

Šolski center Novo mesto

Srednja elektro-računalniška šola in tehniška gimnazija

Šegova ulica 112

8000 Novo mesto

APLIKACIJE IN INFORMACIJSKI SISTEMI

(Maturitetna seminarska naloga)

Predmet: Računalništvo

Avtor: Žiga Kastelic, T4a

Mentor: Gregor Mede, univ. dipl. inž.

Novo mesto, april 2025

ZAHVALA:

Za pomoč pri postavljanju strežnika na splet ter na splošno pri kreaciji strežnika bi se rad zahvalil sošolcu Izaku Fabjanu. Prav tako pa bi se rad zahvalil profesorjema Gregorju Medetu in Simonu Vovku, za vse odgovore na vprašanja, ki sem jih jima postavil ter razjasnitev možnih nejasnosti pri izdelavi te seminarske naloge. Prav tako, sem jima hvaležen za vso znanje, podano v zadnjih dveh ali treh letih, brez katerega ne bi imel podlage, na kateri bi lahko svoje znanje nadgrajeval.

POVZETEK:

V tej seminarski nalogi je predstavljen razvoj spletne aplikacije izdelane s pomočjo različnih spletnih tehnologij. Namenjena je prodajanju računalnikov in računalniških komponent. Rešuje problem primanjkljaja praktično zgrajenih, a grafično dobro oblikovanih aplikacij, ki so uporabniško prijazne.

Ta seminarska naloga je primarno služi opisu izdelave spletne aplikacije v obliki spletne trgovine in obrazložitvi njenega delovanja. Za začetku seminarske naloge je najprej opredeljen problem, ki je služil kot povod za izdelavo te naloge. Sledi opis načrtovanja izdelave naloge po korakih. Nato je na vrsti opis izbranih tehnologij, uporabljenih pri nadaljnji izdelavi aplikacije. Potem je opisano delovanje aplikacije po delih: najprej indeks in noga ter glava aplikacije, nato podatkovna baza, sledi dodajanje vsebine s prijavo in registracijo, potem prikaz izdelkov iskanje ter košarica s plačevanjem, PC Builder ter nato opis dodajanja aplikacije na Oraclov strežnik. Delovanje aplikacije je tudi testirano s tehniko bele in črne škatle.

Pri izdelavi aplikacije je uporabljen HTML z Bootstrapom in CSS za grafično predstavitev elementov, za dodajanje funkcionalnosti PHP in JavaScript, za dostop do podatkovne baze, ki podpira spletno aplikacijo pa je uporabljen SQL. Pri olajšanju in omogočenju izdelave plačevanja je na pomoč priskočil Stripe API, pri postavitvi spletne aplikacije na strežnik pa je uporabljen Oraclov Cloud strežnik z nadzorno ploščo aaPanel.

KLJUČNE BESEDE:

Spletna stran, spletna aplikacija, računalnik, komponente, skripta, strežnik, podatkovna baza, Stripe API, Oracle, PHP, SQL, CSS, Bootstrap, HTML

SUMMARY:

This thesis presents the development of a web application built using various web technologies, designed to sell computers and computer components. It addresses the problem of the lack of practical, well-designed, graphically user-friendly applications.

The primary purpose of this thesis is to describe the development of a web application in the form of an online shop and to explain how it works. To begin the assignment, the problem that served as the trigger for the creation of this assignment is first defined. This is followed by a step-by-step description of the design of the task. Then comes a description of the selected technologies used in the further development of the application. Then the operation of the application is described in parts: first the index and the footer and the header of the application, then the database, followed by adding content with login and registration, then the display of the search products and the shopping cart with payment, the PC Builder and then a description of adding the application to the Oracle server. The functioning of the application is also tested using the white and black box technique.

HTML with Bootstrap and CSS for the graphical representation of the elements is used to build the application, PHP and JavaScript are used to add functionality, and SQL is used to access the database that supports the web application. The Stripe API is used to facilitate and enable the creation of the payment, and the Oracle Cloud rack with the aaPanel control panel is used to deploy the web application on the server.

KEY WORDS:

Web page, web application, computer, component, script, database, Stripe API, Oracle, PHP, SQL, CSS Bootstrap, HTML

SEZNAM KRATIC IN OKRAJŠAV:

HTML - Hyper Text Markup Language – označevalni jezik, namenjen izdelavi spletnih strani

CSS - Cascading Style Sheets – jezik namenjen postavitvi in okrasitvi elementov znotraj spletne strani. Opiše, kako bodo HTML elementi prikazani

PHP – Hypertext preprocessor – splošen jezik, namenjen programiranju spletnih strani in aplikacij

SQL - Structured Query Language – jezik uporabljen za upravljanje podatkov, zlasti v sistemu za nadziranje relacijskih podatkovnih baz.

API - Application Programming Interface – povezava med računalniki ali računalniškimi programi

RAM – Random Access Memory - Notranji pomnilnik računalnika

GPU - Graphics processing unit – grafična kartica

CPU - Central processing unit - procesor

PC - Personal computer – osebni računalnik

XAMPP - Cross-Platform, Apache, MySQL, PHP, and Perl - platforma, ki razvijalcem omogoča lokalno preizkušanje kode na lastnih računalnikih

KAZALO VSEBINE:

Vsebina:

ZAHVALA:	3
POVZETEK:	5
KLJUČNE BESEDE:	5
SUMMARY:	7
KEY WORDS:	7
1. UVOD	1
2. OPREDELITEV PROBLEMA	2
3. NAČRT IZVEDBE	3
3.1. Osnovanje ideje	3
3.2. Izbira jezikov in opreme	4
3.3. Izdelava osnove podatkovne baze	4
3.4. Izdelava navigacije in noge strani	5
3.5. Izbira imena, logotipa in osnovni indeks	5
3.6. Prijava, registracija in povezava do podatkovne baze	6
3.7. Prikaz tipa in izdelkov	6
3.8. Administrator, dodajanje izdelkov in iskanje	6
3.9. Košarica in plačevanje	6
3.10. PHPmailer, dodelovanje prijave	7
3.11. PC builder	7
3.12. Izdelava zunanjega strežnika	7
4. UPORABLJENE TEHNOLOGIJE	8
4.1. HTML, CSS, BOOTSTRAP – Izgled spletne strani	8
4.2. PHP in JavaScript – Dodajanje funkcionalnosti	9
4.3. SQL – Podatkovne baze	9
4.4. Stripe API – plačevanje, PHPmailer	10
4.5. Zunanji strežnik	10
5. OPIS IZDELAVE	12
5.1. Splošna predloga in indeks	12
5.2. Podatkovna baza	14
5.3. Dodajanje vsebine (admin)	15
5.4. Prijava in registracija	16
5.5. Prikaz izdelkov	18
5.6. Iskanje	20
5.7. Košarica	21

5.8.	Plačevanje (integracija Stripe API-ja)	22
5.9.	PC Builder	25
5.10.	Dodajanje aplikacije na Oraclov strežnik	27
6.	TESTIRANJE	29
6.1.	Metoda črne škatle.....	29
6.1.1.	Prijava	29
6.1.2.	Registracija.....	30
6.1.3.	Dodajanje izdelkov.....	31
6.1.4.	Iskanje	32
6.1.5.	Košarica in plačevanje	32
6.1.6.	PC Builder, prikazi.....	34
6.2.	Metoda bele škatle	35
6.2.1.	Prijava	35
6.2.2.	Registracija.....	35
6.2.3.	Dodajanje izdelkov.....	36
6.2.4.	Iskanje	37
6.2.5.	Košarica in plačevanje	37
6.2.6.	PC Builder, prikazi.....	37
7.	ZAKLJUČEK	39
	Bibliografija	40
	PRILOGE:.....	41

KAZALO SLIK:

Slika 1:UML diagram primera uporabe	Slika 2:Groba skica postavitve strani	3
Slika 3: UML diagram zaporedja		3
Slika 4: UML razredni diagram		5
Slika 5: Logotip trgovine		5
Slika 6: Indeks z glavo in nogo		13
Slika 7: Končni razredni diagram		14
Slika 8: Stran za dodajanje.....		16
Slika 9: Prijava	Slika 10:Registracija.....	18
Slika 11:Prikaz izdelkov		20
Slika 12: Podroben prikaz izdelka		20
Slika 13: Košarica		22
Slika 14: Blagajna		25
Slika 15: PC Builder		27
Slika 16: Postopek prijave	Slika 17: Po prijavi	29
Slika 18: Vpis kot normalen uporabnik		29
Slika 19: Napačna prijava		30
Slika 20:Napačen email	Slika 21: Prazno polje.....	30
Slika 22: email	Slika 23: Potrditev emaila.....	30
Slika 24: Uspešna registracija		31
Slika 25:Napačen tip slike		31
Slika 26: Izdelki pred dodajo		31
Slika 27:Po dodaji izdelkov		32
Slika 28:Iskanje brez rezultatov.....		32
Slika 29:Košarica pred izbrisom.....		32
Slika 30: Košarica po izbrisu		33
Slika 31:Prazna košarica		33
Slika 32: Ne veljavna številka kartice.....		33
Slika 33: Napačen datum veljavnosti.....		33
Slika 34: Uspešno naročilo	Slika 35:email s potrdilom.....	34
Slika 36: PC Builder z 1 izdelkom	Slika 37: PC Builder brez izdelkov	34
Slika 38: Prazna košarica, pred dodajanjem	Slika 39: Dodan 1 izdelek v košarico	34
Slika 39: Vsebina tabele.....		35
Slika 40: Vsebina tabele rso_cpu pred dodajanjem		36
Slika 41: Vsebina tabele rso_cpu po dodajanju		36
Slika 42: Košarica pred dodajanjem		37
Slika 43: Košarica po dodajanju		37
Slika 44: Košarica po izbrisu		37
Slika 45:pcbuid pred dodajanjem.....		38
Slika 46:pcbuid po dodajanju.....		38
Slika 47: pcbuid po izbrisu		38

1. UVOD

To seminarsko nalogo izdelujem, da izvem več o tehnologijah izdelave spletnih strani, tako njihovega prednjega, kot tudi zadnjega dela. To znanje, ki ga bom pridobil se mi zdi praktično uporabno, ne samo znotraj računalniškega konteksta, temveč na splošno v vsakdanjem življenju. Skozi to seminarsko nalogo bom na kratko opisal tehnologije, ki sem jih uporabil pri izdelavi spletne aplikacije, to pa bom tudi izdelal in nato testiral z metodo črne in bele škatle. Opisal bom tudi načrtovanje samega projekta po korakih.

Spletna aplikacija bo vsebovala stran na kateri si lahko uporabniki ogledajo izdelke po kategorijah, si jih ogledajo detajlno ter jih nato tudi dodajo v košarico. Izdelal bom tudi stran, ki se bo ukvarjala s plačevanjem tako, da bo lahko kupec izbrane izdelke tudi kupil. Prav tako imam namen izdelati gradilnik računalnikov, ki bo mojo spletno stran razlikoval od drugih podobnih. Seveda se bodo kupci imeli tudi možnost prijaviti v svoj račun in tega ustvariti. Lahko bodo izdelke tudi iskali s pomočjo iskalnika. Prav tako bom administratorju ali pa upravitelju spletne strani omogočil dodajanje novih izdelkov v že obstoječe kategorije. Za te spletne strani bom izdelal tudi podatkovno bazo, ki jo bo podpirala. Tekom te seminarske naloge bom predstavil delovanje moje spletne strani, ki jo bom izdelal s pomočjo znanja, deloma pridobljenega v šoli, a po večini doma, v prostem času.

Končni cilj je izdelati polno funkcionalno spletno aplikacijo, ki bo delovala kot spletna trgovina. Ta bo vsebovala čim manj hroščev in bila čim bolj uporabniško prijazna ter vizualno lepa.

2. OPREDELITEV PROBLEMA

Preden sem začel z izdelavo te seminarske naloge sem se dolgo časa spraševal o tem, kaj naj bo njena tema, ki ima tako velik pomen. Odločil sem se za temo aplikacij in informacijskih sistemov, saj se mi zdi, da mi bo znanje pridobljeno pri taki nalogi splošno še najbolj koristilo v prihodnosti, skozi študij in v moji poklicni poti. Zdelo se mi je praktično uporabno, da vem nekaj o spletnih aplikacijah, saj se tehnologije na tem področju razvijajo relativno hitro ter bi ob pojavu potrebe po izboljšanju znanja na tem področju, to lažje nadgradil, če bi imel že predznanje. Zanimali so me prav tako različni programski vmesnik aplikacij, ki bi mi lahko pri izdelavi koristili. Zanimivo mi je tudi izdelovati vizualno privlačne spletne strani, saj se mi zdi, da vsaj pri spletnih prodajalnah s tem kupce najlažje privabimo.

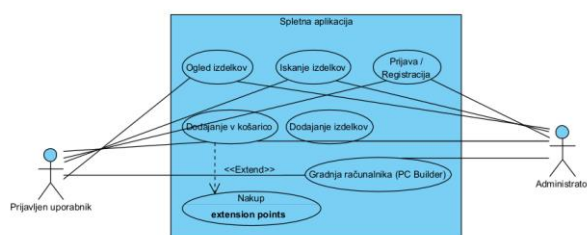
Ugotovil sem, da na spletu ne obstaja veliko portalov za nakupovanje računalniških komponent in celih računalnikov, ki bi bili grafično zadovoljivo in hkrati praktično izdelani. Prav tako sem opazil, da nekatere trgovine izkušajo probleme z dobavo izdelkov, katere bi moja spletna stran izboljšala. Prav tako se mi pri izdelavi strani zdi zelo pomembna transparentnost in dejstvo, da pri prodaji ni skritih stroškov. Nekatere trgovine imajo tudi problem s tem, da si kupci, če kupujejo posamezne komponente, ne znajo predstavljati kaj vse potrebujejo za izdelavo delujočega računalnika. Problem lahko nastane tudi, ko uporabnik kupuje komponente, a si ne zna med nakupom predstavljati celotnega zneska gradnje ali pa ne ve katere komponente so med seboj kompatibilne, ustrezne. Za rešitev vseh teh problemov sem se odločil izdelati sistem za izdelavo računalnikov (t.i. PC builder), ki reši vse te probleme.

Pri izbiri izdelkov, ki jih bom »prodajal« na moji spletni strani sem se odločil za računalniške komponente in cele, že sestavljene računalnike, ker sem s tem že dobro seznanjen in nisem potreboval kaj veliko dodatnega raziskovanja o izdelkih samih. Ta tema me tudi zelo zanima.

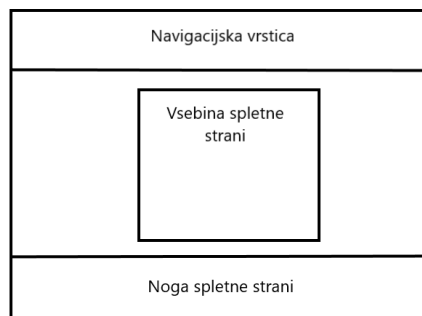
3. NAČRT IZVEDBE

3.1. Osnovanje ideje

Pred kakršno koli izdelavo strani, sem pripravil nekaj idej in z njimi naredil vizualni osnutek moje spletne aplikacije. Kot končno funkcionalno celoto lahko predstavim moje spletne aplikacije prikažem s spodnjim UML diagramom primera uporabe. Dodal sem tudi še grobo skico postavitve spletne strani:

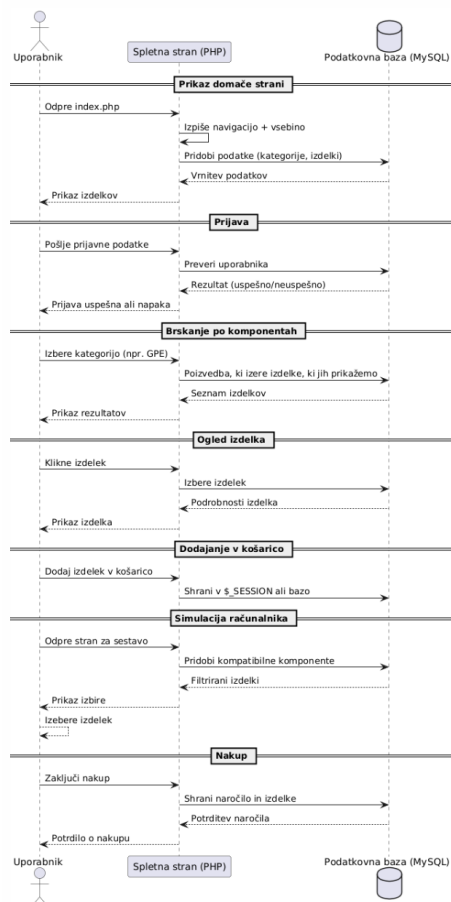


Slika 1: UML diagram primera uporabe



Slika 2: Groba skica postavitve strani

Približno pa sem osnoval tudi funkcionalnost v ozadju spletne strani. To najlažje prikažem z diagramom dejavnosti:



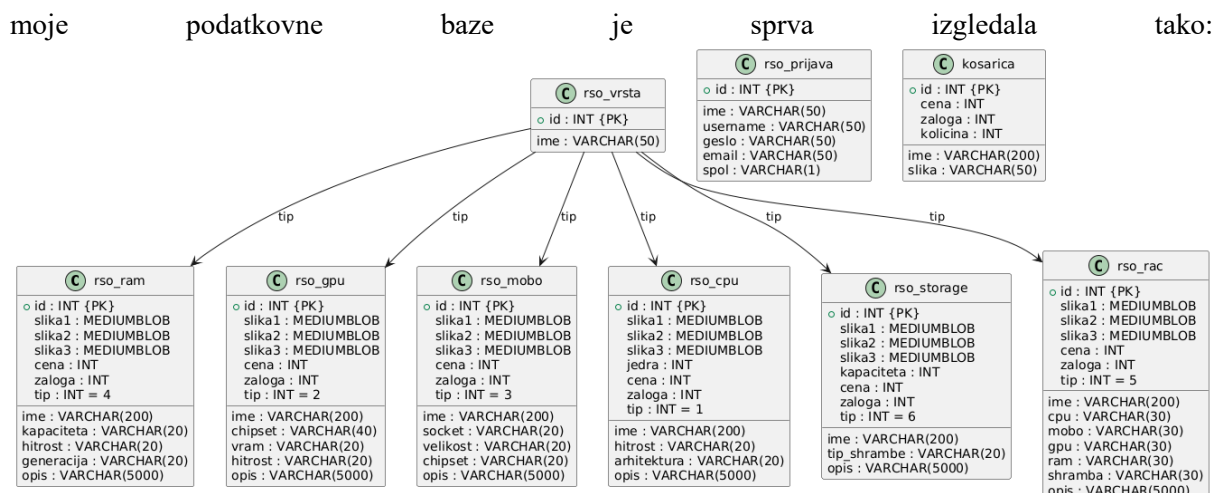
Slika 3: UML diagram zaporedja

3.2. Izbira jezikov in opreme

Na začetku izdelave seminarske naloge, takoj po izbiri teme sem začel razmišljati o tem, katere programske jezike oz. okvirje (t.i. frameworke) naj uporabim. Sprva sem si ogledal nekaj videoposnetkov o izdelavi spletnih strani na splošno ter nato sem se še malo bolj poglobil v samo izdelavo spletne strani namenjeni prodaji. Velika večina avtorjev videoposnetkov, del, ki sem jih prebiral, se je strinjalo, da je za začetnike (med katere sem spadal tudi jaz) najbolj uporaben HTML, katerega lahko grafično dopolnjujemo s CSS – jem. Z obema programskima jezikoma sem imel že malo izkušenj, a z uporabo in funkcionalnostjo specifično surovega CSS – ja nisem bil preveč zadovoljen, saj se mi je ta zdela počasna za kreacijo. Prav tako se mi ne zdi praktično, da potrebujem za kakršen koli dodatek ustvarjati novo skupino elementov znotraj datoteke, namenjene implementaciji CSS-a v spletno stran. Glede HTML – ja pa se mi zdi, da predstavlja nekakšno zlato normo izdelave spletnih strani. Je hkrati prijazen začetnikom, splošno uporabljen tako, da iskanje dokumentacije, možnih dodatkov na spletu ne bi bilo problem. Prav tako se mi HTML zdi kot dobra osnova na kateri lahko gradimo in na njo nalagamo mnogo dodatkov, ki hkrati olajšajo delo z njim in omogočijo izdelavo boljše, bolj uporabniško prijazne spletne strani. Eden izmed teh dodatkov, ki sem ga uporabil se imenuje Bootstrap, CSS, JavaScript in HTML knjižnico, ki zelo olajša in pospeši ustvarjanje elementov v prej omenjenih jezikih. Implementacija tega, mi je prišla zelo prav, saj sem zaradi njega na samemu grafičnemu dizajnu spletne strani porabil veliko manj časa, kot pa bi, če bi uporabljal surov CSS. Za dodajanje dinamičnosti in interaktivnosti spletni strani sem uporabil PHP, ki je prav tako kot HTML splošno znan in uporabljen jezik. Prav tako ga je mogoče dobro integrirati v HTML. Pri uporabi tega sem se odločal tudi, ali bi uporabil katerega od mnogih PHP ogrodij, kot so Laravel, Symfony in druga. Na koncu sem se odločil proti implementaciji ogrodij, saj se mi zdi, da bi na učenju in implementaciji teh izgubil več časa, kot pa bi ga pridobil z uporabo prednosti le-teh, kot so zmanjšanje potrebe po ponavljajočem se kodiranju, pospešitev procesa razvoja... Prav tako se mi zdi, da mi bo splošno znanje PHP – ja prišlo v prihodnosti bolj prav kot pa, da se osredotočim na posamezno ogrodje. Nekaj malega funkcionalnosti sem dodal tudi z JavaScriptom, ki je podoben Javi, jeziku, ki ga dobro poznam. Za izdelavo podatkovne baze in komunikacijo z posameznimi tabelami znotraj nje sem uporabil SQL, ki je tako kot vsi ostali uporabljen zelo na široko in je zelo uporabniško prijazen.

3.3. Izdelava osnove podatkovne baze

Prva stvar, ki sem jo naredil, ko sem izbral jezike, ki sem jih uporabil, je vizualizacija same strukture podatkovne baze, ki jo bom potreboval pri izdelavi spletne trgovine. Najprej sem dodal tabele za posamezne skupine artiklov, ki jih prodajam, to so procesorji, grafične kartice, matične plošče, spomin ter shrambo. Od osnovnih komponent, ki sestavljajo računalnik tako nisem izbral napajalnika, ohišja in hladilnika procesorja, saj se mi zdi da ti ne vplivajo na samo funkcionalnost izdelanega računalnika v tako veliki meri, kot ostale komponente. Nato sem dodal še tabelo za košarico, saj sem se zavedal, da bodo kupci, ki jih bodo nameravali kupiti potrebovali shraniti nekam. Prav tako sem dodal tabelo rso_vrsta, ki je namenjena povezavi vseh tabel skupaj in določanju nekakšnega ID-ja, ki ga ima posamezna tabela za lažjo integracijo teh v sistem. Dodal sem tudi tabelo za že zgrajene računalnike, katere se mi zdi, da je normalno, da jih prodaja prav vsaka računalniška trgovina. Po nekaj razmisleka sem se odločil, da se morajo biti uporabniki tudi morajo biti sposobni prijaviti. Vsaki od teh tabel sem dodal tudi attribute, ki jih je po mojem mnenju najbolj potrebovala oz. so bili smiselni. Prvotna struktura



Slika 4: UML razredni diagram

3.4. Izdelava navigacije in noge strani

Po izdelavi osnov podatkovne baze, ki sem jo imel namen uporabiti pri seminarski nalogi sem se lotil izdelave nekaterih grafičnih delov, ki bodo predstavljeni na vsaki strani spletne aplikacije. To sta navigacija (navbar) in noga (footer) strani. Sprva sem oblikoval funkcionalnost navigacije, to so vse povezave s katerimi bom preusmeril uporabnika na strani za prikaz vseh izdelkov enega tipa (posebej računalniki, grafične kartice, procesorji, RAM, shramba in matične plošče). Dodal sem tudi povezavo na spletno stran, ki bo omogočala prijavo v uporabniški račun, ki ga bo imel uporabnik. Omeniti moram, da sem sprva oblikoval samo povezave, same spletne strani in skripte, ki so bile omenjene oz. na katere sem se skliceval pri povezavah, pa sem dodal kasneje. Zdelo se mi je, da si bom z izbiro takšnega zaporedja izdelave nadaljnje korake izdelave lažje predstavljal.

3.5. Izbira imena, logotipa in osnovni indeks

Po zaključku oblikovanja navigacije in noge spletne strani sem se lotil izbire imena in logotipa spletne trgovine, ki sem jo ustvarjal. Po nekaj minutah razmišljanja in pogovora z bratom mi je predlagal ime Shopotron, za katerega sem logotip ustvaril s pomočjo umetne inteligence. Izdelal sem tudi osnovni indeks spletne strani (index.php), ki je deloval kot osnovna stran spletne strani. Ta sprva ni vseboval kaj veliko elementov in je bil že od začetka mišljen kot kratkotrajna rešitev, zato sem ga kasneje tudi izboljšal in prenovil.



Slika 5: Logotip trgovine

3.6. Prijava, registracija in povezava do podatkovne baze

Nato sem se lotil prijavnega obrazca t.j. login.php, kjer sem oblikoval obrazec potreben za prijavo tako grafično kot funkcionalno. To sem storil s pomočjo PHP – ja. Temu sem kasneje dodal še stran signup.php, namenjeno registraciji v aplikacijo. Z dodajanjem preusmeritev sem omogočil tudi prehajanje med login.php in signup.php, če je uporabnik po preusmeritvi na login.php ugotovil, da še nima računa v katerega se lahko prijavi in obratno. Med izdelavo sem sproti ugotavljal kater tehnologije in metode potrebujem še vključiti v seminarsko nalogo. V tem delu seminarske naloge sem moral zaradi prijavljanja in ustvarjanja računov implementirati še povezavo do podatkovne baze, ki sem jo ustvaril na začetku (db.php). Sprva so bili v njen zapisani podatki šolskega strežnika na katerem sem gostil svojo podatkovno bazo. Omeniti moram, da sem spletno stran samo sprva gostil na lokalnem omrežju (svojem osebnem računalniku in do nje dostopal s pomočjo aplikacije XAMPP). Kasneje sem to nadomestil z gostovanjem spletne strani in podatkovne baze na Oraclovemu Cloud strežniku.

3.7. Prikaz tipa in izdelkov

Naslednja stvar, ki sem jo dodal so bili prikazi (search.php) vsakega tipa izdelka posebej. Najprej sem oblikova strani, ki so se prikazale, ko je uporabnik pritisnil na gumb za preusmeritev na določen tip izdelka. Tako sem za vsak tip izdelka ustvaril svojo datoteko, ki se je prikazala po preusmeritvi. Na vsaki od teh datotek so bili prikazani vsi izdelki določenega tipa, vsak v svojem pravokotnik, ki je imel izpisano sliko izdelka, njegovo ime, ceno in količino. Nato sem dodal sem še povezave vsakemu izdelku znotraj posameznega tipa (prikaz.php), ki je uporabnika ob pritisku preusmeril na stran, ki ga je prikazovala njegove attribute, več slik, ceno... ter imela možnost dodajanja izdelka v košarico. Sprva sem imel prikaz vsakega izdelka v svoji datoteki, kasneje pa sem jih z namenom povečanja učinkovitosti združil v eno datoteko, v katero sem preko URL – ja pri preusmeritvi prenesel podatek o ID – ju posameznega izdelka. Tako se je znotraj datoteka pri vsakem primeru izvajal samo en del kode, ki je izpisoval attribute, ki jih je imel specifično ena vrsta izdelka. Podobno sem storil tudi pri prikazu, kjer sem prav tako preko URL – ja lahko pošiljal spremenljivko imenovano num, ki je določala katero tabela bo bila prikazana.

3.8. Administrator, dodajanje izdelkov in iskanje

Nato sem se lotil pogleda administratorja, ki ima vse pravice enake kot uporabnik z izjemo možnosti dodajanja izdelka v tabele. To sem naredil na strani dodaj.php, ki se prikaže samo na zaslonu administratorja. Na tej strani lahko admin izbere kater izdelek bo dodajal s pritiskom gumba na vrhu strani. Po pritisku tega gumba se prikažejo vnosna polja, v katere lahko admin vnese podatke o izdelku, ki ga dodaja. Prav tako mora v formo vnesti 3 slike izdelka. Sprva je moral admin shraniti slike v mapo na lokalnem disku gostitelja spletne strani, kasneje pa sem nadgradil tako podatkovne baze kot tudi spletno stran tako, da je lahko uporabnik namesto povezave ob dodajanju lahko samo izbral datoteko, naloženo na njegovo napravo, ki se je nato shranila v podatkovno bazo. V navigacijo spletnih strani sem dodal tudi iskalnik izdelkov, realiziran s skripto iskanje.php. Ta je namenjen iskanju izdelkov po njihovem imenu. Po vpisu besedila in potrditvi iskanja je uporabnik iz strani na kateri je preusmerjen na stran iskanje.php, kjer so prikazani izdelki, ki iskanju ustrezajo.

3.9. Košarica in plačevanje

V prikaz.php sem skozi skripto dodaj_v_kosarico.php omogočil tudi dodajanje izdelkov v kosarico. Posledično sem moral v tem delu izdelati tudi stran shopping_cart.php, ki prikazuje košarico, v katero lahko kupci dodajajo izdelke. Po grafični dodelavi te spletne strani sem dodal še funkcionalnost, ki je omogočala prikaz izdelkov, dodanih v košarico. Dodal sem še možnost brisanja izdelkov iz košarice s pomočjo skripte odstrani_iz_kosarice.php. Ta omogoča, da vsak izdelek, ki je zapisan v tabeli kosarica

odstranimo iz nje. Izdelki se po tem dodajo nazaj v tabelo tipa katerega so (če smo imeli v košarici 3 enake procesorje, se ti po odstranitvi vrnejo v tabelo `rsu_cpu`, kjer se njihova količina poveča za 3). Nato sem porabil nekaj časa, da sem raziskal možne načine plačevanja, to so različne knjižnice, API – ji ali pa samo metode s katerimi bi si olajšal oziroma omogočil plačevanje uporabnikom. Na koncu sem se odločil za Stripe API, ki ga bom opisal kasneje. Omogoča integracijo plačevanja direktno v mojo spletno stran in ima hkrati relativno hitro in enostavno implementacijo. Omogoča tudi testiranje uspeha plačevanja, kar mi je zelo pomagalo pri izdelavi te seminarske. Čeprav je mogoče Stripe API izdelan malo ne praktično imajo na spletni strani podjetja zelo dobro opisane načine vključitve v spletno aplikacijo, pa naj bo v celoti implementirano v spletno stran ali pa samo kot preusmeritev na stran, ki jo gosti Stripe sam. Pri tem sem moral izdelati skripto `checkout.php`, ki hkrati služi kot stran za vnos podatkov, potrebnih za tako plačilo kot tudi oddajo naročila izdelkov, ki jih kupec plača. Za ta del seminarske sem moral ustvariti tudi še 2 tabeli: izdelki in narocilo. V izdelke so se shranili izdelki, ki jih je naročil kupec, celotno naročilo pa se je shranilo v tabelo `narocilo`. V skripti `process_payment.php`, ki prejme podatke o iz `checkout.php`, sem nato preko Stipa ustvaril kupca in plačilno metodo ter obdelal plačilo pri čemer zelo pomaga tudi Stripe. Če je plačilo uspešno, te `process_payment.php` preusmeri na `payment_success.php`, ki poskrbi za prikaz uspešnega plačila in uporabniku pošlje mail, ki služi kot potrditev nakupa. Za pošiljanje mailov sem uporabil PHP knjižnico imenovano PHPmailer, ki je namenjena prav pošiljanju e-pošte in to tudi zelo olajša.

3.10. PHPmailer, dodelovanje prijave

Nato sem PHPmailer dodal tudi v prijavo, torej uporabnik po vpisu e-maila v prijavo prejme na omenjen mail sporočilo s kodo ki je naključno generirana v `signup.php` datoteki, ki se ukvarja z registracijo. Uporabnik je nato preusmerjen na stran `confirm.php`, na kateri lahko vpiše kodo, ki jo je prejel na e-poštni naslov.

3.11. PC builder

Kot predzadnji dodatek sem na spletno stran dodal t.i. PC builder oz. stran namenjeno izbiri komponent z namenom ustvariti svoj računalnik po meri. To lahko uporabnik naredi na strani `pcbuilder.php`, povezavo do katere sem dodal v navigacijsko vrstico. V skripti PC builderja sem dodal 6 podstrani, 5 od katerih je namenjenih izbiri posamezne komponente, 1 pa prikaže celotno gradnjo skupaj. Prav tako sem na levi strani spletne strani za lažjo izbiro komponent dodal meni, ki ob kliku uporabnika preusmeri na podstran PC builderja namenjeno izbiri posamezne komponente. S pomočjo PHP-ja sem prav tako omogočil, da se potem ko uporabnik izbere določeno komponento, meni, ki omogoča izbiro te komponente skrije. Izdelal sem tudi gumb, ki s pomočjo `odstrani_iz_pcbuilderja.php` omogoča odstanitev elementa iz PC builderja.

3.12. Izdelava zunanjega strežnika

Nazadnje sem tako podatkovno bazo, kot tudi datoteke spletnih strani prenesel na strežnik, ki sem ga gostil na Oraclovemu Cloud strežniku. Potem, ko sem kupil domeno, s katero dostopamo do moje spletne strani sem ustvaril instanco gostiteljskega strežnika sem preko orodja PuTTY spremenil nekaj nastavitev strežnika, ki so mi na koncu omogočale postavitev strežnika na splet. Prav tako sem se s strežnikom prijavil v orodje Apanel, ki omogoča lažje nalaganje aplikacij na strežnik. Po nekaj dneh dela in pridobitvi SSL certifikata, sem na strežnik naložil vse potrebne datoteke ter podatkovno bazo, tako da je sedaj celotna spletna stran na volja neodvisno od stanja mojega lokalnega računalnika. Prav tako sem na koncu izboljšal izgled in delovanje indeksa spletne strani. Tekom celega dela na spletni aplikaciji, sem vse funkcije, ki sem jih uporabljal v mnogih datotekah združil v datoteki `funkcije.php` zaradi preglednosti in praktičnosti ter principa čistega programiranja.

4. UPORABLJENE TEHNOLOGIJE

4.1. HTML, CSS, BOOTSTRAP – Izgled spletne strani

Sam izgled spletne strani sem ustvaril po veliki večini s HTML-jem, CSS-om in Bootstrapom, vsak od njih pa je bil namenjen svojemu delu grafičnega oblikovanja spletne aplikacije.

HTML ali Hypertext Markup Language je standardni jezik uporabljen za izdelavo dokumentov, katerih namen je biti prikazan na spletnem brskalniku. Definira vsebino in strukturo spletnih vsebin. HTML prejme podatke o tem, kaj naj prikaže iz spletnega strežnika ali pa lokalnega naslova. Elementi so osnovni gradniki HTML strani. Zapišemo jih med značke `< in >` (`<atribut>`). Večina HTML elementov ima začetno in končno značko (prva začne element, druga pa ga konča), nekaterim elementom pa zadostuje že ena značka. Te značke so najrazličnejše in nudijo široko paleto funkcionalnosti, ki jih lahko uporabimo pri ustvarjanju spletnih strani. Prva omemba HTML-ja sega v leto 1991, ko je njegov avtor Tim Berners-Lee na spletu delil dokument imenovan HTML značke. Skozi čas so za njegov razvoj različne organizacije kot so W3C (World Wide Web Consortium) in WHATWG (Web Hypertext Application Technology Working Group). Danes uporabljena verzija tega jezika se imenuje HTML5 in je bila javnosti predstavljena leta 2014. Osnovna struktura strani, realizirane s HTML-jem se začne z oznako `<!DOCTYPE html>`, sledijo `<html>`, `<head>` in `<body>`, vse od teh delov je po koncu pisanja treba tudi zapreti z `</atribut>`. Uporabnost HTML-ja se zelo poveča, ko vanj integriramo še CSS. (HTML, 2025)

CSS ali Cascading Style Sheets je jezik uporabljen za prezentacijo in stiliziranje dokumenta zapisanega v standardnem označevalnem jeziku kot je HTML. Skupno z HTML-jem in JavaScriptom tvori osnovno interneta kot ga poznamo danes. Izdelan je tako, da omogoča delitev vsebine strani in njene predstavitev. To vključuje fonte, sloje, barve in mnogo drugega. Uporabljamo ga, da lahko bolj natančno oblikujemo posamezne HTML elemente ter jih prilagodimo našim željam. CSS kodo lahko pišemo znotraj glave posameznega HTML dokumenta, v vrstici, kjer ga potrebujemo (inline) ali pa v zunanji datoteki (s končnico .css), na katero kažemo s sklicem. Splošne specifikacije programskega jezika zagotavlja World Wide Web Consortium (W3C). Prvič je bil izdan leta 1998, zadnja verzija pa je bila izdana decembra 2023. Sintaksa CSS ukazov je sestavljena iz: tipa selektorja (katere oz. koliko elementov zajema) in deklaracijske kocke `{ }` (zaviti oklepaji) znotraj katere definiramo CSS ukaze ter jim dodelimo vrednosti. (CSS, 2025)

Bootstrap je okvir (framework) za razvoj odzivnih spletnih strani ter spletnih aplikacij. Prvotno sta ga razvila Mark Otto in Jacob Thornton, danes pa ga vzdržuje skupnost na GitHubu. Namenjen je predvsem hitrejšemu in enostavnejšemu razvoju spletnih vmesnikov, saj ponuja že vnaprej pripravljene komponente in sloge, ki nam omogočajo hitrejše pisanje in uporabo HTML-ja, CSS-a in JavaScript-a. Glavna prednost Bootstrapa je to, da se z uporabo sistema mreže (grid system) vsebina samodejno prilagaja različnim velikostim zaslonov. Poleg tega vključuje tudi različne, že vnaprej pripravljene uporabne komponente, kot so navigacijske vrstice, gumbi, obrazci, opozorila in drugo. Uporaben je zato, ker zelo prihrani čas izdelave grafičnega dela projektov, saj programerji prihranijo čas, saj jim ni treba vsake komponente izdelati iz nič. Bootstrap je primeren tako za začetnike kot napredne razvijalce in je pogosto uporabljen v profesionalnem spletnem oblikovanju. (Get started with Bootstrap, 2025)

4.2. PHP in JavaScript – Dodajanje funkcionalnosti

Na spletni strani sem dodal funkcionalnost in izboljšal interaktivnost s pomočjo v veliki količini PHP-ja, nekaj malenkosti pa sem dodal tudi v JavaScriptu.

PHP (Hypertext Preprocessor) je skriptni jezik, ki se uporablja predvsem pri strežniškemu delu razvoja spletnih aplikacij. PHP se izvaja na strežniku, kar pomeni, da uporabnik vidi le rezultat (npr. HTML-kodo), ne pa same PHP kode. Uporablja se lahko za ustvarjanje dinamičnih spletnih strani, obdelavo obrazcev, delo z podatkovnimi bazami in tako dalje. PHP sam po sebi je zelo preprost in enostaven za učenje, hkrati pa je jezik dovolj zmogljiv za razvoj kompleksnih aplikacij. Ker je jezik splošno znan in uporabljen obstaja ogromno dokumentacije in knjižnic, ki olajšajo delo programerjem. PHP je združljiv s skoraj vsemi spletnimi strežniki (npr. Apache, Nginx) in operacijskimi sistemi (Windows, Linux, macOS). Pogosto se uporablja skupaj z drugimi tehnologijami v t.i. LAMP skladu: Linux, Apache, MySQL in PHP (tako sem ga uporabil tudi jaz). Velike platforme, kot so WordPress in Drupal, temeljijo na PHP-ju, kar kaže na njegovo razširjenost in zanesljivost. (What is PHP and what can it do?, 2025)

JavaScript je skriptni programski jezik, ki se uporablja za dodajanje interaktivnosti spletnim stranem. Omogoča, da se spletna stran spreminja glede na dejanja uporabnika brez ponovnega nalaganja strani – npr. pri preverjanju obrazcev, spreminjanju vsebine strani ali ustvarjanju animacij. JavaScript se izvaja na strani odjemalca (v brskalniku), kar pomeni, da deluje neposredno v napravi uporabnika. Skupaj s HTML in CSS tvori temelj sodobnega spletnega razvoja. Poleg uporabe v brskalniku se danes pogosto uporablja tudi na strežniški strani z uporabo okolja Node.js. Zaradi široke podpore, enostavne integracije in velikega števila uporabnikov je JavaScript postal eden najpomembnejših jezikov v spletnem programiranju. (Developer Mozilla, 2025)

4.3. SQL – Podatkovne baze

Za interakcijo z podatkovnimi bazami in njihovo ustvarjanje sem uporabil SQL.

SQL (Structured Query Language) je standardiziran programski jezik namenjen upravljanju in manipulaciji relacijskih podatkovnih baz. Uporablja se za izvajanje različnih operacij, kot so poizvedbe znotraj ene ali več tabel skupaj, vstavljanje elementov v tabelo, posodabljanje in brisanje podatkov ter ustvarjanje in spreminjanje strukture tabel in baz podatkov. Z uporabo SQL lahko programerji dostopajo do podatkov, jih analizirajo in povezujejo med seboj. Osnovne poizvedbe, kot je SELECT stavek, omogočajo pridobivanje specifičnih podatkov iz ene ali več tabel, medtem ko ukazni stavki kot so INSERT, UPDATE in DELETE služijo za spreminjanje vsebine baze. SQL temelji na relacijskem modelu, kjer so podatki organizirani v tabelah (vrstice in stolpci). Omogoča tudi uporabo ključev (tujih in primarnih), indeksov, pogledov (views) in drugih naprednih funkcionalnosti. Uporablja se v številnih sistemih za upravljanje baz podatkov, kot so MySQL, PostgreSQL, Microsoft SQL Server in Oracle Database. Ker je SQL standardiziran jezik, osnovna sintaksa ostaja podobna v vseh teh sistemih, čeprav obstajajo nekatere razlike in posebnosti. Zaradi svoje široke uporabe in ključne vloge v shranjevanju in obdelavi podatkov je znanje SQL zelo pomembno za razvijalce, podatkovne analitike in skrbnike baz. (SQL Tutorial, 2025)

MySQL je odprtokodni (open source) sistem za upravljanje relacijskih baz podatkov, ki temelji na jeziku SQL. Uporablja se za shranjevanje podatkov in tabel, njihovo organiziranje in pridobivanje podatkov iz njih v številnih spletnih aplikacijah. MySQL je znan po njegovi hitrosti, zanesljivosti in enostavnosti uporabe. Pogosto se uporablja v okoljih, kjer je prisoten PHP, kot je LAMP sklad (Linux, Apache, MySQL, PHP). Primeren je tako za majhne projekte kot za velike spletne aplikacije, kot je WordPress. (Why MySQL?, 2025)

4.4. Stripe API – plačevanje, PHPmailer

Stripe API (Application Programming Interface) je zmogljiv in programerjem prijazen vmesnik za upravljanje spletnih plačil in povezanih finančnih storitev. Zasnovan je bil z namenom poenostavitve vključevanja plačevanja v spletne strani, mobilne aplikacije in spletne trgovine. Uporabnikom omogoča izvajanje transakcij z različnimi plačilnimi metodami (kreditne kartice, debetne kartice, digitalne denarnice). Stripe API temelji na REST arhitekturi in uporablja standardne HTTP metode, kot so GET, POST in DELETE. To pomeni, da je združljiv z različnimi programskimi jeziki in ogrodji. Stripe ponuja tudi uradne knjižnice (SDK - software development kit) za jezike, kot so JavaScript, Python, PHP in Java, kar omogoča hitro in učinkovito integracijo. Varnost je ena ključnih prednosti Stripe-a. Vse občutljive informacije o plačilih se obdelujejo neposredno na Stripe-ovih strežnikih, ki so dobro zavarovani, kar programerjem omogoča, da se izognejo shranjevanju občutljivih podatkov in tako lažje izpolnjujejo zahteve različnih standardov. Poleg osnovnih enkratnih plačil Stripe API podpira tudi napredne funkcionalnosti, kot so naročnine, ponavljajoča se plačila, vračila, izdajanje računov in upravljanje uporabniških računov. Stripe API vključuje tudi podporo za Webhooks, s katerimi aplikacija prejema obvestila v realnem času o dogodkih, kot so uspešna plačila, zavrnitve ali povračila. Stripe Dashboard ponuja pregledno vizualno upravljanje vseh transakcij, analitiko in konfiguracijo nastavitev. Zaradi svoje zanesljivosti, prilagodljivosti in odlične dokumentacije je Stripe API ena najbolj priljubljenih rešitev za spletna plačila danes. (API Reference, 2025)

4.5. Zunanji strežnik

Kot zadnji dodatek sem spletno stran postavil na Oraclov Cloud strežnik. Strežnik na Oraclovi oblaki infrastrukturi (Oracle Cloud Infrastructure - OCI), imenovan tudi instanca (Compute Instance), je virtualni strežnik, ki omogoča izvajanje aplikacij, gostovanje spletnih strani, aplikacij in mnogo drugega. OCI strežniki temeljijo na infrastrukturi podjetja Oracle, ki ponuja visoko razpoložljivost, varnost in prilagodljivost glede na potrebe uporabnika. Uporabniki lahko izbirajo med različnimi oblikami (shapes) instanc, ki določajo število jeder, količino pomnilnika, omrežne zmogljivosti in druge vire. OCI ponuja tudi možnost uporabe brezplačnih virov v okviru Always Free programa. Pri ustvarjanju strežnika uporabnik izbere operacijski sistem (npr. Oracle Linux, drugi Linuxi ali Windows), določi konfiguracijo in varnostne nastavitve (npr. SSH ključi za dostop). Strežnik lahko nato upravlja preko konzole OCI ali se nanj poveže prek SSH (Secure Shell). Oracle strežniki omogočajo tudi samodejno spreminjanje velikosti ob potrebi, upravljanje prometa, shranjevanje podatkov in integracijo z drugimi OCI storitvami, kot so baze podatkov, varnostne kopije, omrežja in varnostne politike. (Oracle Cloud Infrastructure Documentation, 2025)

Za dostopanje do strežnika, njegovega terminala sem uporabil orodje PuTTY. PuTTY je brezplačna odprtokodna (open source) aplikacija, ki omogoča varno oddaljeno povezovanje s strežniki prek protokolov SSH, Telnet in SCP. Uporablja se predvsem za administracijo Linux strežnikov prek ukazne vrstice (terminala). PuTTY podpira tudi uporabo javnih in zasebnih ključev za prijavo ter omogoča shranjevanje nastavitev za več strežnikov. Zaradi enostavnosti in zanesljivosti je eno najbolj priljubljenih orodij za upravljanje oddaljenih sistemov. (PuTTY, 2025)

V PuTTY-ju sem nato na strežnik namestil aaPanel. aaPanel je brezplačna in odprtokodna nadzorna plošča, namenjena enostavnemu upravljanju Linux spletnih strežnikov prek grafičnega uporabniškega vmesnika (GUI – graphical user interface). Uporabnikom omogoča, da brez uporabe ukazne vrstice nameščajo in upravljaajo spletne strežnike, domene, baze podatkov, FTP račune, SSL certifikate in drugo. aaPanel podpira različne spletne tehnologije kot so Apache, Nginx, MySQL, PHP, Node.js ... aaPanel lahko na strežnik naložimo samo z enim ukazom v terminalu Po namestitvi uporabnik dobi dostop do spletnega vmesnika, kjer lahko s pomočjo modulov (one-click installs) hitro doda storitve,

kot so MySQL, phpMyAdmin ali požarni zid. . Znotraj aaPanela lahko z enim klikom naložimo tudi aplikacije LAMP (Linux, Apache, MySQL, PHP) ali LEMP (Linux, Nginx (Engine-X), MySQL, PHP) sklada. aaPanel vključuje tudi sistemsko spremljanje, varnostne funkcije, samodejno ustvarjanje varnostnih kopij in možnost dodajanja razširitev za naprednejše funkcionalnosti. Na njemu lahko pridobimo tudi SSL certifikat. (aaPanel, 2025)

Za varovanje strežnika sem uporabil spletno platformo Cloudflare. Cloudflare je oblačna storitev, ki deluje kot posrednik med obiskovalci in spletnimi strežniki ter ponuja zaščito, izboljšuje zmogljivost in razpoložljivost spletnih strani. S svojo globalno mrežo podatkovnih centrov Cloudflare filtrira škodljiv promet, ščiti pred DDoS napadi in drugimi varnostnimi grožnjami. Hkrati pospešuje nalaganje strani s pomočjo CDN (Content Delivery Network), ki shrani statične vsebine in jih dostavi iz najbližjega strežnika uporabniku. Cloudflare prav tako omogoča enostavno aktivacijo SSL certifikata, izboljšuje delovanje DNS sistema, omogoča pametno usmerjanje prometa in ponuja napredna orodja za analizo prometa. Uporabniki lahko preko nadzorne plošče prilagajajo varnostne in zmogljivostne nastavitve svojih spletnih mest. Zaradi enostavne uporabe in učinkovitosti je Cloudflare ena najbolj priljubljenih rešitev za zaščito in optimizacijo spletnih projektov. (So what is Cloudflare?, 2025)

5. OPIS IZDELAVE

5.1. Splošna predloga in indeks

Vsaka stran moje spletne aplikacije, ki je prikazana ima predlogo sestavljeno iz 2 delov: glave in noge spletne strani. Predstavljata čisto spodnji in zgornji del spletne strani, vmes pa je vsebina. Oba dela sem realiziral v obliki PHP funkcije na katero skripta, ki se prikazuje, sklicuje. Obe funkciji sta v skripti funkcije.php, kjer je shranjena velika večina PHP funkcij, ki jih uporabljam na spletni strani. Funkcija Navbar(), izpiše navigacijsko vrstico, torej glavo spletne strani. Najprej funkcija uporabi Bootstrap class navbar, da se ta inicializira. Po inicializaciji se na strogem vrhu spletne strani izpiše poziv »Sledite nam na naših družabnih omrežjih« ter na desni strani zaslona se prikažejo ikone Instagrama, Facebooka in X-a, do katerih so povezave realizirane na način spodaj. Pri prikazu ikon sem uporabil knjižnico ikon Font Awesome.

```
<a href="https://www.facebook.com/" class="link-underline link-underline-
opacity-0 link-secondary">
<i class="fa-brands fa-facebook m-2"></i>
</a>
```

Po izpisu vseh povezav se ustvari drugi del navigacijske vrstice, na začetku z značko img-src prikažem logotip spletne strani, ki je shranjen v mapi slike. Sledijo gumbi, ki uporabnike preusmerijo na podstrani spletne strani: Home preusmeri na index.php, gumb Računalniki in elementi spustnega seznama vsak preusmerijo na prikaze tipa izdelkov v imenu gumba, nato imamo povezavo do PC builder strani, sledi iskalnik, ki ga bom opisal v točki 5.6., nato gumb za prijavo in registracijo ter košarica. V kodi imajo vsi gumbi tako strukturo (pri košarici dodam samo še ikono, ki sem jo dobil v knjižnici Font Awesome):

```
<li class="nav-item">
    <a class="nav-link text-dark" href="search.php?num=5">Računalniki</a>
</li>
```

Na mestu povezave do prijave se v primeru, če je uporabnik že vpisan v račun izpiše funkcija pozdrav(). Ta dobi iz seje uporabniško ime uporabnika in ga shrani v spremenljivko \$username, nato pa s SQL poizvedbo pridobi ime uporabnika z tem uporabniškim imenom. To je razvidno spodaj:

```
$username = $_SESSION["username"];
$sql = "SELECT ime FROM rso_prijava WHERE username = '" . $username . "' LIMIT
1";
$resultat = $db->query($sql);
$user = $resultat ? $resultat->fetch_assoc() : null;
```

Nato, če je uporabnik s tem imenom najden (je v \$user zapisana neka vrednost) izpiše njegovo ime v formatu »Pozdravljen ime« in gumb, ki omogoča izpis iz računa (iz seje izbriše vrednosti vseh atributov, ki so bili v njej shranjeni). Če je \$user prazen pa izpiše napako. Če je v njegov račun prijavljen admin pa se za košarico izpiše tudi gumb, ki z usmeritvijo na dodaj.php omogoča dodajanje novih artiklov.

Za prikaz noge pa spletne strani kličejo funkcijo footer(). Ta se iniciira s Bootstrap funkcijo footer. V nogi spletne strani so prikazani: logotip trgovine, avtorske pravice, dovoljene metode plačila ter podatki o imaginarni fizični prodajalni ter kontaktnih podatkih.

V večini skript, ki se ukvarjajo s prikazovanjem strani je na začetku uporabljen ukaz session_start(), ki začne funkcijo ter skripta nato zahteva dostop do funkcije.php in db.php. Navadno to izgleda tako:

```
session_start();
require_once('funkcije.php');
require_once('db.php');
```

Indeks spletne strani je prva stran, na katero je uporabnik preusmerjen po izbiri spletne strani na internetu. Moj indeks ima uporabnika pozdravi ter mu prikaže gumb, ki uporabnika preusmeri na prikaz izdelkov:

```
<a href="search.php?num=5" class="btn btn-success">Začnite z nakupovanjem</a>
```

Spodaj so prikazani 3 naključni izdelki, ki so vsi iz različnih tabel, da se izognem ponavljanju izdelkov. Tabela in izdelek sta izbrana naključno znotraj for zanke, ki izbere 3 naključne številke \$num od 1 do 6, ki se nato prevedejo v tabele (st=tip tabele). Po ustvarjenju ene številke, je ta dodana v tabelo \$ze_obstaja_num. Za nadaljnje iteracije zanke se po naključnem generiranju števila preveri, če je to že shranjeno v zgornji tabeli in če je, se številka izbere ponovno:

```
for ($i = 0; $i < 3; $i++) {
    do {
        $num = rand(1, 6);
    } while (in_array($num, $ze_obstaja_num));
    $ze_obstaja_num[] = $num;
```

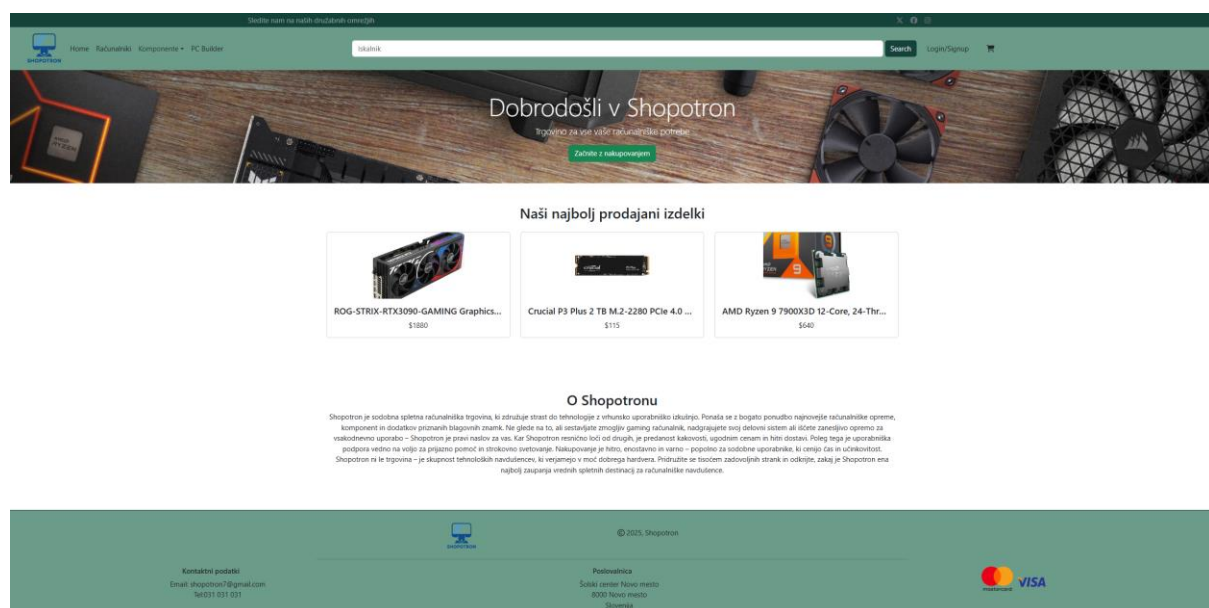
Prav tako je naključno izbran id. Nato se preko switch stavka glede na \$num za vsak posamezen artikel, ki je prikazan izbere tabela iz katere je. Sledi poizvedba, ki dobi podatke o artiklu, ki ga prikazujemo. Na spletni strani se prikažejo 3 stolpci, vsak prikazuje 1 izdelek. Ti so prikazani v obliki kartice in vsebujejo ime izdelka, sliko in ceno. Omogočeno je tudi, da je uporabnik ob kliku na izdelek preusmerjen na stran, kjer je prikazan v podrobnostih. Slika samega izdelka pa je prikazana drugače kot navadno, saj je znotraj tabele iz katere je izdelek, ki je prikazan, slika zakodirana z base64 kodiranjem:

```

```

Od slik je prikazana prva slika izdelka, ki ga prikazujemo. Sledi opis trgovine.

Noga in glava, skupaj z indeksom izgledajo tako:



Slika 6: Indeks z glavo in nogo

5.2. Podatkovna baza

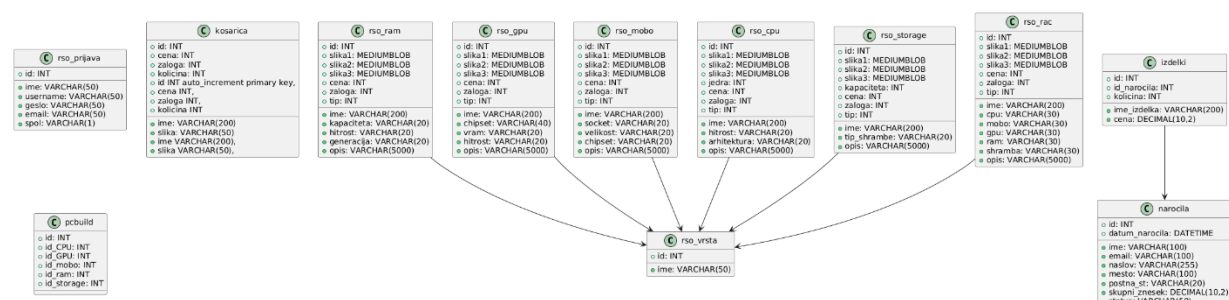
Podatkovna baza je bila oblikovana v večini na začetku izdelave, nekaj pa tudi sproti. Vsebuje tabele v katerih so shranjeni izdelki z njihovimi atributi. To je rso_cpu, ki vsebuje procesorje, z unikatnimi atributi jedra, hitrost in arhitektura. Ta in vse ostale tabele, ki vsebujejo izdelke imajo attribute id, ime, slika1, slika2, slika3, cena, zaloga, opis in tip primarni ključ id ter tuji ključ tip, ki kaže na tabelo rso_vrsta, ki po tipu izdelka pove iz katere tabele je. Take so tudi tabele rso_gpu, ki vsebuje grafične kartice z atributi vram, hitrost in chipset, rso_ram, ki vsebuje RAM-e, z atributi kapaciteta, hitrost, generacija, rso_rac, ki vsebuje sestavljene računalnike, z unikatnimi atributi id, cpu, gpu, ram in shramba, tabela rso_storage, ki vsebuje shrambe, z unikatnimi atributi kapaciteta in tip_shrambe. Zadnja tabela z izdelki je rso_mobo, ki vsebuje matične plošče, z unikatnimi atributi socket, velikost in chipset. V tabelo rso_prijava se shranijo prijavni podatki uporabnikov, to so njihovo ime, email, geslo, uporabniško ime in spol. V tabelo kosarica uporabnik shranjuje izdelke, ki jih ima namen nakupiti. Vanjo se shranijo id, ime izdelka, slika, cena, kolicina (količina izdelkov dodanih v košarico). V tabelo narocila se shranjujejo podatki o kupcu izdelkov. Ima attribute id, ime, email, naslov, mesto, postna_st, skupni_znesek, datum_narocila in status (narocila). Dopolnjuje jo tabela izdelki, v katero se vpiše posamezen artikel, ki ga je kupil kupec, vsebuje id, id_narocila (tuji ključ, ki kaže na id v tabeli narocila). Sledi tabela pcbuild v katero se shranjujejo izdelki iz katerih je sestavljen računalnik, ki ga uporabnik gradi na spletni strani (skripta pcbuild.php). Vsebuje lasten id ter id_CPU, id_GPU, id_mobo, id_ram in id_storage. Vse te tabele skupaj tvorijo podatkovno bazo, ki zadostuje potrebam moje spletne aplikacije. Vse tabele so narejene s shranitvijo SQL ukaza v spremenljivko \$sql.

```
$sql="CREATE TABLE rso_prijava (
    id INT auto_increment primary key,
    ime VARCHAR(50),
    username VARCHAR(50),
    geslo VARCHAR(50),
    email VARCHAR(50),
    spol VARCHAR(1)
);";
```

Nato se izvede poizvedba po podatkovni bazi (povezava do nje je v skripti db.php) ter izpiše se uspešnost/neuspešnost izvedbe ukaza.

```
if($db->query($sql))
    echo("Tabela rso_prijava je bila uspesno ustvarjena.");
else
    echo("Napaka pri ustvarjanju tabele.");
```

Spodaj na shemi UML diagrama, narejeni s spletno aplikacijo PlantUML je razvidna celotna tabela:



Slika 7: Končni razredni diagram

5.3. Dodajanje vsebine (admin)

Na vrhu strani, ki jo izpiše dodaj.php si admin lahko izbere kater tip izdelka od možnih bo dodajal (računalnik, RAM, procesor, grafično kartico, matično ploščo ali pa shrambo). To storijo tako, da jih gumb ob pritisku preusmeri na dodaj.php, zraven pa pripne tudi id, ki pove katero podstran se prikaže. Vsak gumb je napisan tako:

```
<button class="btn siv mx-2" onclick="window.location.href='dodaj.php?id=1'">
Dodaj CPU</button>
```

Spreminjajo se samo id-ji in tip izdelka, ki ga dodajamo. Pri vsaki od podstrani se nato preko URL-ja pridobi id, ki se shrani v spremenljivko \$id:

```
$id = $_GET['id'] ?? null;
```

Nato sledi 6 if stavkov, ki preverjajo kater id je izbran. Tisti, ki je izbran nato izpiše formo za dodajanje elementa znotraj kartice (Bootstrap elementa). Forma za vsakega od izdelkov je sestavljena iz več polj v katere mora admin nato vpisati ali izbrati zahtevane podatke. Vsako posamezno polje ustvarim tako:

```
<div class="form-floating mb-3">
<input type="text" class="form-control" name="ime" placeholder="Ime" required>
<label for="ime">Ime izdelka</label>
</div>
```

Pri vsaki komponenti mora admin vnesti ime, unikatne attribute vsakega izdelka, njegovo ceno, zalogo in njegov opis. Admin mora tudi naložiti 3 slike, ki bodo uporabljene v prikazih, za te, mora izbrati slike iz njegovega lastnega računalnika ter jih naložiti v formo. Pri oblikovanju tega zelo pomaga sam HTML, saj večino oblikovanja olajša atribut file, ki ga podamo znotraj značke input. Na koncu vsake forme je tudi gumb za potrditev oddaje:

```
<button type="submit" class="btn btn-success w-100">Dodaj CPU</button>
```

Na koncu vsakega if stavka sledi klic funkcije addImeTipaIzdelka(), zapisana v funkcije.php.

Vsaka od teh funkcij nato doda izdelek v podatkovno bazo. To naredi tako, da naprej v lokalne spremenljivke shrani podatke posredovane funkciji s formo, ki je bila oddana na dodaj.php. Nato funkcija preveri, če so vse slike veljavne in naložene v celoti. Če to ni res, izpiše napako. Sledi pripravljanje SQL stavka, ki bo v ustrezno tabelo dodal nov izdelek, nato se v stavek pripnejo parametri in se izvede:

```
$stmt = $db->prepare("INSERT INTO rso_cpu (ime, slika1, slika2, slika3, jedra,
hitrost, arhitektura, cena, zaloga, opis) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

$stmt->bind_param("ssssiss", $ime, $slika1, $slika2, $slika3, $jedra,
$hitrost, $arhitektura, $cena, $zaloga, $opis);
$stmt->execute();
```

Ob napaki izvajanja, program vrne napako. Taka funkcija je pripravljena za vsak tip izdelka, ki ga uporabnik lahko doda. Funkcionalni del spletne strani izgleda tako:

Slika 8: Stran za dodajanje

5.4. Prijava in registracija

Prijava in registracija potekata na skriptah `login.php` in `signup.php`. V `login.php` se lahko uporabniki prijavijo v njihove že obstoječe račune, ki so jih predhodno ustvarili na strani `signup.php`. Vpis podatkov poteka znotraj kartice, kjer je ustvarjena forma v katero morajo uporabniki vpisati uporabniško ime in geslo njihovega računa, nato pa pritisniti gumb prijava, ki formo pošlje na skripto `prijava.php`. Element forme je ustvarjen tako:

```
<div class="mb-3">
  <label for="geslo" class="form-label">Geslo</label>
  <input type="password" class="form-control" name="geslo" id="geslo"
placeholder="Vnesite geslo" required>
</div>
```

Če uporabnik še nima računa, je lahko preusmerjen na stran `signup.php`. Ko so podatki poslani na `prijava.php`, so tam shranjeni v spremenljivki `$username` in `$geslo`:

```
$username=$_POST['username'];
$geslo=$_POST['geslo'];
```

Nato ustvarimo SQL poizvedbo, kjer v tabeli `prijava` poiščemo geslo, tam, kjer je uporabniško ime enako `$username`. Če je poizvedba uspešna, skripto preveri, če je geslo pridobljeno s iskanjem po podatkovni tabeli enako tistemu, ki ga je vpisal uporabnik. Če je to res, se v sejo shranita uporabniško ime in geslo:

```
if ($user['geslo'] == $geslo) {
  $_SESSION['username']=$username;
```



```
$_SESSION['geslo']=$geslo;
header("Location: index.php");
```

Če pa to ni res, skripta izpiše napako.

Registracija poteka na strani signup.php, kjer ponovno v obliki kartice od uporabnika zahtevamo vnos podatkov za registracijo, tukaj mora uporabnik vpisati svoje ime, uporabniško ime, geslo, email in spol. Nato lahko za potrditev vpisa pritisne potrditveni gumb. Prav tako ima omogočen prehod na stran login.php, če je ugotovil, da račun že ima. Ko pritisne na gumb potrdi, se podatki z pošljejo naprej z metodo post. V nadaljnjem delu skripte, PHP koda preverja, če smo kaj poslali z metodo POST (smo potrdili registracijo). Če je to res, v sejo shrani vse podatke, ki jih je dobil preko forme. Prav tako pa v spremenljivko \$koda shrani vsebino funkcije ustvariKodo(), ki ustvari naključno kodo dolžine 32 znakov, tudi \$koda je nato shranjena v sejo. Nato je uporabnik preusmerjen na confirm.php. Tukaj mora uporabnik v formo znotraj kartice vpisati 32 mestno kodo, ki jo je prejel na e-pošto, uporabljeno pri registraciji. Vpis kode lahko tudi potrdi. Znotraj PHP kode se v spremenljivko \$koda shrani vsebina kode shranjene v seji:

```
$koda = $_SESSION['koda'];
```

Nato se uporabniku pošlje mail. To se naredi s pomočjo PHPmailerja, knjižnice, namenjene pošiljanju enostavnih email sporočil. Najprej ustvarimo instanco objekta PHPmailer z ukazom:

```
$mail = new PHPMailer(true);
```

Nato znotraj te instance definiramo metodo kodiranja (UTF-8), omogočimo pošiljanje preko SMTP (Simple Mail Transfer Protocol). Kot gostitelja SMTP uporabnikov gmailov strežnik smtp.gmail.php. Dodatno omogočimo SMTP avtentikacijo, ter pod uporabniško ime vpišemo naš mail shopotron7@gmail.com ter geslo zanj (cpfw qnpm nvrk kxsu). Omogočimo tudi tls zakodiranje ter povemo skozi katera vrata strežnika naj se poveže na SMTP strežnik (587). Dodamo še pošiljatelja, ki je shopotron7@gmail.com in naslovnika, ki ga pridobimo iz seje (email uporabika, ki se registrira je še vedno shranjen v seji). Nato povemo, da bo mail oblikovan s pomočjo HTML-ja ter dodamo zadevo maila ter njegovo vsebino. Na koncu to pošljemo. Večina stavkov s katerimi vpisujemo v \$mail podatke je formatirana v stilu:

```
$mail->Username = 'shopotron7@gmail.com';
```

V vsebino maila dodamo tudi kodo, ki smo jo ustvarili v signup.php. Nato v \$vpisanaKoda shranimo kodo, ki smo jo v sejo shranili v signup.php. Potem v if stavku preverimo, če je \$vpisanaKoda enaka \$koda (če je koda, ki jo je vpisal uporabnik enaka tisti, ki smo mu jo poslali po emailu). Če je, v spremenljivke shranimo podatke s katerimi se uporabnik registrira ter jih nato z SQL stavkom vstavimo v rso_prijava:

```
$sql = "INSERT INTO rso_prijava (ime, username, geslo, email, spol) VALUES
('$ime', '$username', '$geslo', '$email', '$spol')";
```

Nato poizvedbo izvedemo ter, če je ta uspešna, v sejo dodamo uporabniško ime uporabnika ter geslo in ga s tem prijavimo v njegov račun. Iz seje nato odstranimo vse nepotrebne spremenljivke. Če poizvedba ni bila uspešna ali pa je uporabnik vpisal napačno kodo vrnemo napako. Formi za prijavo in registracijo izgledata tako:

Slika 9: Prijava

Slika 10: Registracija

5.5. Prikaz izdelkov

Izdelke znotraj spletne aplikacije prikazujem na 2 načina: kot vse izdelke posameznega tipa skupaj, ko uporabniki kliknejo na kakšno od povezav do izdelkov v navigacijski vrstici (search.php) ali vsak izdelek v detajlih posebej (prikaz.php), ko uporabnik s klikom izbere izdelek katerega si želi natančno ogledati. Znotraj search.php se v spremenljivko \$num shrani vrednost num, ki je bila v search poslana preko URL-ja s klikom na povezavo do tipa artikla. Num je vrednost med 1 in 6, ki pove iz katere tabele oziroma katerega tipa so artikli, ki jih trenutno prikazujemo. Nato ustvarimo poizvedbo s katero iz rso_vrsta dobimo ime tabele po kateri rabimo iskati izdelke, ki jih bomo prikazovali:

```
$sql = "SELECT ime FROM rso_vrsta WHERE id=$num limit 1";
```

To poizvedbo tudi izvedemo, nato pa v \$tabela shranimo ime tabele, katere izdelke moramo prikazati (rso_+vrednost imena iz zgornjega stavka). Nato z funkcijo COUNT znotraj novega SQL stavka ugotovimo koliko izdelkov moramo prikazati, če želimo prikazati vse. Rezultat poizvedbe nato vstavimo v for zanko (v obliki spremenljivke \$user v katero smo shranili rezultat zgornje poizvedbe) kot maksimalna vrednost, do katere se bo zanka izvajala, torej for zanka izgleda tako:

```
for($i=1; $i<$user+1; $i++){
```

\$user-ju znotraj definicije te zanke prištejemo 1, zato, da se bo zanka nehala izvajati šele po izpisu zadnjega od artiklov. Znotraj for zanke napišemo nov SQL stavek, ki izbere vse podatke artikla ki ima id, ki je enak \$i (id vsakega elementa znotraj tabele). Rezultat te poizvedbe v obliki asociativne tabele shranimo v \$user1. Nato na spletno stran znotraj kartice prikažemo pravokotnik v katerem bomo prikazali posamezen izdelek, obdamo ga tudi s povezavo do natančnega opisa izdelka:

```
<a href='prikaz.php?id=$i&num=$num' class='text-decoration-none text-body'>
```

Pri tem pa moramo upoštevati, da uporabnika ne bomo samo preusmerili, temveč mu bomo tudi dodali id in num, ki bosta natančno določila do katerega artikla znotraj tabele vodi povezava. Znotraj pravokotnika prikažemo sliko izdelka na enak način kot v indeksu, dodamo še ime izdelka, njegovo ceno in zalogo tega izdelka.

Ko uporabnik klikne na določen izdelek znotraj search.php, je preusmerjen na stran prikaz.php skupno z id-jem in num-om. Prva stvar, ki jo skripta naredi, je to da preveri če obstaja kakšno sporočilo znotraj atributa toast_message. Če to obstaja, pokliče funkcijo toast, ki je definirana znotraj funkcije.php. Ta funkcija uporabi Bootstrap class toast, da prikaže sporočilo, ki uporabniku pove, kar programer vstavi

vanjo. Toast sporočilo je sestavljeno iz glavnega dela s sporočilom ter iz gumba, ki omogoča zapiranje toasta. Nato v spremenljivki \$num in \$id shrani vrednosti, kateri smo v to skripto poslali iz prikaz.php. Nato skripta potrdi, če obstajata \$num in \$id ter če sta obe vrednosti številski. Če nista, skripta vrne napako, da manjkajo parametri in konča izvajanje funkcije. Nato s pomočjo \$num, tako kot na prikaz.php dobimo ime tabele iz katere je element, ki ga prikazujemo. Sledi nova SQL poizvedba, kjer iz tabele pridobljene malo prej dobimo vse podatke o tistem elementu, ki ima id enak \$id. Ta poizvedba se nato izvede. Struktura SQL poizvedbe je zaradi varnosti taka:

```
$sql = "SELECT * FROM $tabela WHERE id = ? LIMIT 1";
$stmt2 = $db->prepare($sql);
$stmt2->bind_param("i", $id);
$stmt2->execute();
$rezultat = $stmt2->get_result();
$user = $rezultat->fetch_assoc();
```

Pri tem se najprej ustvari poizvedba, ki jo pripravimo na izvajanje, nato vanjo pripnemo parametre (pri čemer moramo dodati njihov podatkovni tip), potem pa se poizvedba šele izvede. Rezultat le-te se shrani v spremenljivko \$user kot asociativna tabela. Če je ta poizvedba uspešna se izpiše funkcija carousel, ki ustvari tako imenovan »vrtljak« slik (vse 3 slike, ki so shranjene znotraj). Struktura vrtljaka se začne z klicom class-a carousel. Nato temu dodamo ikone, ki prikažejo na kateri sliki smo, te so tipa button:

```
<button type="button" data-bs-target="#id" data-bs-slide-to="0"
class="active"></button>
```

Sledi jedro vrtljaka, za prikaz vsake slike tukaj uporabimo funkcijo prikaziSliko. Ta prejme id izdelka, ki ga prikazujemo, tabelo iz katere je ta izdelek ter podatek ali prikazujemo prvo, drugo ali tretjo sliko. Funkcija nato preveri, če so podatki ustrezni z preverjanjem če je tabela enaka kateri izmed možnih tabel, shranjenih v tabeli, enako velja za slike. To naredimo z ukazom in array:

```
if(in_array($tabela, $tabele) && in_array($slika, $nums) && $id > 0)
```

Če je zgornji if stavek pravilen, se izvede spodnja koda:

```
$sql = "SELECT * FROM $tabela WHERE id = $id LIMIT 1";
$rezultat = ($db->query($sql));
$user = $rezultat->fetch_assoc();
return '';
```

Najprej izvedemo poizvedbo s katero dobimo vrednost slike, ki jo prikazujemo znotraj podane tabele. Sliko pri prikazu ponovno dekodiramo s pomočjo ukaza base64_encode(). Ta prikaže sliko shranjeno na mestu \$slika znotraj poizvedbe, ki pove vse o iskanem artiklu. Vrtljak to funkcijo uporabi trikrat, vsakič za eno sliko, shranjeno znotraj izdelka, ki ga prikazujemo. Posamezen prikaz slike znotraj vrtljaka izgleda tako:

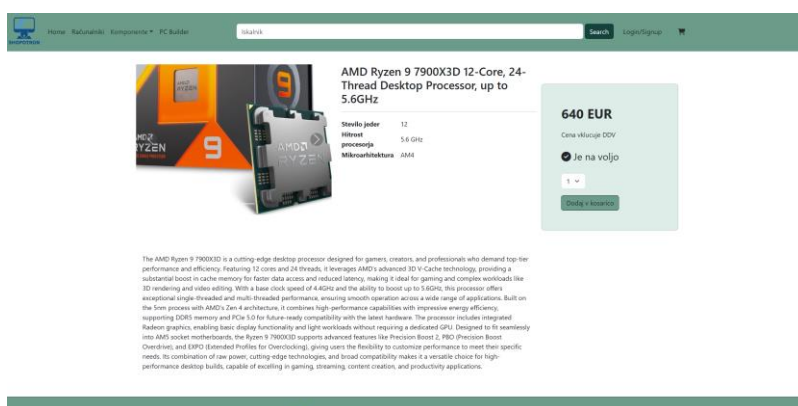
```
<div class="carousel-item">
'.prikaziSliko($id, $tabela, 'slika2').'
</div>
```

Nato pa znotraj vrtljaka dodamo še gumbe s katerimi se lahko premikamo med slikami, ki jih ta prikazuje. Po vrtljaku se izpiše tudi ime izdelka, ki ga prikazujemo. Sledi 6 if stavkov, vsak za določen tip izdelka. Vase sprejmejo \$num ter glede na tip tabele prikažejo tiste unikatne attribute, ki jih elementi tabele imajo. Pri procesorjih je to št. jeder, hitrost in arhitektura, pri grafičnih karticah je to chipset, vram in hitrost... Po izpisu teh unikatnih atributov se izpiše opis izdelka. Nato na strani ustvarimo kartico, v katero izpišemo ceno izdelka ter nato, če je na zalogi več kot 1 izdelek, izpišemo, da je izdelek na voljo. Dodamo še polje za dodajanje izdelkov v košarico. Pri tem uporabnik lahko izbere do 5 izdelkov, ki jih lahko na enkrat doda v košarico. Izbira količine je del forme, ki jo potrdimo s pritiskom

na gumb dodaj v košarico. K formi poleg potrebne količine izdelkov, dodamo še vrednosti id-ja izdelka, njegovo ime in tabelo iz katere prihaja. Če pa je na voljo manj kot 1 izdelek, se prikaže samo sporočilo da izdelek ni na voljo. Kot primer lahko prikažem prikaz procesorjev ter specifičnega procesorja:



Slika 11: Prikaz izdelkov



Slika 12: Podroben prikaz izdelka

5.6. Iskanje

Iskanje je realizirano preko skripte iskanje.php. Ta s potrditvijo iskanja (forme) v navigacijski vrstici katerekoli strani pridobi podatke o iskanemu nizu. Znotraj tabele najprej definiramo vrednost \$a, ki služi kot števec vseh izdelkov, ki bodo ustrezali poizvedbi. Nato v \$poizvedba shranimo poizvedbo, ki jo pridobimo z metodo \$_GET preko URL-ja. Ustvarimo tudi tabelo \$tabele, ki vsebuje imena vseh podatkovnih tabel z izdelki. Za vsako tabelo znotraj \$tabele nato pokličemo funkcijo izpišiRezultate, ki je definirana v funkcije.php. Funkcija, ki zahteva povezavo do podatkovne baze, ime tabele po kateri iščemo, poizvedbo ter \$a. Znotraj najprej izvedemo poizvedbo, ki iz določene tabele pridobi vse rezultate, kjer ime izdelka ustreza poizvedbi:

```
$sql = "SELECT id, slika1, ime FROM $tabela WHERE ime LIKE '%$poizvedba%'";
```

Nato pa, če je v rezultatu, ki ga dobimo število vrst večje kot 0, za vsako vrstico (vsak izdelek) izpišemo pravokotnik, ki ga obdamo s povezavo do prikaza izdelka, ki smo ga našli. Dodamo mu še ime in povečamo \$a za 1. Če je po vseh izvedenih klicih funkcije izpišiRezultate \$a enak 0, potem se izpiše, da tej poizvedbi ne ustreza noben izdelek.

5.7. Košarica

Uporabnik spletne aplikacije lahko na strani prikaz.php izbere izdelke in jih doda v košarico. To je realizirano s skripto dodaj_v_kosarico.php. V njej najprej iz forme na strani prikaz.php, ki smo jo oddali s pritiskom na gumb Dodaj v košarico, pridobimo podatke in jih shranimo v svoje spremenljivke (\$id, \$tabela, \$ime, \$kolicina). Nato ustvarimo poizvedbo, ki znotraj že obstoječe tabele kosarica preverimo, če že obstaja vnos, ki ima za atribut ime, ki je enako tistemu od artikla, ki ga ravnokar dodajamo. Rezultat poizvedbe shranimo v \$user. Potem z if stavkom preverimo, če je \$user prazen, to pomeni, da znotraj košarice še ni tega artikla in lahko znotraj stavka ustvarimo poizvedbo, ki iz \$tabela izbere vse podatke artikla, ki ima id enak \$id (artikel, ki ga dodajamo v košarico). Rezultate poizvedbe shranimo v \$user1. V lokalne spremenljivke \$slika, \$scena in \$num nato shranimo vrednosti slike1, cene in tipa izdelka, zapisanega v \$user1. Nato v košarico vstavimo izdelek s pomočjo SQL insert stavka. Kot attribute vstavimo \$ime, \$slika, \$scena, \$kolicina in \$tabela. To naredimo s pripravo SQL stavka, nato šele vstavimo attribute ter poizvedbo izvedemo:

```
$stmt2 = $db->prepare("INSERT INTO kosarica (ime, slika, cena, kolicina, tabela)
VALUES (?, ?, ?, ?, ?)");
$stmt2->bind_param("sbis", $ime, $null, $cena, $kolicina, $tabela);
$stmt2->send_long_data(1, $slika);
$stmt2->execute();
```

Nato v sejo v atribut toast_message shranimo obvestilo, da je bil izdelek uspešno dodan v košarico in uporabnika preusmerimo na lokacijo, kjer je izdelek prikazan (kjer je uporabnik izdelek dodal v košarico). To sporočilo se bo prikazalo po preusmeritvi in uporabniku povedalo, da je izdelek uspešno dodal v košarico. Če pa \$user ni prazen, pa prav tako kot v zgornjem primeru izberemo izdelek iz \$tabela na id-ju, ki je enak \$id. Nato pa posodobimo količino izdelka, ki ga dodajamo znotraj košarice tako, da že obstoječi količini prištejemo \$kolicina. To naredimo na podoben način kot zgoraj:

```
$stmt3 = $db->prepare("UPDATE kosarica SET kolicina = kolicina + ? WHERE ime =
?");
$stmt3->bind_param("is", $kolicina, $ime);
$stmt3->execute();
```

Nato pa tudi tukaj v atribut seje toast_message dodamo sporočilo, da je bil izdelek uspešno dodan v košarico ter uporabnika preusmerimo na stran iz katere je prišel. Znotraj dodaj_v_kosarico.php ob dodajanju izdelkov v košarico tudi zmanjšamo količino izdelkov v prvotni tabeli, iz katere prihaja omenjeni izdelek.

Košarico prikazujemo s pomočjo skripte shopping_cart.php. Na začetku te skripte najprej izberemo vse izdelke iz tabele kosarica, da jih lahko kasneje prikažemo in ta rezultat shranimo v \$rezultat. Nato izračunamo še skupni znesek vseh izdelkov znotraj kosarice, da ga lahko kasneje izpišemo, s tem SQL stavkom:

```
$sql_total = "SELECT SUM(cena * kolicina) AS skupna_cena FROM kosarica";
```



Znotraj glavnega dela kode ustvarimo kartico, v kateri bomo izpisali vse izdelke, ki so trenutno v košarici. V telesu kartice s pomočjo if stavka preverimo, če je znotraj košarice vsak 1 element tako, da preverimo, če je znotraj \$rezultat vsaj 1 vrstica. Če je to res, ustvarimo tabelo s katero bomo izdelke prikazali ter v njeni glavi naštejemo imena atributov izdelkov, ki jih bomo prikazovali. To so slika, ime izdelka, količina, cena enega kosa izdelka in skupna cena vseh kosov izdelka skupaj. Znotraj pogoja

while zanke nato \$rezultat shranimo v spremenljivko \$user v obliki asociativne tabele. V sami zanki bomo prikazali posamičen izdelek tako, da bomo, ko se bo zanka sprehodila skozi artikole, prikazali vso vsebino košarice. Znotraj zanke najprej iz s pomočjo SQL stavka dobimo vse podatke o izdelku, ki ga prikazujemo s pomočjo njegovega imena in tabele iz katere prihaja. Rezultat poizvedbe nato shranimo v obliki asociativne tabele v \$user1. Nato ustvarimo vrstico tabele, v kateri bomo prikazali izdelek. Znotraj tega ustvarimo stolpec v katerem prikažemo sliko, objeto v povezavo do strani, kjer je izdelek, ki ga prikazujemo, prikazan v detajlih:

```
<a href="prikaz.php?id=<?php echo $user1['id']; ?>&num=<?php echo $user1['tip']; ?>" class="text-decoration-none text-body">
<?php echo ''; ?>
</a>
```

Nato na tak način izpišemo tudi njegovo ime, ki prav tako omogoča preusmeritev. Dodamo še količino izdelka v košarici, ceno enega kosa ter skupno ceno tega izdelka (cena * količina). Na koncu vrstice pa prikažemo gumb, ki omogoči odstranjevanje izdelka s pomočjo skripte odstrani_iz_tabele.php. Nato se while zanka konča, pod zadnjo vrstico izpišemo skupni znesek, vseh izdelkov v tabeli, izračunan na začetku skripte. Pod tabelo dodamo še gumb Na blagajno, ki uporabnika preusmeri na checkout.php, kjer lahko uporabnik zaključi nakup. Zraven je še možnost, da se uporabnik vrne v trgovino z gumbom Nazaj v trgovina, ki uporabnika vrne na index.php.

Znotraj skripte odstrani_iz_tabele.php najprej dobimo id izdelka, ki ga odstranjujemo iz forme, če tega ni vrnemo napako. Če pa id obstaja, ga shranimo v spremenljivko \$id. Iz košarice nato z SQL select stavkom izberemo vse izdelek z id-jem, ki je enak \$id. Iz izdelka v lokalne spremenljivke shranimo vrednosti tabele, iz katere prihaja izdelek, ime izdelka, in količino. Nato posodobimo količino izdelka, ki ga odstranjujemo iz košarice v tabeli, kjer je bil osnovno (rso_cpu, rso_gpu...), s SQL update stavkom. To naredimo tako, da povečamo zalogo za \$skolicina (lokalna spremenljivka, v kateri je shranjena v količina izdelka iz košarice). Izdelek potem odstranimo iz kosarice s SQL delete stavkom in uporabnika preusmerimo na shopping_cart.php. Košarica z nekaj izdelki izgleda tako:

Vaša košarica					
Slika	Izdelek	Količina	Cena	Skupaj	
	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz	3	640.00 €	1,920.00 €	<button>Odstrani</button>
	Gigabyte B550 GAMING X V2 ATX Motherboard for AMD AM4 CPUs	1	105.00 €	105.00 €	<button>Odstrani</button>
Skupno				2,025.00 €	
<button>Na blagajno</button>					

Slika 13: Košarica

5.8. Plačevanje (integracija Stripe API-ja)

Uporabnik je ob pritisku gumba Na blagajno preusmerjen na stran checkout.php. Tukaj najprej izračunamo skupno ceno vseh izdelkov in rezultat shranimo v \$skupno, nato pa ta rezultat zaokrožimo na 2 decimalni mesti (dobimo ceno v centih) in to shranimo v \$znesek. Pripravimo in izvedemo tudi

poizvedbo s katero pridobimo iz košarice vse izdelke. Ustvarimo kartico in v njenem telesu tako kot pri `shopping_cart.php` ustvarimo tabelo ter v njeno glavo vstavimo atribut. Ti so ime izdelka, količina, cena in skupni znesek ($\text{cena} * \text{količina}$). Kot v `shopping_cart.php` tudi tukaj znotraj `while` zanke izpišemo ime, količino ceno in skupni znesek vseh izdelkov v košarici. Pod izpisom vseh izdelkov dodamo še njihovo skupno ceno. Pod kartico, ki prikazuje izdelke, ustvarimo novo v katero morajo uporabniki vnesti podatke o plačilnem sredstvu, ki ga uporabljajo. V telesu kartice ustvarimo formo s polji v katere uporabniki vnesejo podatke o bančni kartici. Formi dodamo še skrita polja v katere se bodo vnesli podatki, ki jih bo uporabnik vnesel v tretjo formo, ki je na desni strani zaslona. Ta bo vsebovala podatke o uporabnikovem imenu, naslovu (ulici), poštni številki, mestu in e pošti, na katero bo dobil potrdilo nakupa v primeru, da bo naročilo uspešno oddano. Doda se še skupni znesek naročila. Skrito polje izgleda tako:

```
<input type="hidden" name="ime" id="hidden-ime">
```

Nato ustvarimo tretjo formo v tretji kartici v kateri bo uporabnik dodal zgoraj omenjene podatke. Posamezno polje izgleda tako:

```
<div class="mb-3">
  <label for="ime" class="form-label">Ime in priimek</label>
  <input type="text" class="form-control" id="ime" required>
</div>
```

Po koncu te forme omogočimo pisanje JavaScript kode s značko `<script>`. V njej ustvarimo spremenljivko stripe tipa Stripe, ki ji za atribut dodamo javni ključ, ki smo ga dobili ob registraciji v Stripe-ovo spletno stran. Stripe nam omogoča poenostavljeno plačevanje z enostavno integracijo v našo spletno stran. Vsa koda znotraj JavaScript značk je namenjena njemu. Ustvarimo tudi spremenljivko elemente z Stripovo metodo `elements`. Ustvarimo tudi kartico, ki jo priprimo na DOM (document object model) element z ID-jem `card-element`. Omogočimo tudi prikazovanje napak s spodnjo kodo:

```
card.addEventListener('change', function(event) {
  var displayError = document.getElementById('card-errors');
  if (event.error) {
    displayError.textContent = event.error.message;
  }
});
```

Ta ustvari spremenljivko `displayError` in vanjo shranjuje napake. Sledi polje za obravnavanje oddaje obrazca. Ustvari spremenljivko `form`, ki vsebuje formo z ID-jem `payment-form`. Ko kupec formo odda, se ustvari objekt z vsemi informacijami o oddaji. V formo nato shranimo vrednosti podatkov pridobljenih v formi oddani ob potrditvi plačila. Dodajanje izgleda tako:

```
document.getElementById('hidden-ime').value =
document.getElementById('ime').value;
```

Znotraj Stripa ustvarimo tudi plačilno metodo tipa `card` (kartica). To naredimo s spodnjo kodo:

```
stripe.createPaymentMethod({
  type: 'card',
  card: card,
  billing_details: {
    name: document.getElementById('ime').value,
    email: document.getElementById('email').value
  }
});
```

Kjer v `billing_details` shranimo ime in email uporabnika. Nato izvedemo funkcija, ki odda plačilo se izvede funkcija, ki prejme za parameter rezultat zgornje kreacije metode. Če ta funkcija za rezultat prejme neuspešno kreacijo, potem izpiše napako, ki je kreacijo preprečila. Če je bila kreacija uspešna, potem pa kot skriti vnos v zgornjo formo shranijo tudi podatki o plačilni metodi. Forma se pa nato pošlje.

Prejme jo skripta `process_payment.php`. Ta v lokalne spremenljivke `$znesek`, `$id_placilne_metode`, `$ime`, `$email`, `$naslov`, `$mesto`, `$postna_st` shrani njihove vrednosti, pridobljene preko forme. Vsa nadaljnja koda se nahaja znotraj try dela try-catch strukture. Stripe nato ustvari kupca (`$customer`) in v njega doda uporabnikovo ime, email, ID plačilne metode, naslov mesto in poštno številko, sam pa za državo določi Slovenijo. V spremenljivko `$paymentIntent` se shrani tudi t.i. plačilni izpisek, v katerega shranimo znesek, kupcu dodamo ID, ID plačilne metode, omogočimo ročno potrjevanje plačila, dodamo opis ter dodamo lokacijo, kamor se vrne uporabnik po plačilu (pri meni je to skripta `payment_success.php`). Nato preverimo status `$paymentIntent`. Če je bil ta uspešen v spremenljivko `$id_narocila` shranimo vrednost funkcije `shraniNarocilo`, definirano v `funkcije.php`. Funkcija kot parametre prejme podatkovno bazo, ime kupca, njegov email, naslov, mesto in poštno številko. Ta funkcija v tabelo `narocila` vstavi kupčevo naročilo s spodnjim stavkom:

```
$stmt = $db->prepare("INSERT INTO narocila (ime, email, naslov, mesto, postna_st, skupni_znesek, datum_narocila) VALUES (?, ?, ?, ?, ?, ?, NOW())");
$stmt->bind_param("sssssd", $ime, $email, $naslov, $mesto, $postna_st, $skupno);
$stmt->execute();
```

Ta stavek je zaradi varnosti najprej pripravljen, nato pa šele izveden. V `$id_narocila`, ki ga kasneje funkcija tudi vrne se shrani id naročila. Nato funkcija izbere vso vsebino košarice in v tabelo `izdelki` s pomočjo `while` zanke in SQL insert stavka vstavi vsak izdelek posebej. To naredi tako:

```
$stmt2 = $db->prepare("INSERT INTO izdelki (id_narocila, ime_izdelka, kolicina, cena) VALUES (?, ?, ?, ?)");
while ($izdelek = $izdelki->fetch_assoc()) {
    $stmt2->bind_param("isid", $id_narocila, $izdelek['ime'], $izdelek['kolicina'], $izdelek['cena']);
    $stmt2->execute();
}
```

Funkcija nato vrne `$id_narocila`. Potem ko se funkcija `shraniNarocilo` izvede, se v sejo pod atribut `email_narocila` shrani `$email` ter tabela košarica se počisti vseh podatkov z SQL ukazom `TRUNCATE TABLE kosarica`. V sejo se shrani tudi id naročila ter sporočilo uspešnosti oddaje naročila. Če pa status `$paymentIntent` ni bil uspešen, pa uporabnika preusmerimo na stran, kjer lahko potrdi dodatne stvari, če je to potrebno. Nato poskušamo z dvema catch metodama ujeti napake, ki bi se lahko zgodile. Pri prvi iščemo napake kartice, nato pa če jo najdemo izpišemo to napako ter uporabnika preusmerimo na `checkout.php`. Druga catch metoda lovi splošne napake, katerih vsebina je nato dodana v `debug.log`, datoteko v katero se shranjujejo napake, hkrati pa je uporabniku poslano sporočilo, da je bilo plačilo neuspešno ter da je prišlo do napake. Prav tako je preusmerjen na `checkout.php`.

Ob uspešnem plačilu je uporabnik preusmerjen na stran `payment_success.php`. Najprej ta stran preveri, če obstaja `id_narocila`, vezan na sejo (da tuji uporabniki, ki niso kupili ničesar, ne morejo dostopati do te strani). Kupcu se pošlje mail na enak način kot pri potrjevanju registracije. Tukaj sta kot zadeva emaila sporočilo za potrditev nakupa ter kot jedro sporočila potrdilo nakupa z id-jem naročila. Prav tako se na strani te skripte znotraj kartice prikaže sporočilo, ki pravi, da je bilo sporočilo uspešno, zraven je napisana še številka naročila. Uporabniku je s gumbom omogočeno tudi nadaljevanje nakupovanja. Čisto na koncu skripte se iz seje odstranita id naročila in sporočilo uspešnosti. Prva stran za vnos podatkov o plačilu izgleda tako:

Blagajna

Povzetek naročila

Izdelek	Kolicina	Cena	Skupni znesek
AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz	3	640 €	1920 €
Gigabyte B550 GAMING X V2 ATX Motherboard for AMD AM4 CPUs	1	105 €	105 €
Skupno			2025 €

Podatki za plačilo

Kartica:

Številka kartice

MMLL CVC

Plačaj 2025 €

Osební podatki

Ime in priimek

Naslov

Mesto

Poštna številka

Email

Slika 14: Blagajna

5.9. PC Builder

Ob kliku na gumb PC builder, je uporabnik preusmerjen na skripto `pcbuilder.php`. Najprej iz URL-ja dobim id strani in ga shranim v `$id_strani`. Inicializiram tudi spremenljivki `$ddr` in `$socket`, namenjeni pa sta, prva shrambi generacije RAM-a, druga pa shrambi vtičnice procesorja. Nato v tabele `$ddr5`, `$ddr4`, `$ddr3` shranim tiste vtičnice (sockete), ki podpirajo določeno generacijo RAMA. To bom kasneje uporabil pri onemogočenju določenih izbir procesorjev in matičnih plošč pri iskanju kosov iz katerih je sestavljen računalnik, ki ga uporabnik gradi. Nato znotraj `if` stavka preverimo, če je `$socket` prazen. Če ni s pomočjo več `if` stavkov preverimo v katero izmed tabel spada socket shranjen v spremenljivki `$socket` (katero generacijo RAM-a podpira socket procesorja). To naredimo z uporabo funkcije `in_array()`. Če podpira ram generacije DDR5 v `$ram_sql` shranimo SQL select stavek, ki izbiro omeji na tiste RAM-e, ki so pripadniki generacije DDR5. To izgleda tako:

```
if(in_array($socket, $ddr5)){
    $ram_sql= "SELECT * FROM rso_ram WHERE generacija LIKE 'DDR5'";
}
```

Na isti princip napišemo tudi preverjanje, če socket podpira DDR4 ali pa DDR3 RAM. Če pa je `$socket` prazen, pa v `$ram_sql` vpišemo samo navaden select stavek brez omejitev. Enako preverimo tudi če je prazen `$ddr` in če ni, preverimo ali je vanj shranjen DDR5, DDR4 ali pa DDR3. Če je rezultat kateri od zgornjih generacij RAM-ov, se v spremenljivki `$cpu_sql` in `$mobo_sql` shranita SQL select stavka, ki poiščeta samo tiste izdelke, kjer je generacija RAM-a ustrezna. To izgleda tako:

```
if($ddr == 'DDR5'){
    $cpu_sql= "SELECT * FROM rso_cpu WHERE arhitektura IN $ddr5";
    $mobo_sql= "SELECT * FROM rso_mobo WHERE socket IN $ddr5";
}
```

Tudi tukaj v primeru, da je `$ddr` prazen v `$cpu_sql` in `$mobo_sql` shranimo poizvedbe brez omejitev. Dodamo še poizvedbe `$shramba_sql` in `$mobo_sql`, ki nimata nikoli omejitev. Nato v spremenljivke `$cpu`, `$gpu`, `$mobo`, `$ram`, `$shramba`. Nato iz tabele `pcbuild` pridobimo vrednosti zadnjih vnosov s select stavkom ter v spremenljivke `$id_cpu`, `$id_gpu`, `$id_mobo`, `$id_storage` in `$id_ram` shranimo njihove vrednosti. Ustvarimo tudi spremenljivke, ki preverjajo, če je v `pcbuild-u` že dodan vsak od artiklov. To naredijo tako, da preverijo če je kaj zapisano v katero od id spremenljivk zgoraj. Znotraj prikaza spletne strani ustvarimo kartico s katero bomo lahko izbirali katerega od artiklov bomo izbirali ter dodali v

pcbuild. To naredimo s kreacijo elementov seznama. Vsak od elementov je objekt v povezavo do pcbuilder.php skripte, a jim je dodan še vsakemu specifičen id (od 1 do 6, id je tip tabele izdelka in se shrani na začetku skripte v \$id_strani). Vsak od elementov seznama je prikazan samo, če je spremenljivka, ki preverja, če je izdelek že dodan v tabeli, enaka false. To izgleda tako:

```
<?php if (!$ima_cpu){ ?>
<a href="pcbuilder.php?id=1" class="list-group-item list-group-item-action
<?php echo ($id_strani == 1) ? 'active bg-success text-white' : ''; ?>">
Procesorji
</a><?php }?>
```

Taka koda je napisana za vse tipe izdelkov. Nato v novi kartici ustvarimo tabelo in v njeno glavo dodamo attribute, ki jih bomo izpisovali. To so: tip komponente, slika izdelka, ime izdelka in njegova cena. Pod tem pa sledita 2 if stavka, ki preverita kater \$id_strani je trenutno izbran v skripti. Če je izbran id št. 5, je prikazana glavna stran. Znotraj tega if stavka se preverja za vsak tip elementa, če je že dodan v tabelo pcbuild. Če ni, se prikaže gumb, ki uporabnika preusmeri na to skripto (pcbuilder.php), zraven pa se doda še id, ki je enak tipu in pove med katerim tipom izdelkov bomo iskali. Če pa je, se ga s pomočjo select stavka izbere iz tabele, iz katere izdelek prihaja (rso_cpu, rso_gpu...), nato pa se poizvedba izvede in preveri, če je v rezultatu vsaj ena vrstica. Če je izpišemo tip komponente, prikažemo njegovo sliko kot je razvidno spodaj in jo obdamo z povezavo do prikaz.php z id-jem in num-om izdelka, ki ga prikazujemo:

```
<a href='prikaz.php?id=" . $user['id'] . "&num=" . $user['tip'] . "'
class='text-decoration-none text-body'" . '' . "
</a>
```

Po prikazu slike dodamo še ime izdelka, njegovo ceno ter gumb, ki omogoča odstranitev elementa iz pcbuild tabele. To naredi s klicom skripte odstrani_iz_pcbuilderja.php. To bom opisal malo kasneje. Če pa \$id_strani ni enak 5, pa sledi še 5 if stavkov, ki povejo iskalnik katerega tipa izdelkov prikazujemo, t.j., če je id enak 1 prikazujemo procesorjem, če je enak 2 prikazujemo grafične kartice, 3 so matične plošče, 4 je RAM, 6 pa shramba. Znotraj vsake spremenljivke v \$rezultat shranimo poizvedbo, ustvarjeno na vrhu skripte, ki bo izbrala vse ustrezne izdelke za tip izdelka, ki ga prikazujemo. Nato znotraj while zanke za vsako vrstico (izdelek) iz tabele po kateri smo iskali prikažemo sliko in ime (ki ju oba obdamo s povezavo na detajlen prikaz izdelka), ceno izdelka ter gumbom s katerim izdelek dodamo v tabelo pcbuild. Gumb kliče funkcijo add_to_pcbuilder.php. Spodaj pod tabelo, a še vseeno znotraj kartice se izpiše tudi skupna cena, ki jo dobimo s klicem funkcije skupnaCenaPcBuilda, ki sprejme atribut podatkovne baze. To vrednost dobimo tako, da znotraj funkcije ustvarimo in izvedemo SQL select stavek s katerim dobimo vse iz tabele pcbuild. Nato ustvarimo asociativno tabelo imenovano \$tabele. Definiramo tudi spremenljivko \$skupna_cena. Vanjo za vsak id določenega tipa tabele podamo ime tabele v kateri je določen izdelek primarno zapisan. Nato znotraj for each zanke najprej pridobimo id izdelka, katerega ceno bomo prikazovali (iz tabele pcbuild dobimo na mestu atributa id_+tip komponente vrednost id-ja izdelka v njegovi osnovni tabeli.) Nato ustvarimo poizvedbo, s katero dobimo ceno izdelka. Nato to ceno prištejemo skupni ceni (\$skupna_cena). Na koncu funkcije vrnemo \$skupna_cena. V pcbuilder.php nato še dodamo toast sporočilo, ki se pojavi, ko je bila komponenta uspešno dodana (ko uporabnik pritisne gumb dodaj).

Odstrani_iz_pcbuilderja.php deluje tako, da najprej s metodo GET dobi id, in tip komponente, ki jo odstranjujemo. Nato znotraj tabele pcbuild izberemo vse s SQL select stavkom, kjer je id enak \$id ter to poizvedbo izvedemo. Rezultat shranimo v \$user in če ta po izvedbi ne obstaja, vrnemo napako. Definiramo tudi \$odstranitev, ki nam pomaga, da vemo, kater atribut odstraniti. Glede na tip izdelka, ki ga odstranjujemo (CPU, GPU...), lahko s pomočjo switch stavka v \$odstranitev shranimo ime atributa tabele pcbuild, ki ga odstranjujemo. Če tip komponente ni podan, pa se lahko s for each zanko



sprehodimo skozi možne attribute, ter znotraj if stavka izberemo tistega, ki ni enak 0 (ga potrebujemo odstraniti). To naredimo tako:

```
foreach (['id_CPU', 'id_GPU', 'id_mobo', 'id_ram', 'id_storage'] as $polje) {
    if ($user[$polje] > 0) {
        $odstranitev = $polje;
        break;
    }
}
```

V primeru, da je \$odstranitev tudi po tem še vedno prazna, vrnemo napako. Nato pripravimo SQL update stavek s katerim postavimo vrednost atributa, katerega ime vsebuje \$odstranitev na 0 (ga odstranimo). Poizvedbo izpeljemo in v primeru napake, to izpišemo. Na koncu skripte uporabnika še preusmerimo na pcbuilder.php z id-jem 5.

Izdelke v pcbuild dodajamo s funkcijo `add_to_pcbuilder.php`. V njej v najprej v lokalne spremenljivke \$id in \$id_strani shranimo tip komponente, ki jo dodajamo in kater id v svoji primarni tabeli ima ta izdelek. S switch stavkom nato v spremenljivko \$ime shranimo ime atributa, v katerega bomo dodali izdelek. Sledi kreacija SQL select stavka, kjer izberemo id iz pcbuild-a in ga razvrstimo padajoče po id-ju (dobimo najnovejše pretekle vnose). Nato poizvedbo izvedemo in v primeru njene uspešnosti s update stavkom popravimo njeno vrednost na najnovejšem vnosu na pravilno. Če poizvedba ni uspešna, pomeni, da v košarici še ni bilo izdelka tega tipa, ki ga dodajamo. V tem primeru v tabelo vstavimo id izdelka s pomočjo insert stavka. Na koncu skripte uporabnika preusmerimo nazaj na glavno stran pcbuilderja. PC Builder z nekaj dodami izdelki izgleda tako:

Komponente	PC Builder			
Grafične kartice				
RAM				
Shramba				

Komponenta	Ime	Cena	
CPU	 AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz	640.00 €	<button>Odstrani</button>
<button>Izberi GPU</button>			
MATIČNA PLOŠČA	 Gigabyte B550 GAMING X V2 ATX Motherboard for AMD AM4 CPUs	105.00 €	<button>Odstrani</button>
<button>Izberi RAM</button>			
<button>Izberi shrambo</button>			
			Skupna cena: 745.00

Slika 15: PC Builder

5.10. Dodajanje aplikacije na Oraclov strežnik

Po izdelani večini spletne aplikacije sem ugotovil, da bi mi bolj ustrezalo, da bi aplikacijo gostil na strežniku, ki bi ga imel v lasti sam, saj je dostop do aplikacije, na pram lokalnega gostovanja in šolske podatkovne baze, veliko lažji. Prav tako pa lahko kodo v ozadju aplikacije dostopam iz katerega koli računalnika, če poznam samo uporabniško ime in geslo za prijavo v račun. Strežnik gostim preko Oraclove platforme Cloud, na kateri sem naredil instanco strežnika z Ubuntu-jem kot programskim jezikom. Procesor strežnika temelji na ARM arhitekturi (Ampere). Sprva sem se na strežnik povezoval s pomočjo aplikacije PuTTY preko SSH protokola, kasneje pa sem na strežnik naložil nadzorno ploščo aaPanel. Ta mi je omogočila dodajanje različnih aplikacij s klikom enega gumba. Tako sem na strežnik naložil phpMyAdmin, PHP sam, MySQL in Apache. Nato sem na strežnik prenesel podatkovno bazo

ter skripte in ostalo kodo, ki jo uporablja spletna stran. Preko spletne strani GoDaddy sem kupil tudi domeno (shopotron.store) in omogočil, da preko te domene dostopam do spletne aplikacije..

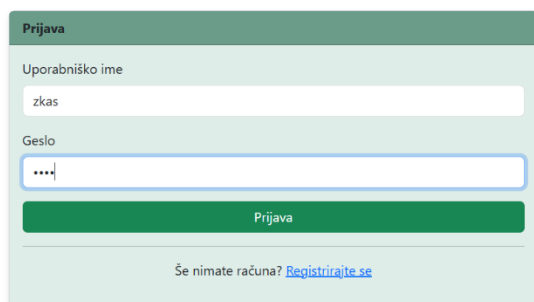
6. TESTIRANJE

6.1. Metoda črne škatle

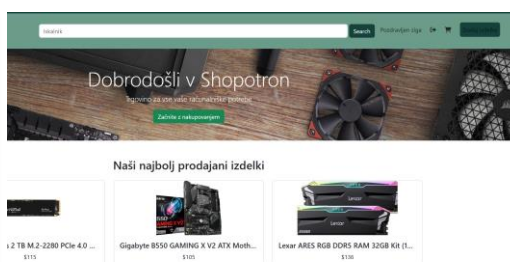
Kot prvi način testiranja bom uporabil tehniko bele škatle, kjer bom preveril delovanje moje spletne aplikacije, pri tem pa ne bom gledal kode, spisane v ozadju, temveč samo vhodno/izhodne podatke.

6.1.1. Prijava

Najprej sem preveril ustrezno delovanje prijave v račun, ki ga je uporabnik že kreiral. Sprva sem se prijavil z ustreznimi podatki, ki bi mi morali dati pravice administratorja:

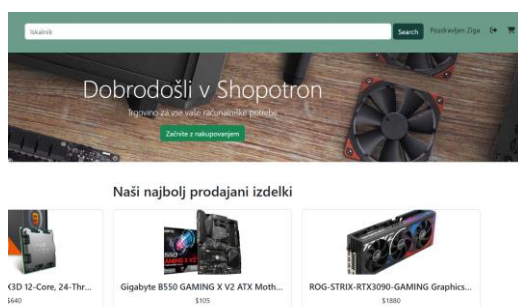


Slika 16: Postopek prijave



Slika 17: Po prijavi

Po potrditvi prijave, sem bi res prijavljen v račun administratorja, kar lahko opazim po pojavu gumba, ki omogoča dodajanje izdelkov na desni strani navigacijske vrstice. Nato sem preveril, kaj se zgodi, če se v sistem vpišem kot navaden uporabnik:



Slika 18: Vpis kot normalen uporabnik

Tudi ta preizkus je bil uspešen, saj sem se v uporabnika lahko prijavil brez težav. Preveril sem tudi, kaj se zgodi, če se uporabnik v račun prijavi z napačnim uporabniškim imenom in geslom. V tem primeru sistem vrne napako v obliki Bootstrap toast sporočila v spodnjem desnem kotu zaslona:

Slika 19: Napačna prijava

S tem sem zagotovil ustrezno delovanje prijave, kjer do napak ne pride.

6.1.2. Registracija

Nato sem preveril pravilno delovanje sistema za registracijo uporabnikov oziroma strani, kjer uporabniki ustvarijo svoj račun. Pri samem vpisu podatkov je uporabniku preprečen napačen vnos email-a, saj ga sistem opozori, če v vpisu maila ni uporabil znaka @. Prav tako je uporabnik opozorjen, če katerega izmed polj ni vpisal (to se naredi tudi pri vseh ostalih vnosnih poljih, tudi na drugih straneh in skriptah po spletni strani):

Slika 20: Napačen email

Slika 21: Prazno polje

Tako se znotraj samega postopka registracije napake ne morejo pojaviti. Preizkusil sem tudi samo registracijo, ki je bila uspešna saj sem na mail prejel sporočilo s kodo za potrditvijo e-maila ter bil preusmerjen na stran za potrditev maila:

Slika 22: email

Slika 23: Potrditev emaila

Po vpisu kode, ki sem jo dobil na e-mail pa sem kot je pravilno preusmerjen na index.php in prejmem toast sporočilo na spodnjem desnem delu strani:

onu

lo tehnologije z vrhunsko uporabniško izkušnjo. Ponaša se z odatkov priznanih blagovnih znamk. Ne glede na to, ali m ali iščete zanesljivo opremo za vsakodnevno uporabo – h, je prednost kakovosti, ugodnim cenam in hitri dostavi. n strokovno svetovanje. Nakupovanje je hitro, enostavno in ost. Shopotron ni le trgovina – je skupnost tehnoloških šem zadovoljnih strank in odkrijte, zakaj je Shopotron ena ij za računalniške navdušence.

Uspešno ste se registrirali!

Slika 24: Uspešna registracija



6.1.3. Dodajanje izdelkov

Pri dodajanju izdelkov bom preveril kako na oddajo vplivajo prazne vrstice forme, ki jo oddajamo, preveril bom tudi, če lahko admin v formo naloži katero koli datoteko. Če uporabnik kakšno od polj ne izpolni, se mu vrne napaka, enaka tisti pri registraciji. Če uporabnik naloži datoteke, ki niso tipa .png, .jpg ali .jpeg, se mu na strani dodaj.php vrne spodje opozorilo:




The screenshot shows a form titled 'Dodaj CPU' with the following fields: 'Število jeder', 'Hitrost CPU-ja (GHz)', 'Ime mikroarhitekture', 'Cena CPU-ja (EUR)', 'Število izdelkov na zalogi', and 'Opis CPU-ja'. Below the form is a green button labeled 'Dodaj CPU'. A red error message is displayed at the bottom: 'Datoteka za slika1 mora biti tipa .jpeg, .jpg ali .png.'

Slika 25: Napačen tip slike

Če uporabnik naloži izdelek in je vnos uspešen se (za primer procesorja) prikaže na strani prikaz.php nov vnos:

Izdelki, ki so na voljo	
	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz Cena: 640.00 € Količina: 18
	AMD Ryzen 9 9900X Processor Cena: 480.00 € Količina: 11

Slika 26: Izdelki pred dodajo

Izdelki, ki so na voljo	
	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz Cena: 640.00 € Količina: 18
	AMD Ryzen 9 9900X Processor Cena: 480.00 € Količina: 11
	AMD Ryzen 7 9800X3D 4.7 GHz 8-Core Processor Cena: 480.00 € Količina: 23

Slika 27: Po dodaji izdelkov

6.1.4. Iskanje

Znotraj iskalne vrstice bom preveril, če se poizvedba izvede ob vsakem vnosu, tudi če je ta nesmiseln ali pa če v iskanje poskušam vpisati kakšen SQL ukaz. Ob vsakem poskusu prelisičenja sistema, mi vrne spodnje sporočilo:



Rezultati Iskanja

Vaša poizvedba ne ustraja nobenemu izdelku


Slika 28: Iskanje brez rezultatov

6.1.5. Košarica in plačevanje

Znotraj košarice bom preveril odstranjevanje izdelkov, ki deluje, saj izdelek po pritisku na gumb odstrani izgine:

Vaša košarica					
Slika	Izdelek	Količina	Cena	Skupaj	
	ASUS TUF Gaming GeForce RTX 4080 Super 16GB GDDR6X OC Edition Gaming Graphics Card	2	1,545.00 €	3,090.00 €	<input type="button" value="Odstrani"/>
	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Processor, up to 5.6GHz	2	640.00 €	1,280.00 €	<input type="button" value="Odstrani"/>
Skupno				4,370.00 €	
					<input type="button" value="Na blagajno"/>

Slika 29: Košarica pred izbrisom

Vaša košarica					
Slika	Izdelek	Količina	Cena	Skupaj	
	ASUS TUF Gaming GeForce RTX 4080 Super 16GB GDDR6X OC Edition Gaming Graphics Card	2	1,545.00 €	3,090.00 €	Odstrani
Skupno				3,090.00 €	
					Na blagajno

Slika 30: Košarica po izbrisu

Prav tako se, če v košarici ni izdelkov izpiše sporočilo, da je košarica prazna:

Vaša košarica

Vaša košarica je prazna.


[Nazaj v trgovino](#)

Slika 31: Prazna košarica

Znotraj plačevanja na strani checkout.php pa bom preveril, če ob vnosu naključne številke kartice transakcija še vedno poteče. Prav tako bom preveril, če vnesemo pravo kartico, a napačen datum poteka. Ob vnosu napačne številke kartice dobimo napako že znotraj samega vpisa:

Podatki za plačilo

Kartica:

 1312 3133 1313 1313

Številka vaše kartice ni veljavna.


Plačaj 3090 €

Slika 32: Ne veljavna številka kartice

Ko pa preverim realno kartico, a vpišem napačen datum, prav tako dobim napako:

Podatki za plačilo

Kartica:

 4242 4242 4242 4242

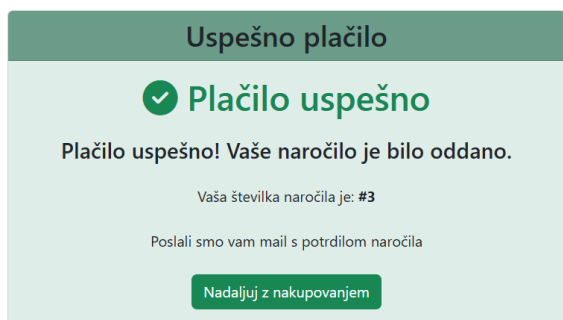
12 / 12 CV

Datum poteka vaše kartice je v preteklosti.

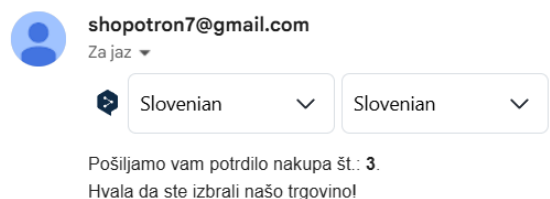
Plačaj 3090 €

Slika 33: Napačen datum veljavnosti

V poljih, kjer mora uporabnik vnesti svoje podatke pa je prav tako kot drugod onemogočeno pošiljanje forme, če niso vneseni vsi podatki in če uporabnikov email ne vsebuje afne. Preveril sem tudi, če deluje oddaja naročila (nakup izdelka). Tudi to je bilo uspešno, saj sem bil preusmerjen na stran s sporočilom o uspehu ter sem na mail prejel potrdilo o nakupu:



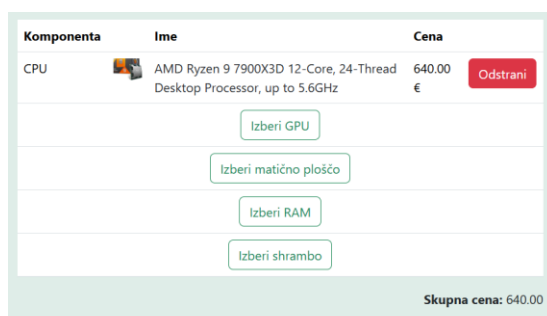
Slika 34: Uspešno naročilo



Slika 35: email s potrdilom

6.1.6. PC Builder, prikazi

Znotraj PC Builderja bom preveril, če dodajanje in odstranjevanje izdelkov deluje kot bi moralo, znotraj prikazov, pa bom preveril, če deluje dodajanje izdelkov v košarico. Najprej sem v prazen PC Builder uspešno dodal element, nato pa sem ga tudi odstranil:

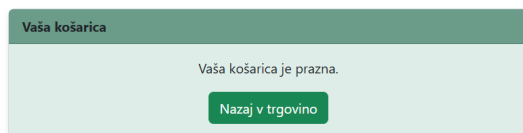


Slika 36: PC Builder z 1 izdelkom



Slika 37: PC Builder brez izdelkov

S tem sem hkrati preveril, če delujejo gumbi Izberi izdelek ter njihova povezava na prikaz ustreznih izdelkov. Nato sem znotraj prikaza računalnika pritisnil gumb Dodaj v košarico, ki je bila prej prazna in vanjo uspešno dodal računalnik:



Slika 38: Prazna košarica, pred dodajanjem



Slika 39: Dodan 1 izdelek v košarico

6.2. Metoda bele škatle

Metoda bele škatle testiranja spletnih strani vključuje testiranje notranje logike, kode in strukture aplikacije. Programer ima pri tej metodi vpogled v izvirno kodo in preverja delovanje posameznih funkcij. Pri tej metodi bom preverjal iste stvari, kot pri metodi črne škatle, saj so to vsa polja oziroma možnosti, ko lahko uporabnik vpliva na delovanje spletne aplikacije. Vredno je omeniti tudi to, da povsod, kjer uporabnik vnaša podatke so ti pred shranitvijo v tabelo spuščeni skozi PHP funkcijo `htmlspecialchars()`, ki onemogoči, da bi uporabnik kakorkoli manipuliral s kodo, ki prejema podatke, saj ta funkcija znake kot so `<` spremeni v `<`, `>` v `>`, `&` v `&`, and `"` v `"`. S tem prepreči t.i. XSS napade (prepreči, da bi uporabnik uporabil škodljive skripte, da bi manipuliral z skripto v katero vnaša podatke). Prav tako pa vse tabele v naprej pripravim ter jih šele nato izvedem kar onemogoči napadalcem napade z SQL injekcijo. Spodaj je primer takega pripravka tabele:

```
$stmt1 = $db->prepare("UPDATE $tabela SET zaloga = zaloga + ? WHERE ime = ?");
$stmt1->bind_param("is", $kolicina, $ime);
$stmt1->execute();
$stmt1->close();
```

Edina šibkost spletne aplikacije, ki jo osebno vidim, se zgodi če uporabnik ureja elemente URL-ja, ko se prikazuje prikaz.php in search.php, saj je od id-ja in num-a v URL-ju odvisno kater element se bo prikazoval na strani.

6.2.1. Prijava

Prva stvar, ki sem jo preveril s metodo bele škatle, je prijava uporabnika. Znotraj kode podatke o prijavi, potem ko jih uporabnik odda, prejme skripta `prijava.php` ta poišče uporabnika, ki ima tako ime, kot ga uporabnik napiše. Če takega uporabnika ni, sistem vrne napako. Nato pa sistem preveri če se geslo, ki ga je napisal uporabnik ujema s tistim, ki je v podatkovni bazi, če se, je uporabnik prijavljen, če pa se ne, pa uporabniku vrnemo napako. Vsa uporabniška imena in gesla so shranjena v tabeli `rso_prijava`. Spodaj je del tabele `rso_prijava`:

				id	ime	username	geslo	email	spol
<input type="checkbox"/>				1	ziga	zkas	pass	ziga.kastelic1@gmail.com	m
<input type="checkbox"/>				2	janez	jnovak	jnovak123	novak@mail.com	m
<input type="checkbox"/>				3	ziga	kastelic	zkastelic	ziga.kastelic@gmail.com	m
<input type="checkbox"/>				4	zan	zankas	zanko	mail@mail.com	m
<input type="checkbox"/>				5	Ziga	ziga	ziga	ziga.kastelic1@gmail.com	m

Slika 39: Vsebina tabele

6.2.2. Registracija

Pri registraciji pa so podatki potem, ko jih uporabnik vpiše, preko seje poslani skripti `confirm.php`. Ustvari se tudi skrita koda, ki je uporabniku poslana na mail s katerim se prijavi. V `confirm.php` mora uporabnik vpisati kodo, ki jo je prejel na mail. Potem se v lokalne spremenljivke shranijo podatki, ki jih je uporabnik vpisal. Nato v tabelo `rso_prijava` vstavimo podatke uporabnika s pomočjo SQL insert stavka. To naredimo s kodo:

```
$koda = $_SESSION['koda']; // Koda, ki je bila ustvarjena v signup.php se
presese v $koda
$vpisanaKoda = $_POST['code']; // Koda, ki jo je uporabnik vnesel v obrazec
if ($vpisanaKoda === $koda) {
    $sql = "INSERT INTO rso_prijava (ime, username, geslo, email, spol)
VALUES ('$ime', '$username', '$geslo', '$email', '$spol')"; }
```

Če vnos v bazo ni uspešen ali pa koda uporabnika ni pravilna, se kupcu prikaze napaka.

6.2.3. Dodajanje izdelkov

Največ možnih napak se lahko pojavi pri dodajanju izdelkov, saj tukaj nastopa največ spremenljivk in tudi sam proces je najbolj odvisen od uporabnika. Pri dodajanju se tip izdelka izbere tako, da se pritisne na gumb, ki uporabnika preusmeri na isto stran, a z drugim id-jem. Ta pove izdelek katerega tipa dodajamo. Znotraj dodaj.php izdelke po pritisku na gumb dodaj pošljemo v funkcijo v skripti funkcije.php, ki je odgovorna za dodajanje določenega tipa izdelkov. Znotraj vsake od teh funkcij se prejeti podatki prilagodijo podatkovnemu tipu, ki bi ga morali imeti z t.i. castanjem pri shranjevanju podatkov, pridobljenih preko \$_POST metode, v lokalne spremenljivke. Primer je spodaj:

```
$hitrost = (double)$_POST['hitrost'];
```

V nadaljevanju kode preveri, če je tip datoteke, ki jih je uporabnik dodal tipa .png, .jpg ali pa tipa .jpeg (da uporabnik ne mora dodati XYZ datotek neznanih podatkovnih tipov). Končnico datoteke tako preveri za vsako sliko posebej:

```
foreach (['slika1', 'slika2', 'slika3'] as $slike) {
    $koncnica = strtolower(pathinfo($_FILES[$slike]['name'],
PATHINFO_EXTENSION));
    if (!in_array($koncnica, $dovoljeneKoncnice)) {
        echo "<div class='alert alert-danger'>Datoteka za $slike
mora biti tipa .jpeg, .jpg ali .png.</div>";
        return; } }
```

Nato za vsako sliko posebej preveri če je bila pravilno naložena s PHP funkcijo `is_uploaded_file`, ki preveri če je bila datoteka naložena. Če ni bila, vrne napako. Podatke nato s pripravljanjem tabel ter kasnejšim izvajanjem doda v tabelo izdelkov. V primeru kakršnih koli drugih napak, to napako izpiše. Ko izdelek uspešno dodamo v podatkovno bazo, se ta v njej prikaže kot nova vrstica, nov vnos:

id	ime	slika1	slika2	slika3	jedra	hitrost	arhitektura	cena	zaloga	opis	tip
1	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Pro...	[BLOB - 126,2 KiB]	[BLOB - 126,6 KiB]	[BLOB - 256,4 KiB]	12	5.6	AM4	640	18	The AMD Ryzen 9 7900X3D is a cutting-edge desktop ...	1
2	AMD Ryzen 9 9900X Processor	[BLOB - 106,6 KiB]	[BLOB - 196,2 KiB]	[BLOB - 105,2 KiB]	12	5.6	AM5	480	11	The AMD Ryzen 9 9900X is a high-performance deskto...	1

Slika 40: Vsebina tabele `rso_cpu` pred dodajanjem

id	ime	slika1	slika2	slika3	jedra	hitrost	arhitektura	cena	zaloga	opis	tip
1	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Pro...	[BLOB - 126,2 KiB]	[BLOB - 126,6 KiB]	[BLOB - 256,4 KiB]	12	5.6	AM4	640	18	The AMD Ryzen 9 7900X3D is a cutting-edge desktop ...	1
2	AMD Ryzen 9 9900X Processor	[BLOB - 106,6 KiB]	[BLOB - 196,2 KiB]	[BLOB - 105,2 KiB]	12	5.6	AM5	480	11	The AMD Ryzen 9 9900X is a high-performance deskto...	1
5	AMD Ryzen 7 9800X3D 4.7 GHz 8-Core Processor	[BLOB - 22,1 KiB]	[BLOB - 20,6 KiB]	[BLOB - 13,7 KiB]	8	4.7	Zen 5	480	23	AMD Ryzen 7 9800X3D je eden izmed najnovejših in n...	1

Slika 41: Vsebina tabele `rso_cpu` po dodajanju

6.2.4. Iskanje

Sledi preverjanje iskanja, kjer iskani niz, potem ko ga uporabnik potrdi, prejme skripta iskanje.php. Tukaj za vsako tabelo s pomočjo SQL select stavka preverimo, če kater izdelek znotraj nje vsebuje iskani niz v imenu. Prav tako s pripravo SQL stavka tudi tukaj onemogočimo SQL injekcijo:

```
$sql = "SELECT id, slika1, ime FROM $tabela WHERE ime LIKE ?";
$stmt = $db->prepare($sql); // Pripravi poizvedbo
$likePoizvedba = '%' . $poizvedba . '%'; // Dodaj wildcard znake za LIKE
$stmt->bind_param("s", $likePoizvedba); // Pripravi parameter
$stmt->execute(); // Izvrši poizvedbo
$resultat = $stmt->get_result();
```

Celoten izdelek, ki ustreza poizvedbi nato izpišemo kot pri search.php.

6.2.5. Košarica in plačevanje

Znotraj nakupovalnega vozička se kaj veliko napak ne more pojaviti. Edino kar lahko preverimo je dodajanje in odstranjevanje izdelka v tabelo kosarica. Dodajanje izvaja skripta dodaj_v_kosarico.php. Ta preveri, če je izdelek že dodan v košarico in če je samo posodobi njegovo količino, če pa ga v košarici še ni, ga doda. Odstranjevanje izvaja skripta odstrani_iz_kosarice.php. Ta izdelek odstrani iz košarice in doda količino izdelka, ki je v košarici, nazaj v njegovo osnovno tabelo. Vse SQL poizvedbe tudi tukaj so pripravljene vnaprej, da se izognemo nepotrebnim šibkostim sistema. V primeru napake pri poizvedbi sistem izpiše napako. Spodaj lahko vidimo spremembe v podatkovni bazi ob dodajanju izdelka v košarico ter odstranjevanju drugega izdelka:

id	ime	slika	cena	kolicina	tabela
1	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Pro...	[BLOB - 126,2 KiB]	640	3	rso_cpu
2	ROG-STRIX-RTX3090-GAMING Graphics Card	[BLOB - 200,3 KiB]	1880	2	rso_gpu

Slika 42: Košarica pred dodajanjem

id	ime	slika	cena	kolicina	tabela
1	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Pro...	[BLOB - 126,2 KiB]	640	3	rso_cpu
2	ROG-STRIX-RTX3090-GAMING Graphics Card	[BLOB - 200,3 KiB]	1880	2	rso_gpu
3	Gigabyte B550 GAMING X V2 ATX Motherboard for AMD ...	[BLOB - 320,1 KiB]	105	1	rso_mobo

Slika 43: Košarica po dodajanju

id	ime	slika	cena	kolicina	tabela
1	AMD Ryzen 9 7900X3D 12-Core, 24-Thread Desktop Pro...	[BLOB - 126,2 KiB]	640	3	rso_cpu
3	Gigabyte B550 GAMING X V2 ATX Motherboard for AMD ...	[BLOB - 320,1 KiB]	105	1	rso_mobo

Slika 44: Košarica po izbrisu

6.2.6. PC Builder, prikazi

V PC Builderju in prikazih je največja šibkost (po mojem mnenju) to, da lahko uporabniki zmanipulirajo URL ter se s tem učinkovito preusmerijo na drugo stran. Razen tega večjih šibkosti ni. Edino kar lahko

testiram je dodajanje in odstranjevanje izdelkov v PC Builderju, ki deluje podobno kot tisto v košarici. Razlika je samo, da pri pcbuild tabeli namesto tega, da dodajamo celotne izdelke, samo spreminjamo attribute tabele, ki služijo kot kazalec na izdelek, ki je v PC Builderju trenutno. To je razvidno iz spodnjega prikaza, kjer vsak atribut prikazuje posamezen tip izdelka:

1	2	0	0	0
---	---	---	---	---

Slika 45: pcbuild pred dodajanjem

1	2	1	0	0
---	---	---	---	---

Slika 46: pcbuild po dodajanju

1	0	1	0	0
---	---	---	---	---

Slika 47: pcbuild po izbrisu

7. ZAKLJUČEK

V tej seminarski nalogi sem izdelal delujočo spletno aplikacijo, ki je namenjena prodaji računalnikov in komponent. Pri tem sem najprej ustvaril funkciji, ki izpišeta navigacijsko vrstico in nogo, nato sem ustvaril indeks spletne strani (prvo stran). Sledila je kreacija potrebne podatkovne baze z več tabelami. Adminu strani sem omogočil tudi dodajanje novih izdelkov v podatkovno bazo. Nato sem izdelal stran za prijavo v uporabnikov račun in stran, ki kreacijo omenjenega računa sploh omogoča. Izdelal sem tudi stran namenjeno prikazu določene vrste komponente (CPE, GPE, matična plošča, RAM in shramba) in stran namenjeno prikazu računalnikov. Uporabnik lahko na tej strani klikne na izdelek, ki si ga želi bolj natančno pogledati, in ta se mu bo odprl na novi strani. Po tabelah z izdelki sem tudi omogočil iskanje, tako da lahko uporabnik poišče prav določen izdelek znotraj vseh kategorij. Uporabnik lahko izdelke tudi dodaja v košarico ter te, ki jih je dodal v košarico, tudi kupi. Če ima željo, lahko uporabnik sestavi tudi simulacijo svojega računalnika, stran pa mu prav tako nudi samo možne izdelke, ki pašejo skupaj. Samo spletno stran in podatkovno bazo sem dodal tudi na strežnik, ki ga gostim preko Oracle Cloud-a.

Pri izdelavi te seminarske naloge sem se naučil veliko novega tako o samih programskih jezikih, uporabljenih znotraj izdelave spletne strani kot tudi o oblikovanju spletnih strani. Prav tako se mi zdi, da je zelo uporabno to, da sem se pri izdelavi te naloge naučil uporabe Bootstrapa, saj mi bo to v prihodnosti prišlo zelo prav.

Na začetku naloge nisem predvideval, da bom dodal spletno stran in podatkovno bazo na strežnik, a sem to naredil, saj sem imel pred rokom oddaje še čas, prav tako pa se mi je to zdelo zelo praktično tako zame kot programerja, kot tudi uporabnika spletne strani. Čisto sem dokončal tudi plačevanje s pomočjo Stripe API-ja, na začetku izdelovanja, pa sem mislil, da bom to moral narediti manualno.

Če bi spletno aplikacijo razvijal še naprej, bi uporabnikom dodal možnost izbire profilne slike, ki bi jo lahko izbrali ob prijavi. Prav tako bi morda dodal popust, ki ga imajo prijavljeni uporabniki. Mogoče bi izdelal tudi stran na kateri lahko administrator vidi vse tabele z izdelki ter njihovo vsebino.

Bibliografija

aaPanel. (12. april 2025). Pridobljeno iz aaPanel: <https://www.aapanel.com/>

API Reference. (11. april 2025). Pridobljeno iz Stripe: <https://docs.stripe.com/api>

CSS. (10. april 2025). Pridobljeno iz Wikipedia: <https://en.wikipedia.org/wiki/CSS>

Developer Mozilla. (10. april 2025). Pridobljeno iz JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Get started with Bootstrap. (10. april 2025). Pridobljeno iz Bootstrap: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

HTML. (10. april 2025). Pridobljeno iz Wikipedia: <https://en.wikipedia.org/wiki/HTML>

Oracle Cloud Infrastructure Documentation. (12. april 2025). Pridobljeno iz Oracle: <https://docs.oracle.com/en-us/iaas/Content/Compute/Concepts/computeoverview.htm>

PuTTY. (12. april 2025). Pridobljeno iz PuTTY Documentation Page: <https://www.chiark.greenend.org.uk/~sgtatham/putty/docs.html>

So what is Cloudflare? (12. april 2025). Pridobljeno iz Cloudflare: <https://www.cloudflare.com/learning/what-is-cloudflare/>

SQL Tutorial. (10. april 2025). Pridobljeno iz W3Schools: <https://www.w3schools.com/sql/>

What is PHP and what can it do? (10. april 2025). Pridobljeno iz PHP: <https://www.php.net/manual/en/introduction.php>

Why MySQL? (10. april 2025). Pridobljeno iz MYSQL: <https://www.mysql.com/why-mysql/>

PRILOGE:

Povezava in QR koda do GitHub repozitorija s kodo: https://github.com/ZKastelic/rac_matura

