TEHNIČKA ŠKOLA RUĐER BOŠKOVIĆ

GETALDIĆEVA 4, ZAGREB

ZAVRŠNI STRUČNI RAD:

Izrada mrežnog sjedišta Servisa elektroničkih uređaja

MENTORICA: UČENIK: Leo Marušić  
mr. sc. Kristinka Maček Blažeka, mag. ing. el. RAZRED: 4.C

Zagreb, travanj 2022.

**SADRŽAJ**

[**1. UVOD** 1](#_Toc101914689)

[**2. TEHNOLOGIJE** 2](#_Toc101914690)

[**2.1** **Jezici** 2](#_Toc101914691)

[2.1.1 Programski jezici 2](#_Toc101914692)

[2.1.2 Markup jezici 3](#_Toc101914693)

[2.1.3 Skriptni jezici 3](#_Toc101914694)

[2.1.4 HTML 4](#_Toc101914695)

[2.1.5 CSS 5](#_Toc101914696)

[2.1.6 JavaScript 5](#_Toc101914697)

[2.1.7 PHP 6](#_Toc101914698)

[2.1.8 MySQL 7](#_Toc101914699)

[**2.2** **Integrirano razvojno okruženje** 8](#_Toc101914700)

[2.2.1 Phpstorm 8](#_Toc101914701)

[**2.3** **Server stack** 9](#_Toc101914702)

[2.3.1 XAMPP 9](#_Toc101914703)

[**2.4** **Verzija kontrole** 10](#_Toc101914704)

[2.4.1 Git 11](#_Toc101914705)

[2.4.2 GitHub 11](#_Toc101914706)

[**2.5** **Haikei** 12](#_Toc101914707)

[**3.** **Struktura i funkcionalnosti web sjedišta** 13](#_Toc101914708)

[**3.1** **Početna stranica** 14](#_Toc101914709)

[**3.2** **Cjenik** 16](#_Toc101914710)

[**3.3** **Funkcije** 17](#_Toc101914711)

[**3.4 Webshop** 18](#_Toc101914712)

[3.4.1 Struktura webshop-a 19](#_Toc101914713)

[3.4.2 Popunjavanje stavki 20](#_Toc101914714)

[3.4.3 Filtriranje i pretraživanje 21](#_Toc101914715)

[3.4.4 Dinamički upit 24](#_Toc101914716)

[3.4.5 Straničanje 30](#_Toc101914717)

[**3.5** **Stranica stavke** 33](#_Toc101914718)

[**3.6** **Login** 35](#_Toc101914719)

[**3.7** **Administracija** 37](#_Toc101914720)

[3.7.1 Modificirani webshop 37](#_Toc101914721)

[3.7.2 Dodavanje stavki 39](#_Toc101914722)

[3.7.3 Izmjena stavki 45](#_Toc101914723)

[3.7.4 Brisanje stavki 49](#_Toc101914724)

[**4.** **Zaključak** 50](#_Toc101914725)

[**5.** **Literatura** 51](#_Toc101914726)

[**6.** **Popis slika** 52](#_Toc101914727)

# **UVOD**

Danas se sve više stvari nalazi na *web*-u, od svake poslovnice očekujemo da ima svoju *web* stranicu na kojoj se mogu vidjeti relevantne informacije o poslovnici koja nas zanima. Ako poslovnica nema *web* stranicu ili je loše napravljena, to može jako utjecati na broj kupaca te poslovnice. Stoga je važno za svaku poslovnicu da uloži resurse za izradu *web* stranice kako bi pridobili kupce s inovativnim dizajnom i upotrebljivošću stranice.

Upravo zbog tog razloga, *web* programer je najpopularnije zanimanje među programerima. *Web* programere možemo podijeliti u tri skupine; *front-end* developeri, *back-end* developeri i *full-stack* developeri. *Front-end* developeri se obično bave onime što korisnik vidi, brinu se o izgledu stranice, korisnikovom iskustvu (*User Experience*) i izgledu onome što se događa na poslužiteljskom dijelu. Za njihov posao potrebno je dobro poznavanje jezika HTML (*HyperText Markup Language*), CCS (*Cascading Style Sheets*) i JS (*JavaScript*), te za naprednije projekte i poznavanje *framework*-ova poput *jQuery*, *React*, *Angular*… *Back-end* developeri su zaduženi za poslužiteljski dio *web* stranice, obično je to rukovanje korisničkim podatcima u obliku korisničke autentikacije i autorizacije, korisničkim zahtjevima u obliku upita i optimizacija poslužiteljskog sustava. Zbog manjka vizualnih elemenata i kompliciranosti njihovih zadataka, posao *back-end* *developera* se može smatrati najtežim dijelom izrade *web* stranice. *Full-stack developeri* na sebe uzimaju poslove *front-end* i *back-end developera*. Njihova velika svestranost ih čini vrlo poželjnima u svijetu programera te se obično još dodatno bave komunikacijom s klijentom.

Svrha ovog završnog rada je izrada mrežnog sjedišta gdje se jasno i lagano mogu pronaći informacije za poslovnicu koja se bavi popravkom, prikupljanjem i prodajom elektroničkih uređaja. Izabrao sam ovaj rad zbog toga što jedna takva poslovnica treba *web* stranicu, te bi ovaj rad mogao ispuniti taj zadatak. Sjedište se sastoji od: glavne stranice na kojoj su gotovo odmah vidljivi svi podatci koji bi zanimali potencijalnog kupca/klijenta, cjenika koji sadrži informacije o cijenama usluga servisa i *web* trgovine koja omogućuje pregled postavljenih uređaja ili komponenata. Vlasniku stranice omogućeno je dodavanje, brisanje i mijenjanje uređaja/komponenata koje se prikazuju u *web* trgovini.

# **TEHNOLOGIJE**

Pri izradi završnog rada, korišteni su različiti programski jezici, aplikacije, alati i pomagala kako bi se izradilo željeno *web* sjedište.

Ova cjelina nabraja i objašnjava koji su se programski jezici, alati i pomagala koristili pri izradi mrežnog sjedišta.

## **Jezici**

### 2.1.1 Programski jezici

Programski jezik je skup pravila s kojim ljudi mogu komunicirati s računalom preko zadavanja instrukcija. Mogu se podijeliti na pet generacija:

Programski jezici prve generacije – strojni kod u binarnoj formi, komunikacija s računalom preko probušenih kartica, papirnatih/magnetskih vrpca ili preklopnika na prednjoj ploči računala.

Programski jezici druge generacije – sklopovski (*assembly*) jezici, namijenjeni da budu puno više ljudski čitljivi, olakšali posao programera pri računanju adresa. Npr. COBOL, C, JavaScript…

Programski jezici treće generacije – jezici visoke razine, koriste jaku apstrakciju strojnog koda što omogućava pisanje koda normalnim jezikom, moguća kompletna automatizacija upravljanja memorijom. Npr. C, C++, C#, Python, Ruby…

Programski jezici četvrte generacije – vrlo slični programskim jezicima treće generacije s fokusom na baze podataka, izvješća i *web* stranice. Neki napredni jezici treće generacije poput Python, Ruby i Perl kombiniraju elemente treće i četvrte generacije, maskirajući razliku.

Programski jezici pete generacije – dizajnirani prema tome da računalu kažu koji problem treba riješiti bez potrebe za programerom. Koriste se u svrhe istraživanje umjetne inteligencije. Npr. OPS5 i Mercury.

### 2.1.2 *Markup* jezici

*Markup* jezik je skup pravila koja određuju koje *markup* informacije smiju biti uključene u dokument i kako se kombiniraju sa sadržajem dokumenta tako da se olakša upotreba od strane ljudi i računala. Markup jezici se fokusiraju na tipografiju, prezentaciju i identifikaciju komponenata dokumenta s očekivanjem da će tehnologije poput *stylesheets* napraviti oblikovanje sadržaja. Npr. HTML, XML, Tex, LaTex, Scribe…

### 2.1.3 Skriptni jezici

Skriptni jezik je programski jezik koji se izvodi za vrijeme izvršavanja, automatizira izvršavanje zadataka koje bi inače izvodio čovjek. Zbog čestog korištenja s drugim jezicima, nazivaju se i ekstenzijskim jezicima. Tipični skriptni jezici namijenjeni su da budu vrlo lagani za naučiti i pisati, razlog tome su relativno jednostavna sintaksa i semantika. Skriptni jezici se mogu podijeliti u više skupina, no uz mnogo preklapanja:

Glue jezici – koriste se kako bi povezali različite dijelove softvera, u *web* programiranju često se koriste kako bi se baza podataka povezala s *web* stranicom. Vrlo su korisni u slučajevima gdje želimo napraviti: prilagođene komande za ljusku, manje programe za koje sastavljeni jezici su neupotrebljivi, pripremu za izvršenje programa, mijenjajuće skripte, itd. Npr. JavaScript, PHP, Python, Ruby, Bash, PowerShell.

Jezici i ljuske za kontrolu poslova – kontrola ponašanja sistema, mnogi jezici su prerasli tu funkciju, no i dalje ju podržavaju. Mnogi ti jezici se koriste kao CLI (*Command Line Interpreter*) poput Unix ljuske ili COMMAND.COM.

GUI skriptiranje – njihov posao je simulacija korisnikovog *input*-a radi automatizacije, nazivaju se i „*macros*“ kada se aktiviraju pritiskom gumba ili miša. Namijenjeni su da rade s bilo kojom GUI aplikacijom, no u stvarnosti njihova iskoristivost je vrlo ograničena. Npr. OREXX.

Jezici za specifične aplikacije – jezici dizajnirani za pojedinu aplikaciju prema potrebi korisnika aplikacije. Npr. QuakeC, Emacs Lisp.

Ekstenzijski/Ugradbeni jezici – namijenjeni kao zamjena za jezike specifičnih aplikacija time što bi se ugradili u sami program u kojem je aplikacija. Njihova prednost je ta što programeri mogu prenositi svoje vještine s aplikacije na aplikaciju. Npr. JavaScript, Python, Lua, Tcl.

### 2.1.4 HTML

*HyperText Markup Language* ili HTML je standardni markup jezik za dokumente dizajnirane za prikaz na *web*-pregledniku. HTML elementi su temelj HTML stranica, određujući kakav će se sadržaj nalaziti u pojedinom elementu. Npr. <article>, <img>, <input>, <footer>, itd. Pošto HTML može samo odrediti strukturu i sadržaj dokumenta uz vrlo ograničenu mogućnost dizajna, gotovo nikada se ne koristi samo HTML, uz njega koriste se tehnologije poput *Cascading Style Sheets* i *JavaScript.* One omogućuju kompleksno oblikovanje HTML dokumenta i interakciju s korisnikom.

Trenutna i finalna verzija HTML-a je HTML5, preporučena od *World Wide Web Consortium* (W3C) i održavana od *Web Hypertext Application Technology Working Group* (WHATWG). HTML5 je kompletna nadogradnja na HTML4, nudeći puno više mogućnosti koje su prije trebale biti implementirane pomoću *Cascading Style Sheets* i *JavaScript*. Npr. Podrška za multimediju pomoću <video>, <audio> i <canvas> elemenata, proširivi elementi kao <summary> i <details>, podrška za SVG grafike i MathML za matematičke jednadžbe.

Graphical user interface, text, application

Description automatically generated

Slika 2.1 Primjer HTML koda i njegov rezultat

### 2.1.5 CSS

*Cascading Style Sheets* ili CSS je stilski listni jezik (*style sheet language)* koji se koristi za prezentaciju i dizajniranje dokumenta napisanog u *markup* jeziku poput HTML. Svojim mogućnostima nudi razdvajanje prezentacije i sadržaja pomoću kontrole nad rasporedom, bojama, fontovima, veličinama elemenata, događajima… Takvo razdvajanje nam nudi veću pristupačnost sadržaju i veću fleksibilnost i kontrolu nad prezentacijskim karakteristikama. U CSS-u se često koriste pravila koja vrijede na mobilnim uređajima kako bi osigurali da se stranica pravilno prikazuje na svim uređajima. Isto tako omogućava drukčije metode prikaza sadržaja poput; na ekranu, na papiru, pomoću glasa, na uređajima s Brailleovim pismom.

Text

Description automatically generated

Slika 2.2 Primjer CSS koda

Više web stranica mogu koristiti jednu CSS datoteku, što smanjuje kompleksnost i omogućava brže učitavanje stranica pošto se ista datoteka ne treba učitavati više puta.

### 2.1.6 JavaScript

*JavaScript* ili JS je programski jezik dizajniran za izradu *web* stranica uz HTML i CSS. Izvršava se na strani klijenta, služi za bolju interakciju s *web* stranicom, mnogi *web* preglednici imaju ugrađeni *JavaScript enigne* kako bi se kod mogao izvršiti. Preko 97% stranica koristi JavaScript ili neki JavaScript *framework* poput *jQuery*, *React*, *NextJS* ili *Angular* kako bi poboljšali interaktivnost i funkcionalnost *web* stranice. To je jezik visoke razine što znači da ima veliku apstrakciju od strojnog koda. Klasificira se kao programski jezik ali se može još nazvati i skriptnim jezikom zbog toga što se kod prevodi u strojni kod i izvršava pri izvršavanju aplikacije.

U *JavaScript*-u se ne moraju koristiti tipovi podataka pri deklariranju varijabli, njihovi tipovi će se dodijeliti kada se kod izvršava. Skoro sve u *JavaScript*-u je objekt, stoga je potrebno znanje objektno orijentiranog programiranja ako ga želimo razumjeti. Vrlo je svestran jezik, ima API-ove (*Application Programming Interface*) koji omogućuju rad s tekstom, datumima, pretragom podataka (*regex*), strukturama podataka i DOM (*Document Object Model*) modelom.

Text

Description automatically generated

Slika 2.3 Primjer JavaScript koda

### 2.1.7 PHP

PHP ili originalno *Personal Home Page*, naziv koji se danas ne koristi, je skriptni jezik opće namjene dizajniran za *web development*. Njegov kod se izvršava na poslužiteljskoj strani *web* stranice, često služeći kako bi se podatci iz baze podataka mogle prikazati na stranici. Njegov standardni prevoditelj je besplatni softver, napravljen na Zend Engine-u i omogućuje da PHP može biti implementiran na raznovrsnim operacijskim sustavima i platformama. Korišten je od 78.1% stranica čiji je serverski kod poznat.

Dodatno, PHP ima ostale mogućnosti izvan *web developmenta*, kao što su to izrada samostalne grafičke aplikacije, kontrole robotskih dronova ili izvršavanje unutar CLI.

Text

Description automatically generated

Slika 2.4 Primjer PHP koda

### 2.1.8 MySQL

MySQL je *open-source* sustav upravljanja relacijskim bazama podataka (RDBMS). Relacijske baze podataka su baze podataka čiji podatci u tablicama mogu biti povezani s podatcima drugih tablica. Pomoću SQL-a korisnici mogu komunicirati (pisati, brisati, modificirati) s MySQL bazom podataka na nekom operacijskom sustavu, uz to omogućava kontrolu korisnika, pristup mreži, olakšava testiranje integriteta baze podataka i stvaranje sigurnosnih kopija. MySQL je dio LAMP strukture za web aplikacije, što je akronim za Linux, Apache, MySQL, PHP/Perl/Python.

Text

Description automatically generated

Slika 2.5 Primjer MySQL koda

## **Integrirano razvojno okruženje**

Integrirano razvojno okruženje ili IDE je softver za izgradnju aplikacija koji kombinira uobičajene alate za *developere* u jedno grafičko sučelje. Omogućuje lagani i brzi početak izgradnje aplikacije jer *developeri* ne moraju ručno podešavati postavke, isto tako na mnoge načine pomažu tijekom same izgradnje aplikacije. IDE se tipično sastoji od:

Uređivača izvornog koda – omogućuje pisanje, izmjenu i brisanje koda, bojanje sintaksa, automatsko dovršavanje, provjera *bug*-ova  
 Lokalne automatizacije izgradnje – sastavljanje izvornog koda u strojni kod, pakiranje binarnog koda, izvršavanje automatiziranih testova  
 Program za ispravljanje pogrešaka – program za testiranje koji grafički može prikazati lokaciju bug-a u izvornom kodu

### Phpstorm

*Phpstorm* je više-platformski IDE za PHP *development*. Pruža uređivač za PHP, HTML i JavaScript s analizom koda u letu, sprječavanje pogreški i automatskom refaktorizacijom za PHP i JavaScript. Sadrži i SQL uređivač u kojem se mogu kreirati i mijenjati upiti. *Phpstorm* je savršen alat za ovaj projekt, omogućava povezivanje s bazom podataka, te javlja greške, predlaže poboljšanja, automatski popunjava kod i automatizira neke zamorne procese.

U IDE-u se nalazi sve što treba za izradu aplikacije, tako da nema potrebe u većini slučajeva uopće mijenjati radni prozor. Za vrijeme izrade projekta najviše su se koristile ugrađene funkcije za povezivanje i interakcijom s bazom podataka, ugrađenu kontrolu verzije pomoću *Git*, mijenjanje prozora je bilo potrebno samo kada su vizualni elementi stanice bili mijenjani. Još neke korisne funkcije su lagano uspoređivanje datoteka i njihovih promjena, automatsko popunjavanje imena elemenata iz jedne datoteke u drugoj, te detaljni ugrađeni opisi alata (*tooltips*) koji su puno pomogli pri razumijevanju funkcija.

Text

Description automatically generated

Slika 2.6 Phpstorm sučelje

## **Server stack**

*Server stack* je skupina softvera koji zajedno formiraju operacijsku infrastrukturu na nekom računalu kako bi izvršili neki zadatak. Takvo rješenje mora sadržavati softversku potporu za više programskih jezika i web servisa. Npr. LAMP, WISA, MEAN.

### XAMPP

XAMPP je besplatan i *open-source* više-platformski *web server stack* razvijen od Apache Friends. Sastoji se od Apache HTTP *servera*, MariaDB baze podataka i prevoditelja za skripte pisane u PHP i Perl jezicima. Ime XAMPP se mijenja ovisno o operacijskom sustavu, X u imenu označava njegovu podršku za više platforma, LAMPP za Linux, WAMPP za Windows i MAMPP za MacOS. XAMPP ima vrlo laganu instalaciju, zahtijevajući samo jednoj .zip ili .exe datoteci kako bi se instalirao, s početnim postavkama koje odmah omogućuju pokretanje i funkciju *servera*. Redovito je ažuriran na najnovije verzije Apache, MariaDB, PHP i Perl, te dolazi s dodatnim modulima poput OpenSSL, phpMyAdmin, MediaWiki, Joomia, WordPress i dr. Pošto tipični *web serveri* koriste iste komponente kao i XAMPP, moguće je prebaciti *web* stranicu s lokalnog *servera* na Internet.

Graphical user interface, application

Description automatically generated

Slika 2.7 XAMPP sučelje

## **Verzija kontrole**

Verzija kontrole je vrlo važan aspekt *developmenta*, pogotovo u većim projektima i projektima u timu. Ona nam pomaže pri čuvanju svakog koraka tijekom pisanja koda projekta, omogućavajući testiranje promjene koda bez straha za trajnom promjenom, te vraćanje koda u slučaju pogreške ili gubitka. Npr. Git, CVS, SVN, Mercurial.

### Git

*Git* je softver za verziju kontrole koji se koristi za koordinaciju rada između *developera* koji zajedno razvijaju neki softver. Njegov cilj je brzo i efikasno obavljati posao verzije kontrole, čuvajući integritet podataka i podršku za nelinearne poslove u obliku grana. Svaki *Git* direktorij na bilo kojem računalu je svoj potpun repozitorij s kompletnom povijesti, neovisan o dostupnosti mreže ili nekog centralnog *servera*. Svaki *Git* direktorij ima grane, početna grana je „*main*“ koja se napravi na početku izrade repozitorija, iz te grane se mogu granati druge grane koje imaju svoje verzije, te iz njih se može granati još grana, a mogu se i spojiti nazad sa „*main*“ preko „*merge*“ procesa.

Text

Description automatically generated

Slika 2.8 Primejr Git komande

### GitHub

*GitHub* je kompanija koja pruža usluge internetskog posluživanja za razvoj softvera i kontrolu verzije pomoću *Git*. Nudi sve mogućnosti *Git-*a, dodajući svoje funkcije koje poboljšavaju koordinaciju timova i nudi ljepši izgled procesa kontrole verzije. Vrlo često se koristi kako bi neovisni *developeri* objavili svoje *open-source* projekte. Od studenog 2021. godine, *GitHub* ima preko 73 milijuna *developera* i preko 200 milijuna repozitorija od kojih su 28 milijuna javni.

Graphical user interface, text, application, email

Description automatically generated

Slika 2.9 Primjer GitHub repozitorija

## **Haikei**

Grafički izgled svake stranice je važan aspekt koji pridonosi kvaliteti stranice, *Haikei* je online alat koji omogućava vrlo jednostavno kreiranje SVG oblika ili uzoraka koji se mogu lagano implementirati na *web* stranicu. Neke od opcija su *Blob, Wave, Blurry Gradient, Layered Waves, Low Poly Grid, Layered Peaks, Stacked Steppes* i dr. SVG grafika je vrlo korisna jer se ne nalazi u tradicionalnom piksel obliku, već je napravljena pomoću matematičkih vektora, što omogućava da slika zadrži svoju kvalitetu neovisno o tome koliko mi zumiramo na nju.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 2.10 Slika Haikei sučelja

# **Struktura i funkcionalnosti web sjedišta**

*Web* sjedište se sastoji od više *web* stranica koje su međusobno povezane i svaka ispunjava neku ulogu kako bi se postigao cilj prikazivanja informacija korisniku koje on želi. Glavne stranice koje korisnik vidi su; početna stranica (*index*), cjenik, *webshop* i *product*. Uz glavne stranice, postoje i sporedne stranice koje služe za administratora stranice i omogućavaju mu izmjenu podataka u *webshop*-u, a to su: adminlogin, adminshop, prodadd i prodchange.

Gotovo svaka stranica je napravljena s *grid* elementima zbog lagane tranzicije na mobilne uređaje. Indeks i cjenik stranice ne koriste PHP, za njih nema neke posebne potrebe da se spajaju s bazom podataka te su napravljene s tipičnom HTML, CSS i JS kombinacijom. *Webshop* stranica s druge stane, je gotovo cijela napravljena preko PHP-a i nekih kombinacija PHP-a i JS-a koje se ne preporučuju, no u ovom slučaju sve uspješno funkcionira. Svakoj stranici dodani su efekti valova na vrhu i dnu pomoću SVG grafike napravljenoj u Haikei *online* aplikaciji. To stranicama dodaje zanimljiviji i ljepši izgled nego samo ravne linije.

## **Početna stranica**

Početna stranica služi tome da korisnik može lagano pristupiti općim informacijama o servisu, to uključuje; radno vrijeme, adresu, mobitel i email. Na vrhu stranice se nalazi navigacijski meni s kojim korisnik može pristupiti ostalim stranicama *web* sjedišta. Ispod navigacijskog menija nalazi se karusel slika logotipa firma s čijim se uređajima servis obično bavi.



Slika 3.1 Početna stanica 1



Slika 3.2 Početna stranica 2

Karusel je izveden tako da se unutar jednog <div> elementa klase „*slideshow*“ stave slike klase „*slideimg*“ koje su na početku sve skrivene. Preko JavaScript funkcije „vrti()“ zovemo funkciju koja će staviti animacije na slike i u intervalima rotirati beskonačno zadane slike.

Text

Description automatically generated

Slika 3.3 Kod za karusel slika 1

Text

Description automatically generated

Slika 3.4 Kod za karusel slika 2

Funkcija „vrti()“ uzima broj elemenata klase „slideimg“, u „for“ petlji postavlja da se slike ne mogu vidjeti, zatim počinje brojač od 0 i preko „x[0].style.display“ mijenja se vidljivost 1. slike. „setTimer“ funkcija služi da se funkcija ponovo pozove nakon nekog vremena, u ovom slučaju nakon dvije sekunde.

## **Cjenik**

Cjenik je jednostavna stranica, gdje se u tabličnom obliku ispisuju cijene usluga servisa, s njihovom početnom cijenom i cijenom s PDV-om. Na vrhu stranice, kao i na svim ostalim stranicama se nalazi navigacijski meni za kretanje web sjedištem.



Slika 3.5 Izgled cjenik stranice 1



Slika 3.6 Izgled cjenik stranice 2

## **Funkcije**

Ako neke linije koda se moraju često ponavljati ili biti na svakoj stranici, dobro je napraviti posebnu datoteku koja sadrži te funkcije, te ostale stranice mogu koristiti te funkcije bez da se taj kod nalazi u njihovoj datoteci

Text

Description automatically generated

Slika 3.7 Slika PHP funkcija

U „functions.php“ datoteci nalaze se neke korisne funkcije koje se koriste na skoro svim stranicama sjedišta. Funkcija „OpenCon()“ uspostavlja vezu s bazom podataka i omogućava slanje upita i naredbi na *server*. Funkcija „QueryDestroy()“ koristi se kako bi se resetirao upit na *webshop* stranici, korisno kada korisnik otiđe sa stranice vezane uz *webshop* i vraća se na zadani izgled *webshop*-a.

## **3.4 Webshop**

*Webshop* stranica je glavni fokus ovog projekta, napravljen je da korisnik može pretraživati, filtrirati i sortirati stavke uređaja iz baze podataka. Na *webshop* stranici se administrator stranice može prijaviti, te otići na modificiranu verziju *webshop*-a koji mu omogućuje dodavanje i mijenjanje stavki.

Kako bi uopće stanica mogla komunicirati s bazom podataka, moramo ju spojiti preko funkcija koje smo odredili.

Text

Description automatically generated

Slika 3.8 Povezivanje sa bazom podataka

„$conn“ varijabla će nam služiti pri slanju upita, u njoj se nalaze podatci za povezivanje s bazom, dok se u „$page“ varijablu sprema trenutan broj stanice koji je definiran u URL-u. Funkcija session\_start() se koristi kako bi mogli imati pristup sesijskim varijablama.

### 3.4.1 Struktura *webshop*-a

Ispod navigacijskog menija nalazi se <div> element klase „okvirshop“ koji sadrži glavni sadržaj stranice, odnosno rešetku (grid) u kojem su posloženi elementi u sljedećem poretku:

Text

Description automatically generated

Slika 3.9 Rešetka webshop-a 1

Sama rešetka je velika 4x4 elementa, koji se preko „fr“ mjerne jedinice rasporede u odnosu na duljinu roditeljskog (*parent*) elementa. „okvirshop“ u sebi sadrži još jednu rešetku, „okvirgrid“ u kojoj se nalaze stavke uređaja iz baze podataka u 3x3 veličini za sveukupno 9 uređaja po stranici.

Text

Description automatically generated

Slika 3.10 Rešetka webshop-a 2

### 3.4.2 Popunjavanje stavki

Kako bi popunili rešetku sa stavkama iz baze podataka, moramo koristiti PHP da pošaljemo upit *serveru* koji će nam dati informacije iz boze podataka.

Text

Description automatically generated

Slika 3.11 Popunjavanje stavki

Preko upita koji se sastavi korisničkim *inputom* ili unaprijed određenim na učitavanju stranice dobivamo informacije koje su nam potrebne da popunimo naše elemente s informacijama. Varijabla „$stmt“ predstavlja rezultat našeg upita, to je niz (array) u nizu iz kojeg možemo izvaditi informacije iz pojedinog retka. Na početku provjeravamo da li uopće imamo rezultata na postavljeni upit, i ako nemamo, izbaci se poruka. Ako upit vrati rezultate, rezultat se stavlja u „foreach“ petlju koja onda za svaki redak rezultata napravi <div> element klase „item“ i popuni ga s informacijama tog retka.

### 3.4.3 Filtriranje i pretraživanje

Kako bi *webshop* bio funkcionalan, treba biti implementirana mogućnost pretraživanja, filtriranja i sortiranja stavki kako bi se korisniku olakšao pronalazak traženog uređaja ili dijela. Pretraživati se može prema; imenu, cijeni i proizvođaču, te sortirati; od jeftinijeg prema skupljem, od skupljeg prema jeftinijem, od A do Z, od Z do A, od najnovijeg i od najstarijeg. Izvedeno je sa skupinom <input> elemenata koje korisnik popunjava.

Text

Description automatically generated

Slika 3.12 Tražilica u HTML-u

A screenshot of a computer

Description automatically generated

Slika 3.13 Traženje po minimalnoj i maksimalnoj cijenu u HTML-u

Kod filtriranja po cijeni imamo jednostavnu provjeru vrijednosti polja, u slučaju da korisnik pokuša unijeti više od osam brojeva, svaki broj nakon osmog će se pobrisati na svakom unosu.

Text

Description automatically generated

Slika 3.14 Odabir sortiranja u HTML-u

Kod sortiranja, imamo <select> element koji je padajući izbornik s opcijama za sortiranje. Vrijednosti opcija, ukoliko odabrane, će se koristiti u dinamičkom strukturiranju upita, isto kao i vrijednosti ostalih filtera.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.15 Odabir proizvođača

Za izbor proizvođača, imamo kompleksniju implementaciju. Pošto se proizvođač proizvoljno može postaviti, trebamo dinamički napraviti <input type=“checkbox“> elemente. Započinje se s upitom, koji pregledava sve jedinstvene proizvođače u bazi podataka, pojedine rezultate postavlja u „$array“ niz koji se stavlja u „for“ petlju koja dinamički kreira elemente.

Text

Description automatically generated

Slika 3.16 Gumbi za pretraživanje i resetiranje

Na kraju opcija filtriranja i sortiranja, ostaje nam gumb koji započinje proces filtriranja i sortiranja i gumb koji nakon filtriranja i sortiranja resetira upit.

Text

Description automatically generated

Slika 3.17 Resetiranje upita

Kada je pritisnut gumb „reset“, uništava se upit i vrati se na prvu stanicu *webshop*-a.

### 3.4.4 Dinamički upit

Text

Description automatically generated

Slika 3.18 Proces izrade dinamičkog upita

Ovako izgleda proces izrade dinamičkog upita, sada objašnjenje svakog dijela:



Slika 3.19 Postavljanje varijable filtera

Na početku imamo provjeru pritiska gumba za pretraživanje, ako je gumb pritisnut, sesijska varijabla „filters“ se stavlja u 1 što označava da se umjesto zadanog upita koristi dinamički upit.

A screenshot of a video game

Description automatically generated with medium confidence

Slika 3.20 Zadani upit

Provjera da li je korisnik zadao kakve filtere u kasnijem dijelu koda prije slanja upita.

Text

Description automatically generated

Slika 3.21 Upit za pretraživanje

U ovome dijelu gledamo koja je vrijednost postavljena u „search“ input, te ukoliko nije postavljamo da traži sva imena, a u suprotnome u upit se dodaje vrijednost *inputa* te se nastavlja dalje sastavljati. Metoda $\_POST nam omogućava da dobijemo vrijednost *inputa* naziva „search“ i sa htmlspecialchars() pretvaramo posebne znakove u HTML znakove.

Text

Description automatically generated

Slika 3.22 Upit za minimalnu cijenu

Ovdje se gleda vrijednost „mincijena“ *inputa*, ukoliko korisnik nije postavio vrijednost, automatski se pretpostavlja da je 0, u suprotnome, na upit se dodaje vrijednost inputa.

Text

Description automatically generated

Slika 3.23 Upit za maksimalnu cijenu

Slična situacija kao i za „mincijenu“, ukoliko nije postavljena vrijednost, automatski se stavlja kao veliki broj koji ne bi nikada trebao biti u bazi, u suprotnome se vrijednost „maxcijena“ stavlja u upit.

Text

Description automatically generated

Slika 3.24 Upit za proizvođača 1

Ovdje se provjerava da li su označeni potvrdni okviri (checkbox) i preko „foreach“ petlje se stavljaju vrijednosti svih označenih vrijednosti u upit.

Graphical user interface, text

Description automatically generated

Slika 3.25 Upit za proizvođača 2

Kasnije imamo provjeru ako nije označena niti jedna vrijednost za proizvođača.

Text

Description automatically generated

Slika 3.26 Sortiranje upita

Text

Description automatically generated

Slika 3.27 Odabir sortiranja u HTML-u

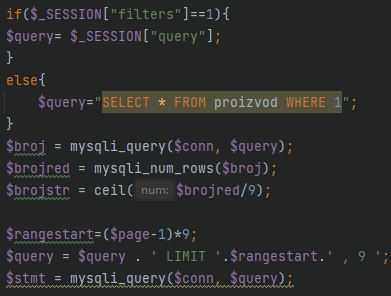
U ovome dijelu provjerava se postavljena vrijednost <select> elementa pod imenom „order“, ukoliko nije postavljena, sortira se prema ID, u suprotnome uzima vrijednost odabranog <option> elementa i dodaje ga u upit.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.28 Sastavljanje upita

Na kraju se sastavlja upit od vrijednosti koje smo prije dodali. Svaki upit započinje sa „SELECT \* FROM proizvod WHERE“, zatim s implode() funkcijom uzimamo vrijednosti iz „$sql“ niza i između svake vrijednosti se stavi „AND“, dodatno kako bi odabrali proizvođača u „IN“ stavljamo još jednu implode() funkciju koja između vrijednosti „$sqlpro“ niza stavlja zarez, te na kraju se dodaje oblik sortiranja koji je korisnik odabrao. Nakon završetka sastavljanja upita, lokacija se promjeni na prvu stanicu *webshop*-a gdje će su učitati upit.



Slika 3.29 Upit tijekom učitavanja stranice

Svaki put kada se stranica učita, provjerava se da li je sesijska varijabla „filters“ postavljena, ukoliko je postavljena, kao upit se stavlja dinamički sastavljeni upit od prije, a u suprotnom upit glasi: „SELECT \* FROM proizvod WHERE 1“. Varijabla „$broj“ u sebi sadrži rezultat upita, preko funkcije mysqli\_num\_rows() dobivamo broj redaka iz varijable „$broj“. Varijabla „$brojstr“ služi kako bi znali koliko stranica stavki imamo, pošto na jednoj stranici imamo 9 stavki, te ako imamo više od 9 stavki moramo ih razdijeliti na stranice. „$rangestart“ nam služi da pratimo od koje stavke stranica mora početi, te na posljetku na „$query“ varijablu dodajemo granice kako bi dobili samo 9 rezultata, te u varijabli „$stmt“ dobivamo rezultate za tih 9 stavki.  
 Izgled zadanog (*default*) upita: „SELECT \* FROM proizvod WHERE 1 LIMIT 0 , 9“ Izgled upita kada korisnik pretražuje bez da je upisao podatke: „SELECT \* FROM proizvod WHERE Ime LIKE '%' AND Cijena BETWEEN 0 AND 999999999 AND Proizvodac IN ( SELECT Proizvodac FROM proizvod ) ORDER BY ID LIMIT 0 , 9“

### 3.4.5 Straničenje

Kao što je prije navedeno, svaka stranica *webshop*-a može imati samo 9 stavki, stoga moramo uvesti sistem straničenja. Iznad stavki se nalazi navigacijska traka s kojom korisnik može prebacivati stranice.

Text

Description automatically generated

Slika 3.30 Navigacijksa traka

U <div> elementu klase „navig“ nalazi se <form> koji sadrži gumbe za promjenu stranice. Gumbi su napravljeni tako da gumb u sredini prikazuje trenutnu stranicu dok onaj lijevo od njega stranicu prije, a onaj nakon njega stranicu nakon. U slučaju da stranica prije je jednaka nuli, umjesto nule će se ispisati tri točke koje označavaju da nema više stranica u tome smjeru, ista situacija je i kod gumba za sljedeću stranicu. Osim ta tri gumba postoje još dva, jedan na svakoj strani koji rade istu funkciju kao i oni ispred njih, omogućavajući korisniku pregledniji način za listanje ako želi koristiti strelice.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.31 Funkcije za prebacivanje stranice

Ovisno o gumbu, na njegov klik aktivirat će se PageMinus() ili PagePlus() funkcija. Način na koji je to izvedeno nije preporučen, no u ovome slučaju radi kako je namijenjeno. Cilj je bio mijenjati stranice što glađe, prvobitno je bilo izvedeno samo pomoću PHP-a, no kada bi korisnik osvježio stanicu bi dobio poruku želi li poslati podatke ponovno, što je bilo vrlo naporno.



Slika 3.32 Limitiranje upita

U ovome dijelu sastavljanja upita od prije stavlja se granica na koliko rezultata može upit vratiti. LIMIT nam omogućuje da od neke točke koje je određena ovisno o stanici (npr. 0 za 1. stanicu, 9 za 2. stranicu, 18 za 3. stranicu, itd.) uzmemo 9 sljedećih stavki.

Graphical user interface

Description automatically generated

Slika 3.33 Izgled webshop-a 1



Slika 3.34 Izgled webshop-a 2

U <footer> elementu nalazi se *link* koji vodi na *login* stranicu za administratora stranice.

## **Stranica stavke**

Svaka stavka unutar *webshop* stranica je u principu samo poveznica na dinamički kreiranu stranicu o toj stavki. Klikom na područje neke stavke će korisnika odvesti na stranicu o tom uređaju/dijelu gdje će se prikazati dodatne informacije poput; gdje naručiti uređaj/dio, koliko ih je raspoloživo, te dugi opis koji pruža detaljniji opis uređaja/dijela.



Slika 3.35 Izlged stranice stavke



Slika 3.36 Dobivanje imena iz URL-a

Text

Description automatically generated

Slika 3.37 Popunjavanje stranice stavke

Radi tako da se na početku iz URL-a dobije ime uređaja/dijela te se u <div> elementu klase „prodbox“ stvara upit čiji se rezultat preko mysqli\_query() funkcije sprema u „$stmt“ varijablu. Preko „foreach“ petlje spremamo dobivene rezultate u njihove varijable, te popunjavamo stranicu na mjestima gdje je to potrebno.

## **Login**

*Login* stranica ima samo jednu funkciju, a to je omogućiti administratoru ulazak na stranicu koja mu omogućuje dodavanje i mijenjanje stavki.



Slika 3.38 Stranica za login

Sastoji se od jednog <div> elementa klase „adminbox“ i u sebi sadrži polja za upis korisničkog imena i lozinke, gumb za povratak i gumb koji inicijalizira provjeru podataka.

Text

Description automatically generated

Slika 3.39 Proces autorizacije korisnika

Klikom na gumb za *login* započinje provjera upisanih podataka. Na početku šaljemo upit bazi i spremamo rezultat kako bi mogli provjeriti da li korisnik postoji u bazi podataka. Ukoliko korisnik postoji provjeravamo lozinku pomoću password\_verify() funkcije, tu funkciju možemo koristiti samo ako smo lozinku napravili pomoću password\_hash() funkcije. Ako je lozinka ispravna postavljamo sesijske varijable kako bi znali da se administrator uspješno ulogirao te ga preusmjerujemo na modificiranu verziju *webshop*-a na kojoj može dodavati i izmjenjivati stavke.

Graphical user interface, text

Description automatically generated with medium confidence

Slika 3.40 Automatsko prosljeđivanje ukoliko se korisnik prijavio prije

U slučaju da se administrator prije ulogirao, ova stranica se preskače.

## **Administracija**

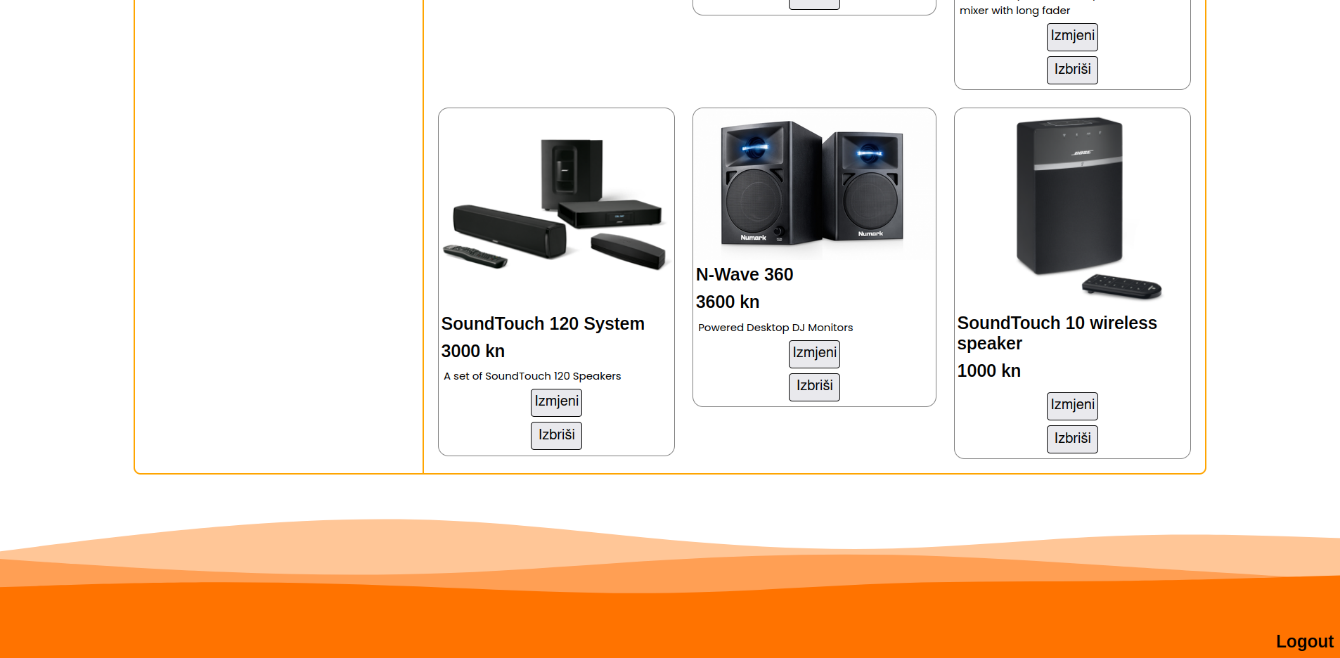
Nakon uspješne autorizacije administrator stranice dobiva pristup modificiranom *webshop*-u gdje ima mogućnosti dodavanja novih stavki, te brisanja i izmjenjivanja postojećih.

### Modificirani *webshop*

Modificirani *webshop* radi na istom principu kao i običan webshop koji korisnik. Jedine razlike su u tome što je prva stavka zamijenjena s opcijom za dodavanje nove stavke i svakoj stavki su dodani gumbi za brisanje i izmjenjivanje.



Slika 3.41 Izgled modificiranog webshop-a 1



Slika 3.42 Izgled modificiranog webshop-a 2

U donjem desnom kutu nalazi se gumb za odjavljivanje, klikom na njega administrator se odjavljuje i vraća nazad na početnu stranicu.

Graphical user interface, text

Description automatically generated

Slika 3.43 Gumbi za izmjenu i brisanje u HTML-u

Unutar svake stavke je dodan novi <div> klase „prodbutton“ koji u sebi sadrži gumbe za izmjenu i brisanje. Pritiskom na gumb za izmjenu odvodi administratora na stranicu za promjenu informacija za taj uređaj/dio. U drugom dijelu se nalazi <form> koji sadrži gumb za brisanje. Pritiskom na taj gumb aktivira se „action“ tog obrasca, odnosno izvršava se „delete.php“ datoteka.

Graphical user interface, website

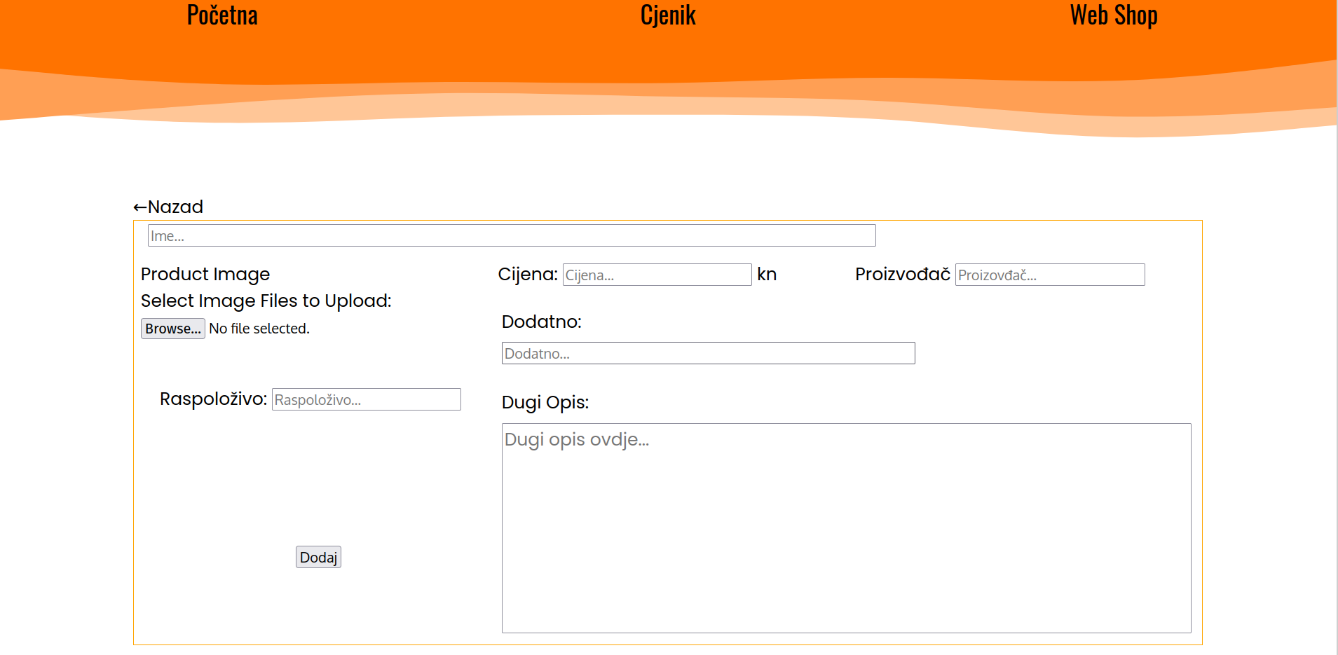
Description automatically generated

Slika 3.44 Provjera autorizacije

Prije učitavanja stranice nalazi se provjera kako se ne bi dogodilo da neautorizirana osoba dobije pristup administratorskim alatima preko upisa lokacije u URL. Komanda „exit;“ ovdje naređuje da se ostatak stranice ne učitava, tako da iako neautorizirana osoba dođe do ove stranice, ne može dobiti nikakve informacije iz lokalne memorije. Ta provjera se nalazi na svakoj stranici koja zahtjeva administratorska prava.

### Dodavanje stavki

Pritiskom na „DODAJ NOVO“ stavku vodi na stranicu koja omogućuje upis informacija o novoj stavki.



Slika 3.45 Izgled dodavanja stavki

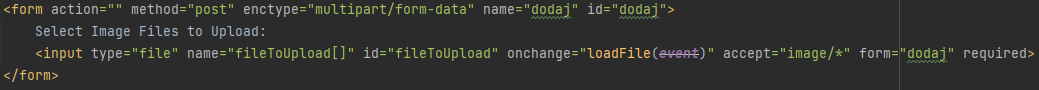
Koristi isti raspored kao i stranice za pojedine stavke, razlika je u tome što sada imamo <input> za svaku informaciju o uređaju/dijelu. Ovdje administrator upisuje informacije, obavezna polja koja mora ispuniti su: ime, slika, cijena i raspoloživo.

Kada se slika odabere, postoji funkcija za predpregled slike napravljena na sljedeći način.

Text

Description automatically generated

Slika 3.46 Funkcija za pred pregled slike



Slika 3.47 Područje za sliku u HTML-u

Kada odabere slika, aktivira se „onchange“ uvjet forme koji izvršava loadFile(event) funkciju. Ta funkcija uzima sliku koju je administrator odabrao, te ju prikazuje u elementu čiji je ID jednak „preview“.

Nakon što administrator ispuni polja koja želi i klikne gumb „Dodaj“ započinje se proces dodavanja informacija u bazu podataka.

Text

Description automatically generated

Slika 3.47 Proces dodavanja nove stavke

Text

Description automatically generated

Slika 3.48 Sakupljanje upisanih informacija

Na početku se uzimaju sve vrijednosti koje je administrator upisao u željena polja i miču se sve HTML i PHP oznake.



Slika 3.49 Određivanje željenog direkorija i dobivanje imena slike

Za sliku definiramo direktorij u koji želimo sliku spremiti i spremimo ime slike u varijablu.

Text

Description automatically generated

Slika 3.50 Početak provjere ispravnosti slike

Ako postoji slika, u „$target\_file“ se sprema putanja slike gdje će se spremiti, „$uploadOk“ se postavlja u 1 i označava da je za sada sa slikom sve u redu. U „$imageFileType“ spremamo tip datoteke kao bi kasnije provjerili da li je datoteka slika koji baza podržava.

Graphical user interface, text, chat or text message

Description automatically generated

Slika 3.51 Provjera da li je slika zapravo slika

Moramo provjeriti da li je datoteka zapravo slika, to se provjerava preko funkcije getimagesize() koje će ako datoteka nije slika dati „FALSE“. Ako je sve u redu „$uploadOk“ će ostati u jedinici, a u suprotnome promijeniti u nulu i ispisati pogrešku.

Text

Description automatically generated

Slika 3.52 Provjera da li već postoji ta slika

Ukoliko datoteka već postoji, neće se dodati nova i prikazati će se greška.

Graphical user interface

Description automatically generated

Slika 3.53 Provjera veličine slike

Provjeravamo koliko je velika datoteka i ako je veća od 50 Mb javlja se pogreška i „$uploadOk“ se stavlja u nulu.

A screenshot of a computer

Description automatically generated

Slika 3.54 Provjera tipa slike

Sliku možemo prihvatiti samo u JPG, JPEG, PNG i GIF tipu, te ako nije jedna od tih, prikazuje se pogreška i „$uploadOk“ se stavlja u nulu.

Text

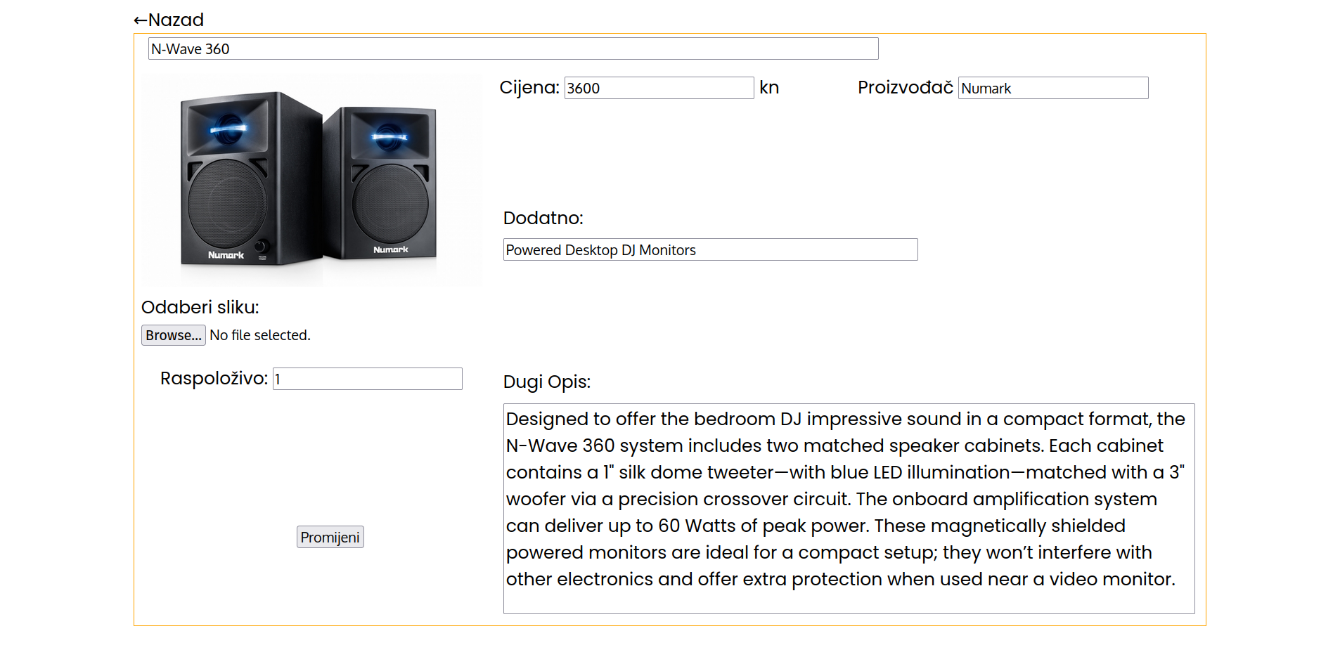
Description automatically generated

Slika 3.55 Postavljanje slike

Ukoliko je nešto bilo u krivu i postavilo „$uploadOk“ u nulu, neće se postaviti slika i prikazuje se zadnja poruka. Ako je sve u redu, slika će se pokušati staviti na željenu putanju preko move\_upload\_files() funkcije, ako ne uspije prikazuje se još jedna greška. Nakon što je slika uspješno postavljena, u bazu podataka se trebaju postaviti upisane informacije i putanja do slike pomoću „INSERT INTO“ SQL komande. Nakon uspješnog prebacivanja slike i ubacivanja informacija, lokacija se prebaci na praznu stranicu za dodavanje nove stavke.

### Izmjena stavki

Stranica za promjenu informacija isto prati raspored za stranice pojedinačnih stavki, razlika je u tome što se postojeće varijable mogu izmjenjivati preko <input> polja.



Slika 3.56 Izlged stranice za izmjenu

Svako <input> polje je automatski popunjeno prilikom učitavanja stranice, to omogućuje brzu promjenu podataka bez potrebe da se sve ponovno upisuje.



Slika 3.57 Dobivanje imena iz URL-a

Text

Description automatically generated

Slika 3.58 Dobivanje informacija iz baze podataka

Podatci se dobivaju tako da se pomoću $\_GET funkcije dobije ime uređaja/dijela, te se šalje upit bazi za ostale podatke koji se stavljaju kao vrijednosti <input> elemenata.

Text

Description automatically generated

Slika 3.59 Popunjavanje elemenata pomoću JavaScript-a

Podatci se preko JavaScript-a upisuju u njihova namijenjena polja.

Graphical user interface, text

Description automatically generated

Slika 3.60 Promjena informacija

Nema puno razlike između načina na koja se mijenjaju informacije na ovoj stranici i načina na koji se dodaju informacije na za to predviđenoj stranici. Razlika ovdje je ta što prebacivanje slike ne treba biti uspješno da se ostale informacije promijene. Zbog tog razloga imamo drukčiji način izbacivanja greške. Uveli smo dvije varijable, „$message“ i „$condition“.

Text

Description automatically generated

Slika 3.61 Provjera slike sa drukčijom porukom

U „$message“ varijablu se, ukoliko ima pogreški, dodaju poruke tijekom provjere slike.

Text

Description automatically generated

Slika 3.62 Postavljanje slike i uvjetna varijabla

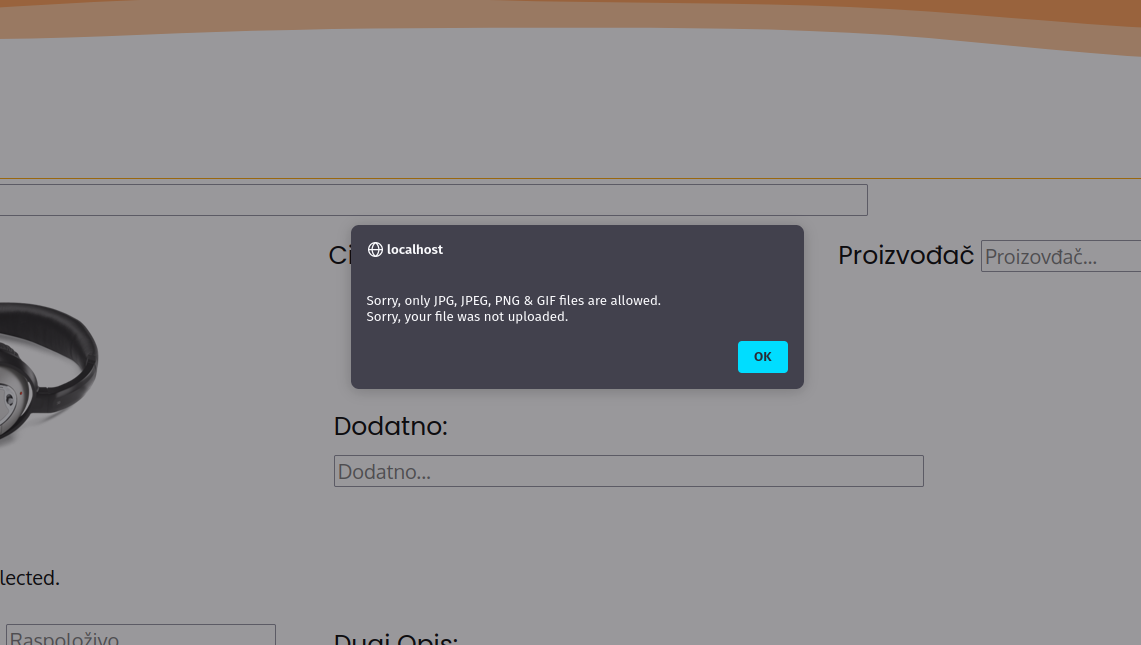
„$condition“ varijabla se postavlja u jedinicu ako je nešto bilo pogrešno tijekom provjere slike i kasnije se koristi za prikazivanje poruke. Ako je slika uspješno prebačena, putanja slike se mijenja na novom imenu (ili starome ako nije promijenjeno), te se lokacija prebacuje na stranicu za promjenu stavke kojoj smo upravo izmijenili podatke. U slučaju da se slika nije prebacila, javljamo pogrešku i „$condition“ se postavlja u jedinicu.

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

Description automatically generated

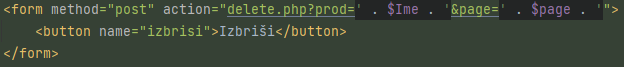
Slika 3.63 Provjera pogreške u prenosu slike

Preko ovog dijela koda provjeravamo da li je „$condition“ bio promijenjen nekom pogreškom i ukoliko je, prikazuje se poruka s greškama, te kad administrator potvrdi, mijenja se lokacija na stranicu za mijenjanje podataka s novim podatcima (osim slike).



Slika 3.64 Poruka nakon pogreške prenošenja slike

### Brisanje stavki



Slika 3.65 Gumb za brisanje u HTML-u

Unutar svake stavke nalazi se gumb za brisanje te stavke, klikom na taj gumb aktivira se „action“ tog obrasca i izvršava se „delete.php“ datoteka kojoj damo dodatne varijable.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.66 Brisanje stavke iz baze podataka

U datoteci „delete.php“ se prvo otvara sesija pomoću session\_start() funkcije koja dozvoljava pristup sesijskim varijablama, zatim se provjerava da li je korisnik ulogiran, i ako je, nastavlja dalje. Prvo dobiva ime iz URL-a, te zatim šalje upit za sliku tog uređaja/dijela. Kada dobije ime slike, pomoću funkcije unlink() briše tu sliku sa *servera*. Zatim se šalje komanda da se pobriše redak odabranog uređaja/dijela, nakon što se to obavi, vraća se nazad na stranicu gdje je korisnik prije bio.

# **Zaključak**

Izrada mrežnog sjedišta je bitna stvar za bilo koju tvrtku ili poslovnicu, za izradu *web* stranice potrebni su *web developeri* koji su danas u velikoj potražnji upravo zbog tog razloga. Ovakav projekt je posao jednog *full-stack developera* jer projekt obuhvaća *front-end* i *back-end* elemente *developmenta*, a i poslove dokumentacije projekta.

Svrha ovoga rada je napraviti funkcionalno *web* sjedište na kojem neka manja firma može postaviti svoje podatke, te imati funkcionalan *webshop*. Iako nema funkcije plaćanja za uređaje/dijelove, ostali elementi za funkcionalan *webshop* su ovdje.

Fokus je bio na brzom pokazivanju informacija koje korisnik želi, zbog toga razloga je početna stranica toliko kratka, nema nepotrebnih informacija na njoj. Isto kao i kod stranice s cjenikom, korisnik ne treba ništa dodatno kliknuti kada otiđe do cjenika, sve su mu informacije odmah na raspolaganju. Stvari koje su se mogle poboljšati u ovome projektu su; grafički dizajn, sigurnost stranice i korištenje nekih drugih tehnologija koje bi bolje odradile posao. Dokumentacije pruža detaljan opis svake stranice na sjedištu uz mnoštvo slika, te bi svatko trebao shvatiti kako svaka stranica funkcionira bez velikih poteškoća.

Ovaj projekt je dao mnogo prilika za učenje i iskustvo izrade *web* stranice koja se može koristiti u stvarnome svijetu. Kompliciranost ovog zadatka i čudni načini na koji programi mogu ponekad raditi pružilo je mnogo trenutaka patnje, no i mnogo trenutaka veselja kada bi neka ideja uspjela.

# **Literatura**

Front-end developer: <https://www.educba.com/what-is-front-end-developer/>   
Back-end developer: <https://www.bitdegree.org/tutorials/what-is-a-back-end-developer/>   
Full-stack developer: <https://bootcamp.cvn.columbia.edu/blog/what-is-a-full-stack-developer/>   
Programski jezici: <https://en.wikipedia.org/wiki/Programming_language>   
Markup jezici: <https://en.wikipedia.org/wiki/Markup_language>   
Skriptni jezici: <https://en.wikipedia.org/wiki/Scripting_language>   
HTML: <https://en.wikipedia.org/wiki/HTML> <https://en.wikipedia.org/wiki/HTML5>  
CSS: <https://en.wikipedia.org/wiki/CSS>  
JavaScript: <https://en.wikipedia.org/wiki/JavaScript> <https://fireship.io/courses/javascript/intro-how-js-works/>   
PHP: <https://en.wikipedia.org/wiki/PHP> <https://www.php.net/>   
MySQL: <https://hr.wikipedia.org/wiki/MySQL>  
IDE: <https://www.redhat.com/en/topics/middleware/what-is-ide>   
Phpstorm: <https://en.wikipedia.org/wiki/PhpStorm>  
Server stack: <https://www.techtarget.com/whatis/definition/server-stack>  
<https://www.ionos.com/digitalguide/server/know-how/web-stacks-the-basics-and-examples/>   
XAMPP: <https://en.wikipedia.org/wiki/XAMPP>  
Kontrola verzije: <https://www.atlassian.com/git/tutorials/what-is-version-control>  
Git: <https://en.wikipedia.org/wiki/Git>  
GitHub: <https://en.wikipedia.org/wiki/GitHub>  
Sve poveznice su prestupljene istog datuma: 24.4.2022.

# **Popis slika**

Slika 2.1 Primjer HTML koda i njegov rezultat: <https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic_document>   
Slika 2.3 Primjer JavaScript koda: <https://appdividend.com/wp-content/uploads/2019/07/Javascript-Instanceof-Operator-Example-Instanceof-in-Javascript.png>   
Slika 2.4 Primjer PHP koda: <https://appdividend.com/wp-content/uploads/2019/06/PHP-Array-Sum-Example-PHP-array_sum-Function-Tutorial.png>   
Slika 2.5 Primjer MySQL koda: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fwww.tutorialgateway.org%2Fwp-content%2Fuploads%2FMySQL-Inner-Join-8.png&f=1&nofb=1>  
Slika 2.6 Phpstorm sučelje: <https://pesktop.com/uploads/0042dda94088de4a71abcb62da331eeb.png>   
Slika 2.7 XAMPP sučelje: <https://sundakreatif.com/wp-content/uploads/2018/10/xampp-cp.png>   
Slika 2.8 Primjer Git komande: <https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2018/07/4-4.png>   
Slika 2.9 Primejr GitHub sučelja: <https://stefanscherer.github.io/content/images/2018/06/github-edit-readme.png>   
Slika 2.10 Slika Haikei sučelja: <https://app.haikei.app/>   
Ostale slike su slike koda iz vlastitog projekta.