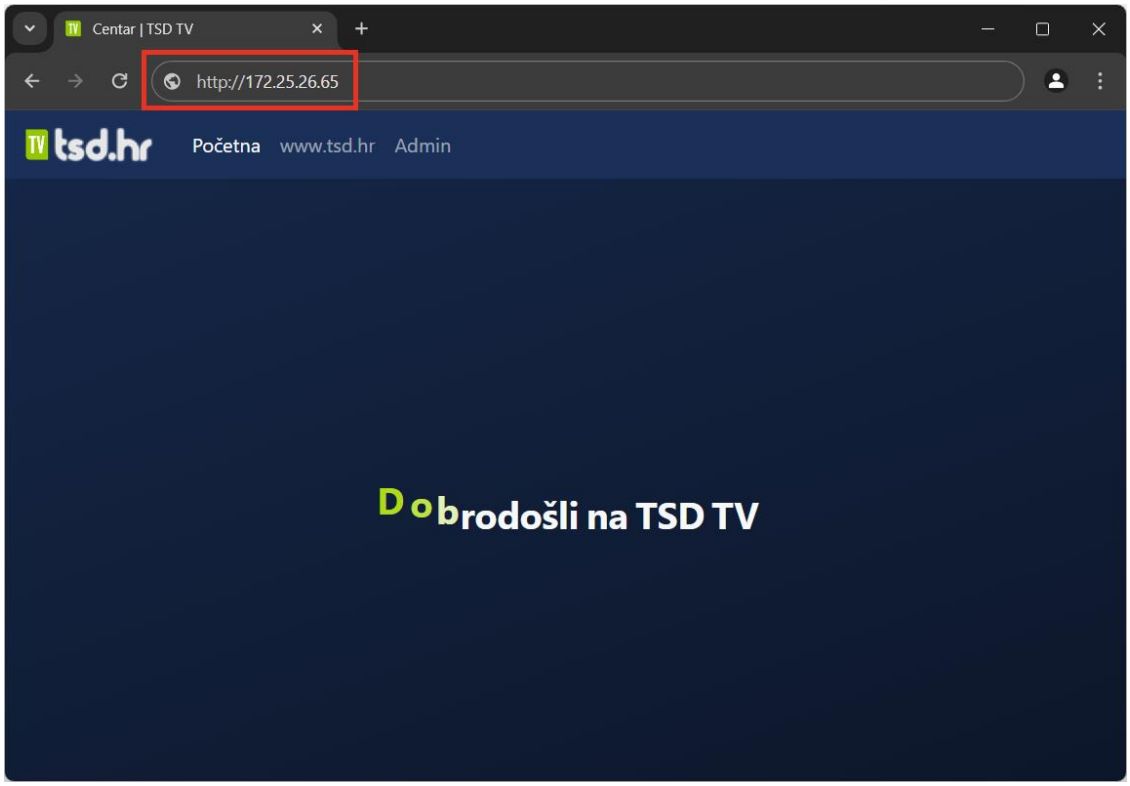
Upit ovom programskom sučelju jest HTTP POST zahtjev sa sljedećim podacima: *username*, *password*, *action* i *page*. Svojstva *username* i *password* potrebna su za autorizaciju korisnika tijekom obrade zahtjeva. Svojstvo *action* u ovom slučaju postavljamo na vrijednost „*get*“ kako bi dohvatili slike. Posljednje svojstvo upita jest *page* koje govori koju stranicu je potrebno dohvatiti. Nakon što smo poslali sve ispravne podatke poslužitelj vrši upit bazi podataka te dobiveni odgovor šalje natrag. Odgovor u ovom primjeru po vrsti je objekt, preciznije polje objekta, gdje svaki

objekt predstavlja jednu sliku tj. jedan zapis u tablici baze podataka. Primjer odgovora može se vidjeti na slici 30. Na ovakav i sličan način rade i sva ostala programska sučelja.

4 Upute za rad web aplikacijom

4.1.1 Kako pristupiti?

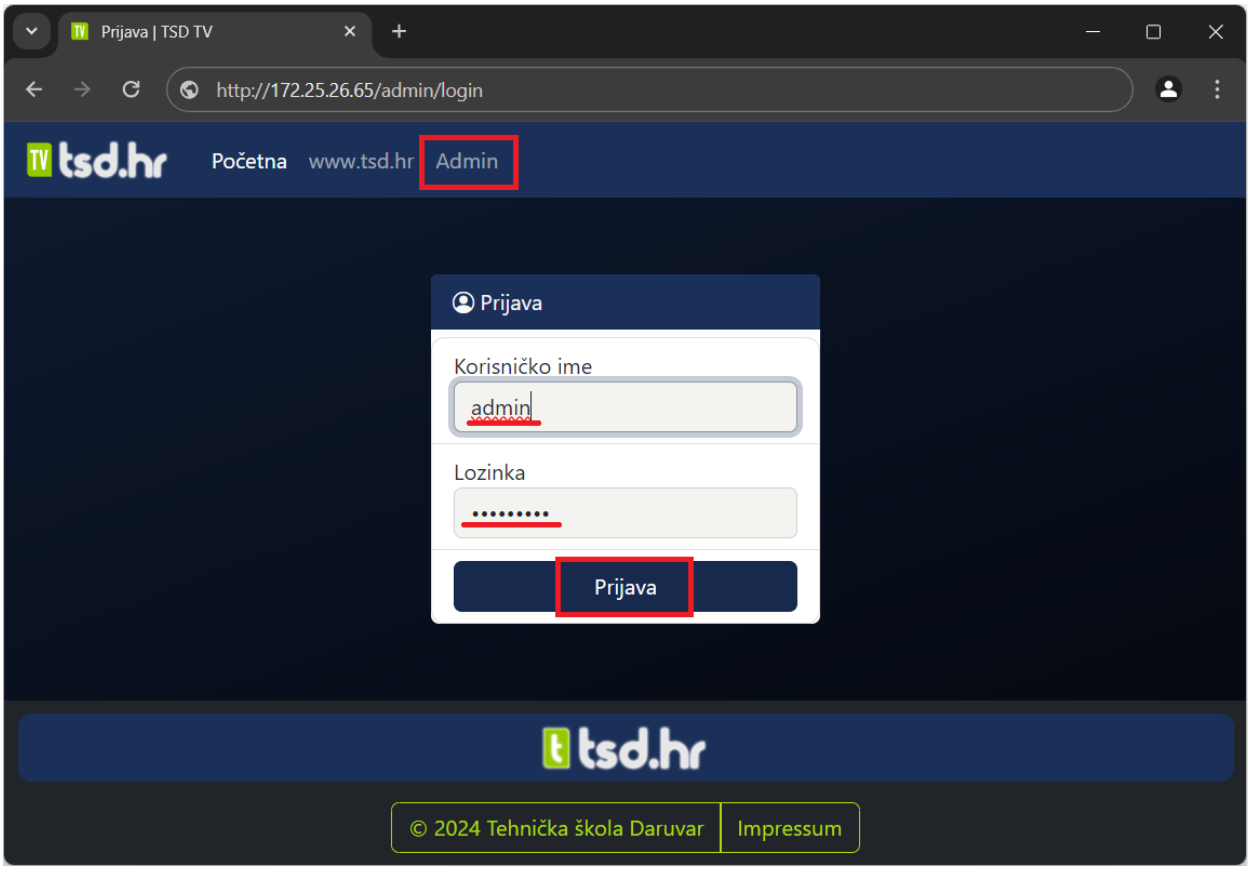
U ranijim poglavljima već je napomenuto da se web aplikaciji može pristupiti samo preko školske mreže. Prvi korak koji je potrebno poduzeti da bi se aplikaciji uopće pristupilo je spajanje na školsku mrežu. Pristup školskoj mreži ima svako računalo u vlasništvu Tehničke škole Daruvar. Točnije, svako računalo koje na sebi ima pristup AD (engl. *Active directory*) sustavu škole. Ako se na računalo možete prijaviti preko školskog računa, onda možete pristupiti i ovoj web aplikaciji. Nadalje potrebno je obaviti prijavu na računalu te otvoriti internetski preglednik poput *Google Chrome-a, Microsoft Edge-a, Mozzile FireFox* ili slično. U preglednik je potrebno upisati poveznicu „*http://tv.tsd.hr*“ ili „*http://172.25.26.65/*“. Nažalost, zbog tehničkih mogućnosti školske mreže, trenutno nije moguće konfigurirati domenu *tv.tsd.hr,* no u skoroj budućnosti to možda bude slučaj. Nakon što se upiše navedena adresa, otvorit će se web aplikacija kao što je prikazano na sljedećoj slici.



Slika 31. Pristup web aplikaciji preko web preglednika

4.1.2 Prijava u web aplikaciju

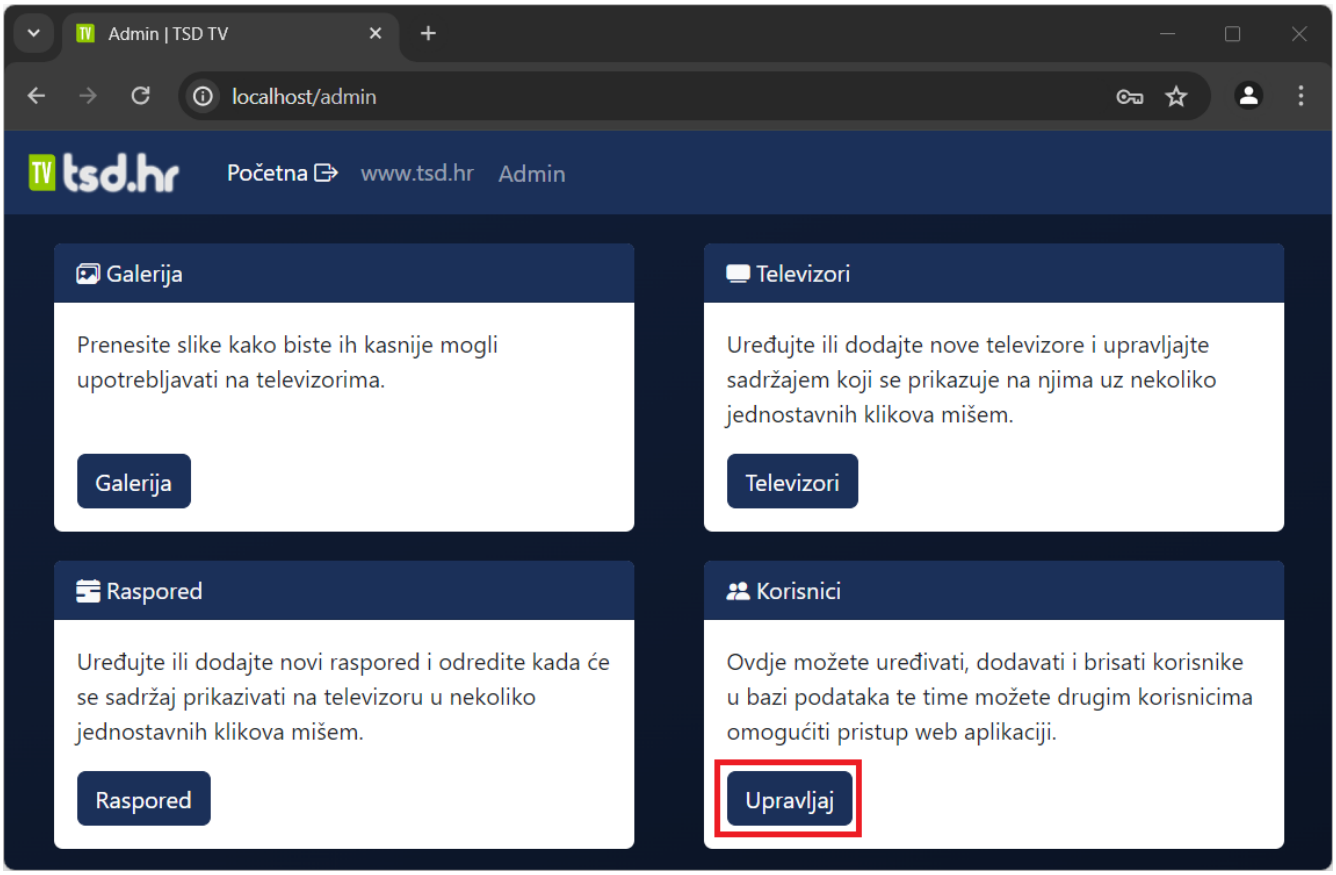
Kako bi bila omogućena administracija podataka unutar web aplikacije, potrebno je kliknuti na poveznicu *Admin*, koja vodi do stranice za prijavu. Kada se pritiskom na poveznicu *Admin* dođe na web stranicu adrese *http://172.25.26.65/admin/login*, potrebno je unijeti podatke u web formu za prijavu te nakon unosa podataka kliknuti na gumb *Prijava.* U primjeru je prikazana prijava kao korisnik *admin.* Ako se ispravno unesu podaci otvorit će se početna kontrolna ploča.



Slika 32. Prijava na web aplikaciju

4.1.3 Upravljanje korisnicima

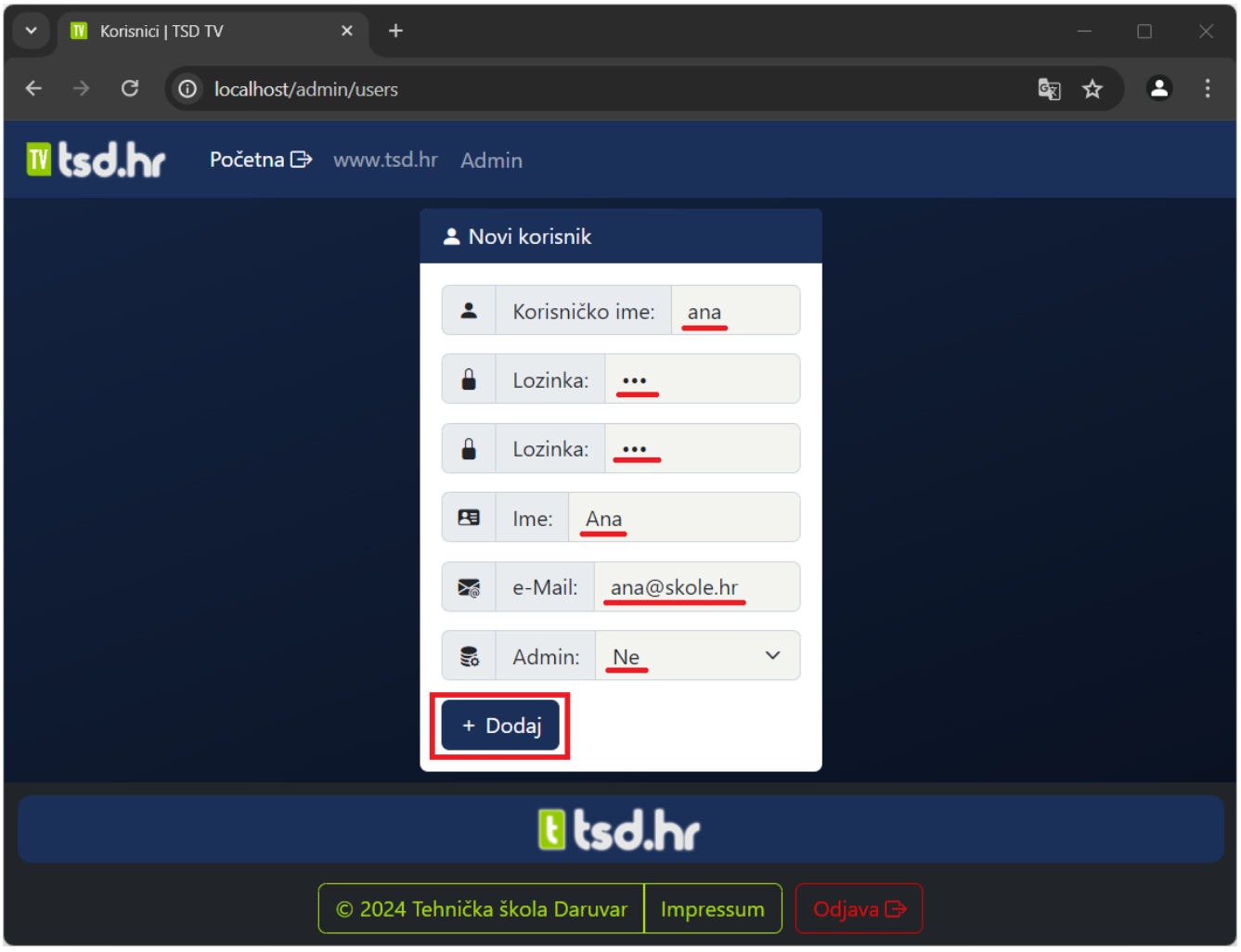
Nakon uspješne prijave dolazi se na glavnu stranicu kao što je prikazano na slici 33. Da bi se došlo do sučelja za upravljanje korisnika, dovoljno je pritisnuti gumb *Upravljaj* na kartici *Korisnici*.



Slika 33. Kontrolna ploča administratora na web aplikaciji

4.1.3.1 Dodavanje korisnika

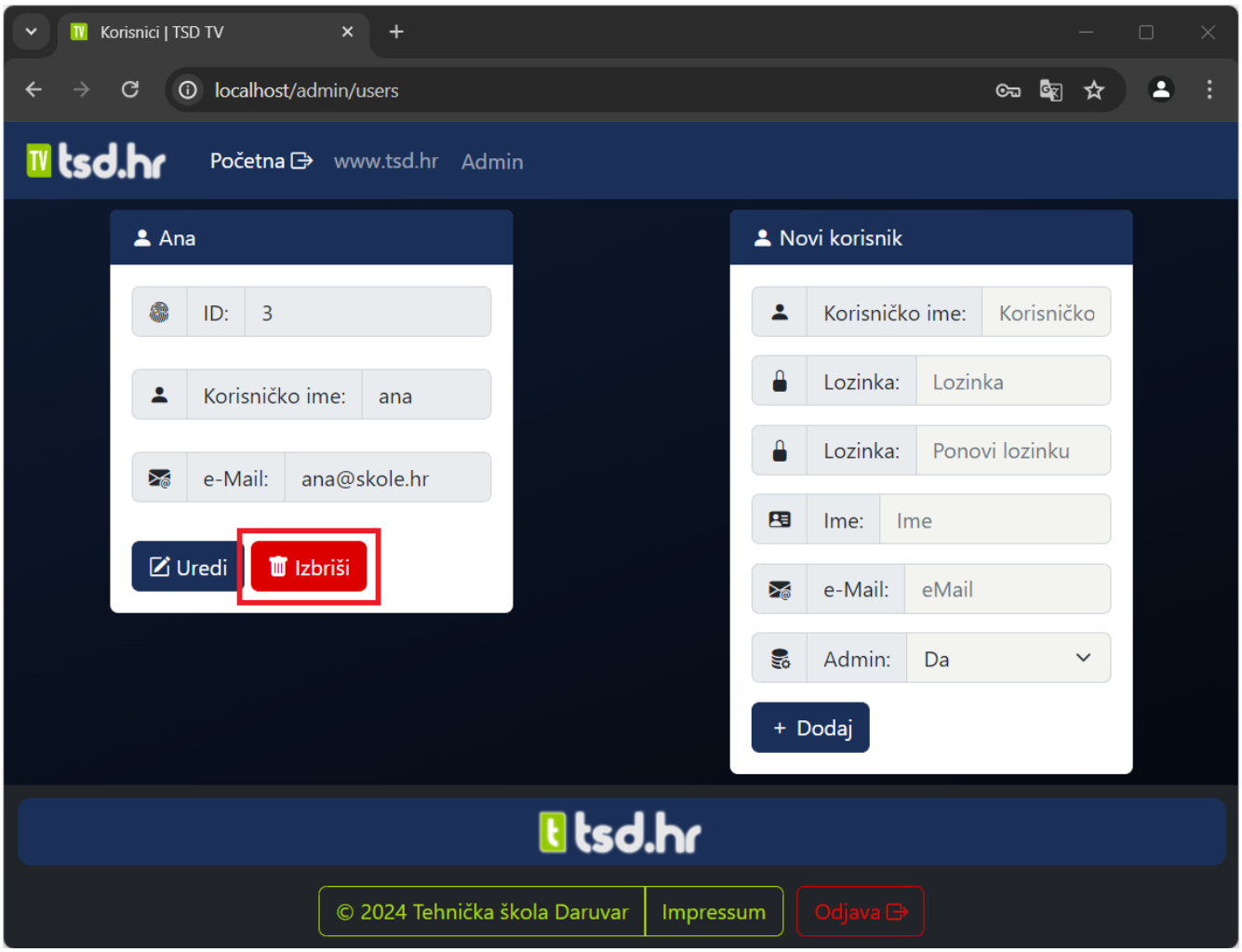
Na stranici za upravljanje korisnikom, u formu prikazanu na slici 34. potrebno je unijeti podatke o korisniku kojeg želimo dodati u sustav. Na primjer, dodaje se korisnica Ana pod korisničkim imenom *ana*. Nakon što popunimo formu potrebno je pritisnuti gumb + *Dodaj* te će korisnik biti unesen u sustav.



Slika 34. Dodavanje novog korisnika u sustav web aplikacije

4.1.3.2 Brisanje korisnika

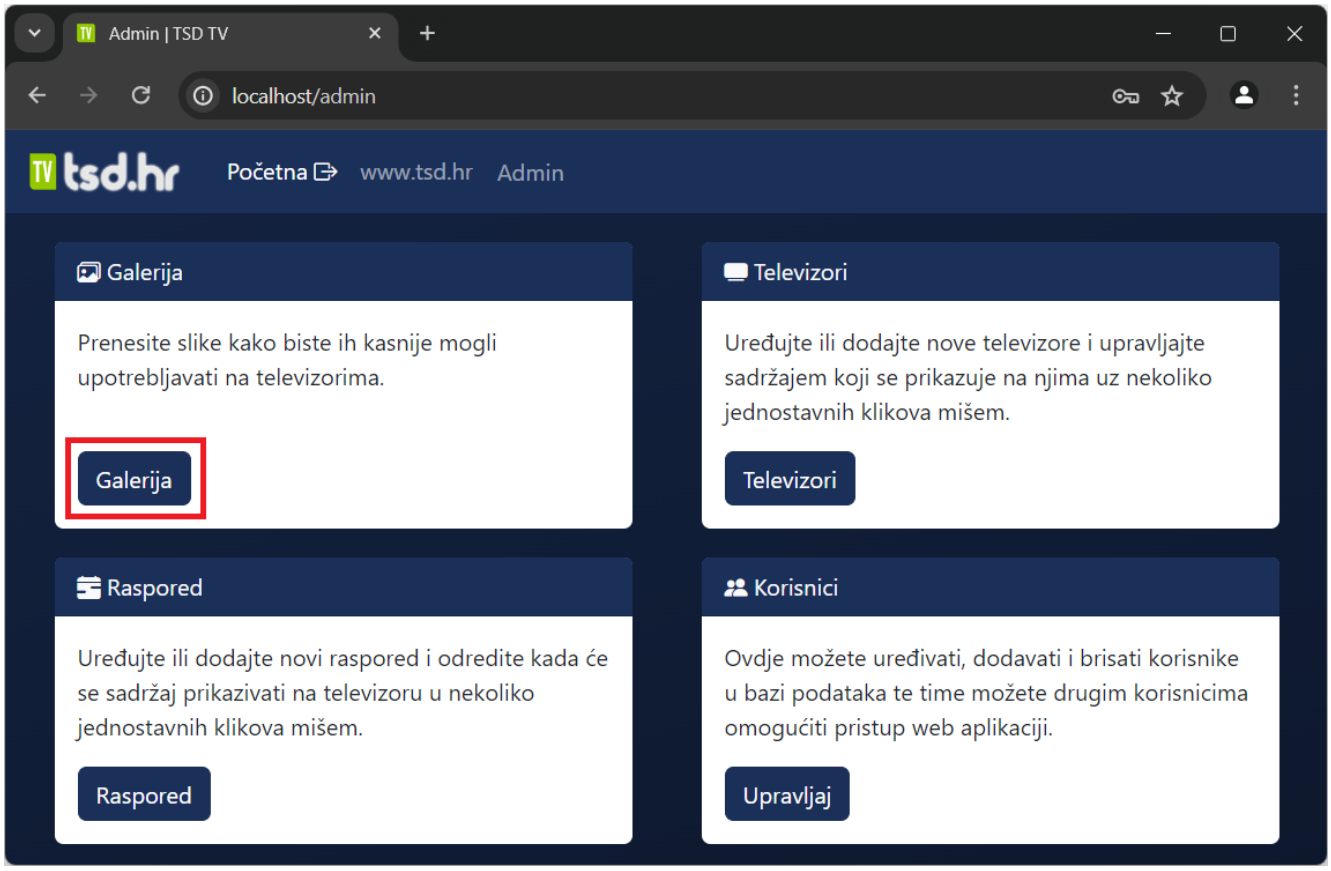
Nakon uspješnog dodavanja korisnika u sustav, sljedećim primjerom prikazuje se brisanje istog korisnika iz sustava. Kako bi korisnika izbrisali iz sustava dovoljno je na kartici samoga korisnika pritisnuti gumb *Izbriši*. U sljedećem primjeru obrisat ćemo korisnicu pod korisničkim imenom *ana*, što je prikazano na slici 35.



Slika 35. Brisanje korisnika iz sustava web aplikacije

4.1.4 Upravljanje slikama

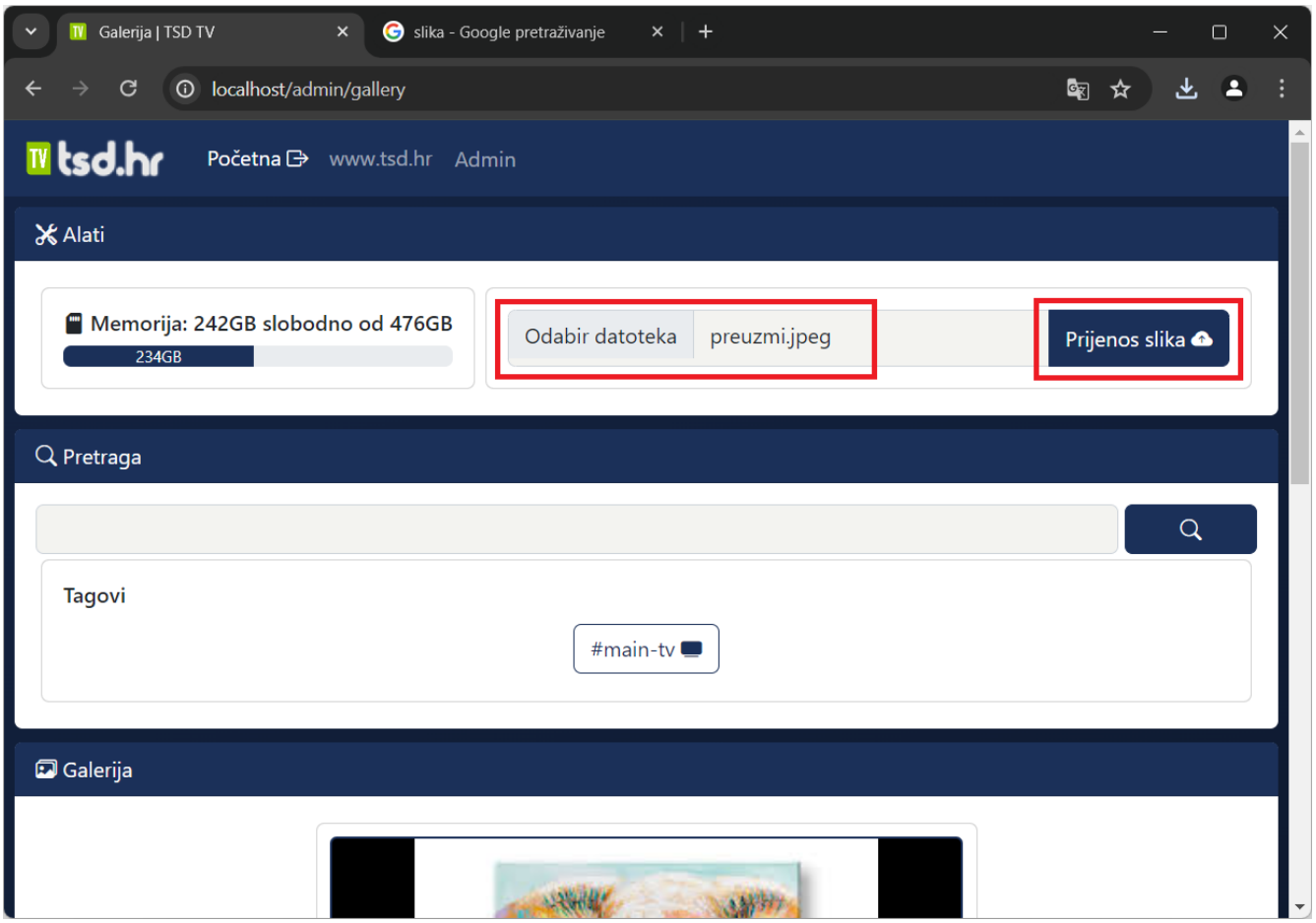
Kako bi se upravljalo vizualnim sadržajem na televizoru tj. slikama, potrebno je vratiti se na početnu kontrolnu administratorsku ploču ili se prijaviti u aplikaciju i doći do nje. Nakon toga potrebno je klikom na gumb *Galerija* doći do upravljačkog sučelja za upravljanje vizualnim sadržajem, kao što je prikazano na slici 36.



Slika 36. Kontrolna ploča administratora na web aplikaciji

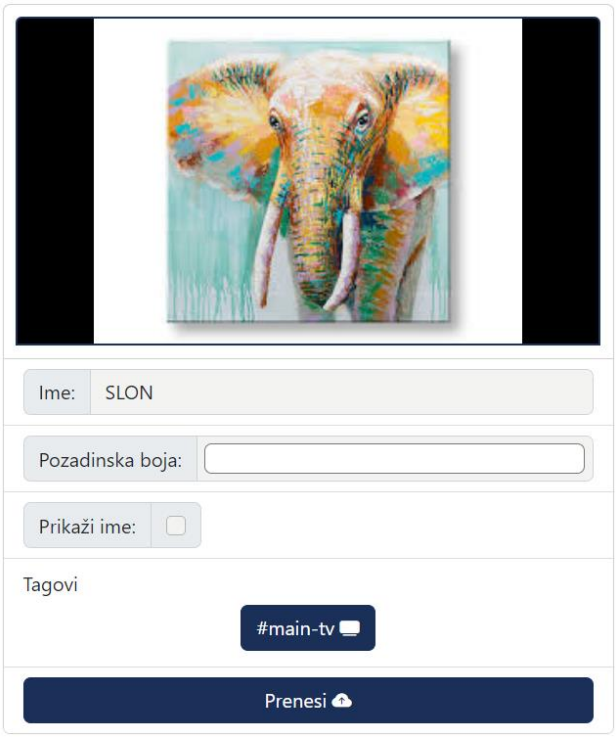
4.1.4.1 Dodavanje slike

Kako bi prenijeli sliku na web aplikaciju, pa tako i na sam televizor, najprije je potrebno klikom na *Odabir datoteka* odabrati željene slike za prijenos, a nakon toga potvrditi odabrane datoteke klikom na gumb *Prijenos slika*. Slika će se zatim prikazati u galeriji, što se može vidjeti na slici 37.



Slika 37. Odabir slike za prijenos u sustav web aplikacije

Nadalje, nakon što je odabrana jedna ili više slika, potrebno je za svaku sliku prilagoditi postavke i prenijeti je u sustav. Nakon uređivanja forme potrebno je na kraju klikom na *Prenesi* prenijeti sliku u sustav. Kako bi se slika prikazivala na televizoru svakako je potrebno označiti televizor na kojem se slika treba prikazati. Sve te postavke mogu se vidjeti na slici 38.



Slika 38. Prijenos slike u sustav web aplikacije

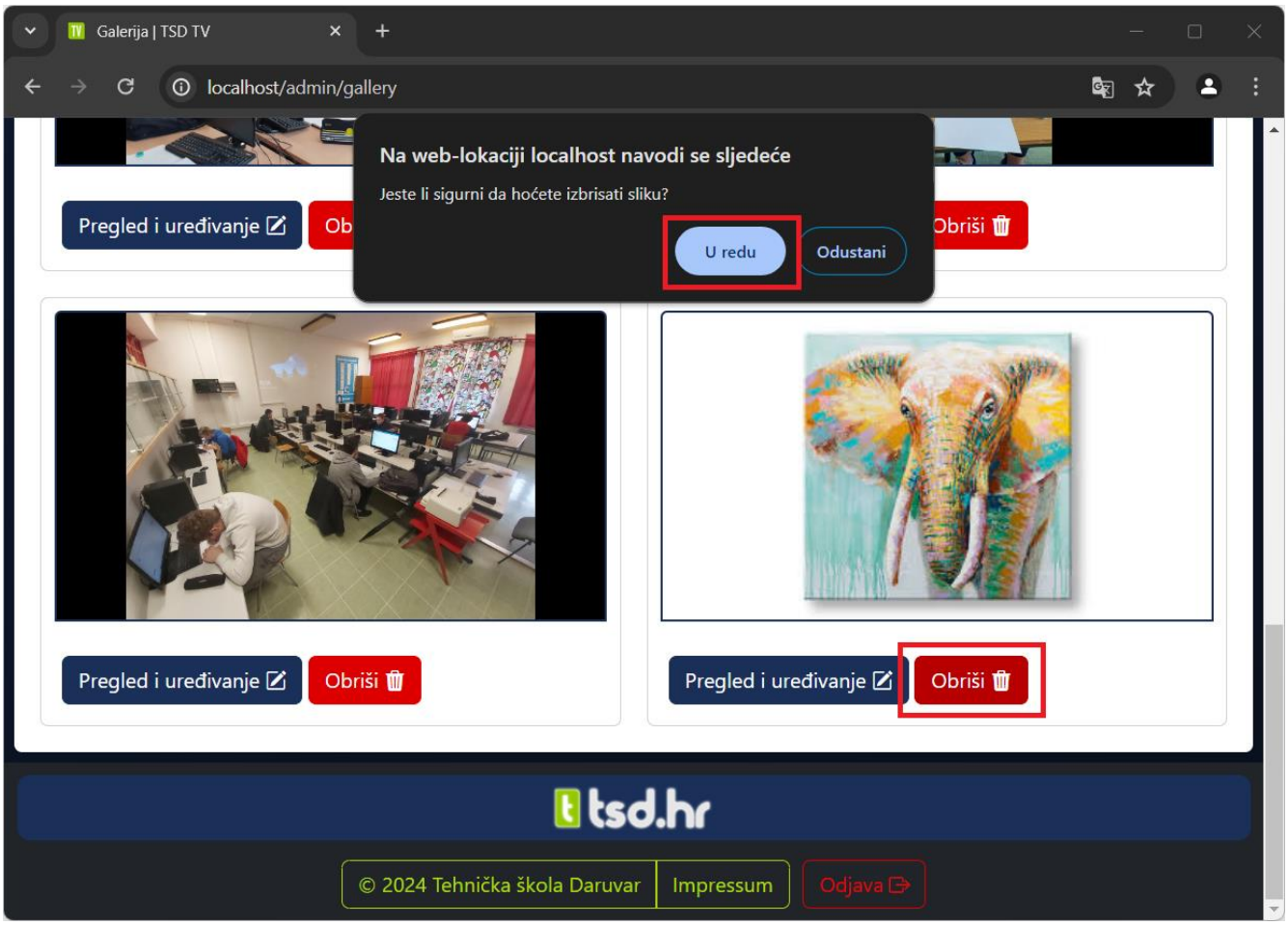
4.1.4.2 Uređivanje slike

Ako je slika već prenesena, a potrebno je naknadno uređivati, u galeriji je potrebno pronaći željenu sliku te se pritiskom na *Pregled i uređivanje* otvara grafičko sučelje za uređivanje odabrane slike. Ovaj korak prikazan je na slici 39. Nakon što uredimo podatke potrebno je na kraju pritisnuti gumb *Spremi* kako bi se promjene spremile.

Slika 39. Uređivanje slike u sustavu web aplikacije

4.1.4.3 Brisanje slike

Ukoliko je potrebno obrisati neku sliku iz sustava web aplikacije, potrebno je u galeriji pronaći te pritisnuti na gumb *Obriši* i dodatno potvrditi akciju brisanja pritiskom gumba *U redu*. U ovom primjeru prikazano je brisanje slike *SLON* koja je prenesena i uređena u sustavu web aplikacije u prethodnom primjeru. Sam postupak brisanja prikazan je na slici 40.



Slika 40. Brisanje slike iz sustava web aplikacije

5 Zaključak

Na kraju dugotrajnog istraživanja tehnologija te napornog truda i rada dobio sam funkcionalnu aplikaciju koja će znatno olakšati rad osoblju škole u budućnosti. Djelatnici škole koji će biti zaduženi za upravljanje sadržajem na televizoru moći će samostalno uređivati vizualni sadržaj. Web aplikacija ne služi samo za prikaz vizualnog sadržaja, poput fotografija, nego je pomoću nje omogućeno upravljanje obavijestima poput promjena u rasporedu i slično. Ono što me posebno čini zadovoljnim jest što sam proširio dosadašnje mogućnosti, od jednostavnog prikazivanja slika do interaktivnog sučelja koje pruža obavijesti važne za svakodnevno funkcioniranje škole. Korištenjem Raspberry Pi-ja od običnog sam televizora napravio sučelje za vizualni prikaz sadržaja koje je jednostavno za administraciju. Rad aplikacije je samo-održiv i aplikacija se uvijek pokreće s Raspberry Pi-jem, bez dodatnih poteškoća. Rezultat je samoodrživa i stabilna web aplikacija na Linux poslužitelju.

6 Podaci za prijavu

Tablica 9. Podaci za prijavu na Raspberry Pi

Raspberry Pi OS i SSH	
Korisničko ime	********
Lozinka	********

Tablica 10. Podaci za prijavu administratora na web aplikaciju

Web Stranica	
Korisničko ime	********
Lozinka	********
Poveznica	********

Tablica 11. Podaci za prijavu u web aplikaciju phpMyAdmin

phpMyAdmin	
Korisničko ime	********
Lozinka	********
Poveznica	********

Tablica 12. Podaci za povezivanje na FTP poslužitelj

FTP protokol	
Korisničko ime	********
Lozinka	********
Adresa	********
Port	********
SSL Protokol	********

7 Životopis autora

Tablica 13. Osobni podaci

Osobni Podaci	
Ime i Prezime	Alexander Gustovarac
Adresa	********
Telefon	********
E-mail	alexander.gustovarac@skole.hr
Datum rođenja	********
Školovanje	Osnovna škola braće Radića, Pakrac; Osnovna glazbena škola Pakrac, Pakrac; Tehnička škola Daruvar, Daruvar; Glazbena Škola Brune Bjelinskog, Daruvar
Smjer	Tehničar za računalstvo, Glazbenik klavirist

8 Popis slika

9 Popis tablica

10 Literatura

- Frančić, I. (Listopad 2021). Izrada web stranica pomoću Bootstrap 5 komponenti. Varaždin, Hrvatska: Sveučilište Sjever.
- GNOME. (n.d.). *GNOME*. Dohvaćeno iz https://apps.gnome.org/hr/Epiphany/
- Naglić, K. (n.d.). Razvoj Web aplikacije u programskom jeziku Node.js. Varaždin, Hrvatska: Sveučilište u zagrebu, Fakultet organizacije i informatike.
- Topolovec, F. (Listopad 2015). Web sustav upravljanja bazom podataka MySQL. Varaždin, Hrvatska: Sveučilište Sjever, odjel za multimediju, oblikovanje i primjenu.
- Vrbančić, A. (2020). RAZVOJ WEB APLIKACIJE U PROGRAMSKOM JEZIKU PYTHON. Varaždin, Hrvatska: Sveučilište u Zagrebu, Fakultet organizacije i informatike.
- Živković, N. (2017). Raspberry Pi. Osijek, Hrvatska: Sveučilište J.J. Strossmayera u Osijeku, Odjel za matematiku.