Академија струковних
студија Шумадија
Одсек Крагујевац

Студијски програм: Информатика
Предмет: Пројектовање апликација база података

Уметничка дела

- Пројекат –

Предметни наставник:
Др Хрвоје Пушкарић

Студент:
Сара Стевановић, 012/2023

Крагујевац 2025.

Поставка задатка

Projektovanje aplikacija baza podataka – Uputstvo za izradu projektnog zadatka

Projektni zadatak: Tema projektnog zadatka je projektovanje aplikacije baza podataka koristeći standardne internet tehnologije **(HTML, CSS, Bootstrap, PHP, MySQL)** uz primenu objektno-orijentisanog programiranja (**OOP**). Na projektnom zadatku student može da osvoji **maksimalno 15 poena**. Aplikacija mora da obuhvati sledeće

Na projektnom zadatku student može da osvoji **maksimalno 15 poena**.

Aplikacija mora da obuhvati sledeće:

1. **Korisnički interfejs** koji podrazumeva **obaveznu upotrebu Bootstrap okvira i upotrebu Bootstrap mreže**. Interfejs mora da uključi Bootstrap kartice, tabele i forme.
2. **Aplikacija** mora da obuhvati sistem za prijavu i registraciju korisnika što podrazumeva upotrebu login forme, registracione forme kao i upotrebu sistema sesija. Implementacija mora koristiti PHP klase za upravljanje korisnicima i sesijama.
3. **Aplikacija** mora da sadrži bazu podataka koja je definisana odgovarajućim tabelama i atributima. Student bira bazu podataka iz liste ponuđenih baza podataka unutar Excel fajla.
4. Na osnovu definisane baze podataka, aplikacija mora da sadrži sistem čitanja iz baze podataka, upis u bazu podataka i osvežavanje podataka. Ove CRUD (create, read, update, delete) operacije moraju biti implementirane korišćenjem PHP klase koja sadrži specifične metode za CRUD funkcionalnosti (npr. create(), read(), update(), delete()). Obavezno je definisati najmanje jedan interfejs koji specificira ove CRUD metode i implementirati ga u odgovarajućoj klasi. Bootstrap komponente moraju biti korišćene za prikaz i unos podataka.
5. Aplikacija mora koristiti objektno-orijentisane principe, uključujući definisanje klasa za upravljanje bazom podataka, korisnicima i drugim entitetima, kao i metode za obradu podataka. Klase treba da koriste nasleđivanje uz implementaciju najmanje jednog interfejsa koji definiše CRUD operacije.

1. Увод

Циљ овог пројекта је развој једноставне веб-апликације за вођење евиденције о уметницима, уметничким делима, галеријама и продајама. Систем омогућава основне CRUD операције (креирање, читање, измену и брисање) над четири повезана ентитета. Апликација је реализована у PHP-у, са MySQL базом и Bootstrap-ом за кориснички интерфејс.

Поред функционалних захтева, посебна пажња посвећена је безбедности и поузданости. Форме користе припремљене упите како би се спречиле SQL инјекције. Пријава корисника заснива се на сесијама. На нивоу интерфејса, апликација нуди прегледне табеле, модалне форме за брзо уређивање и мини „dashboard" са основном статистиком и скорашњим продајама, чиме се убрзава рад и побољшава преглед над системом.

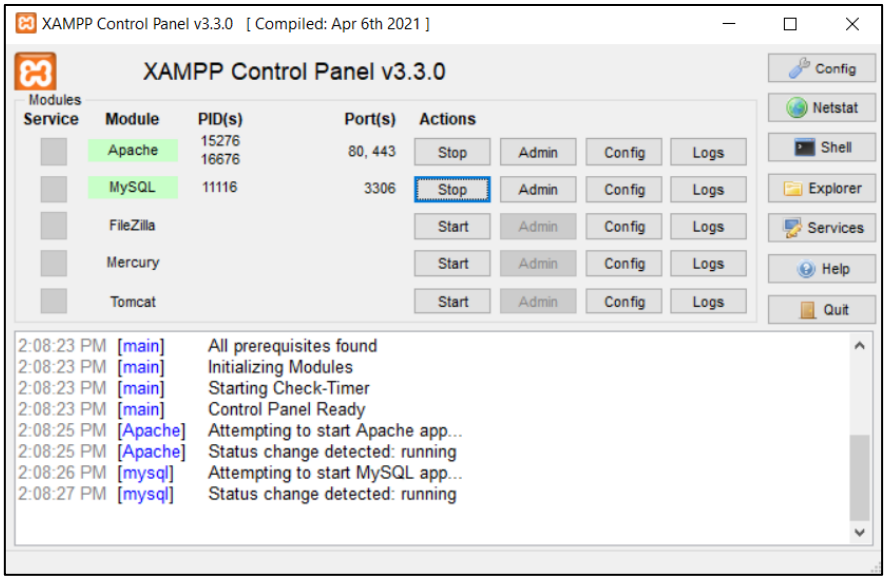
2. Теоретска основа

2.1 Xampp

XAMPP је интегрисани развојни пакет намењен локалном покретању веб-апликација. Обједињује четири кључне компоненте класичног веб стека: Apache као веб сервер који обрађује HTTP захтеве, PHP као интерпретер серверског кода, MySQL/MariaDB као релациону базу података за трајно чување информација и phpMyAdmin као графички алат за управљање базом. Циљ XAMPP-а је да на једном рачунару обезбеди цело окружење за развој и тестирање, тако да програмер може да ради офлајн, без посебног подешавања сваке компоненте појединачно.

Архитектонски посматрано, XAMPP симулира типичан клијент–сервер модел: прегледач шаље захтеве Apache-у, Apache их прослеђује PHP интерпретеру који извршава пословну логику, по потреби приступа бази података преко MySQL/MariaDB сервиса, а затим враћа припремљени HTML или JSON као одговор.

У оквиру једног контролног панела корисник покреће и зауставља Apache и MySQL/MariaDB, а у конфигурационим датотекама подешава опште опције сервера, модуле, ограничења PHP-а и понашање базе.



Слика1: Контролни панел

Адреса http://localhost/phpmyadmin/ значи да у прегледачу отвараш веб-апликацију phpMyAdmin која ради на твом рачунару. Да би ова адреса радила, у XAMPP-у морају бити покренути Apache и MySQL/MariaDB.

Слика2: Лого

2.2 SQL основе

SQL је декларативни језик за рад са релационим базама података. Обухвата DDL за дефинисање шеме (CREATE/ALTER/DROP), DML за манипулацију подацима (SELECT/INSERT/UPDATE/DELETE), DCL за дозволе (GRANT/REVOKE) и TCL за управљање трансакцијама (BEGIN/COMMIT/ROLLBACK). Типови података попут INT, DECIMAL, VARCHAR и DATE омогућавају тачно моделирање домена, док примарни и страни кључеви, UNIQUE, NOT NULL и CHECK обезбеђују интегритет и доследност.

Упити се ослањају на JOIN за повезивање табела (INNER/LEFT), WHERE за филтрирање, ORDER BY за сортирање и агрегате са GROUP BY/HAVING за извештаје. Индекси убрзавају претрагу и спајање, али имају цену при упису, па их треба пажљиво бирати и анализирати план извршавања (нпр. EXPLAIN). Трансакције обезбеђују ACID својства и атомичност сложених измена, што је кључно за пословна правила.

Сигурност се постиже припремљеним упитима и параметрима ради спречавања SQL инјекција, ограниченим привилегијама корисника базе и ваљаном валидацијом уноса.

2.3 PHP основе

PHP је серверски скрипт језик намењен обради HTTP захтева и генерисању HTML одговора. Рад са улазом се врши преко суперглобала $_GET, $_POST и $_FILES, излаз се шаље функцијама попут echo, а преусмеравање странице помоћу header(). Сесије се покрећу са session_start(), колачићи постављају setcookie(), а излаз се штити од XSS-а функцијом htmlspecialchars(). Рад са базом се обично обавља преко mysqli или PDO; уобичајен образац је припремљени упит са prepare(), везивање параметара (bind_param / bindValue) и извршавање (execute()), што спречава SQL инјекције. За лозинке се користе password_hash() и password_verify(). Укључивање заједничког кода врши се преко require/include (или строжих require_once/include_once). Овим основама PHP омогућава да се у једном циклусу „захтев → логика → одговор“обради унос корисника, упише/прочита податак из базе и врати динамички HTML.

Слика3: Лого

Поред тога, PHP нуди практичне механизме за организацију кода и рад са подацима: креирање класа и метода (ООП) уз типске декларације и try { } catch (Throwable $e) { } за обраду грешака, шаблонизацију преко include/require и баферисање излаза путем ob_start() и ob_get_clean(). За сигурнији унос користан је filter_input()/filter_var(), а за рад са JSON-ом json_encode() и json_decode(); датотеке се читају и пишу преко file_get_contents() и file_put_contents(). Подешавања у развоју могу да се контролишу позивима као што су ini_set('display_errors', '1') и error_reporting(E_ALL), уз error_log() за записивање грешака. У пракси се препоручује да се слојеви одвоје на модел, контролер и приказ, да се комуникација са базом своди на припремљене упите ($pdo->prepare()/$stmt->execute() или mysqli->prepare()), а да се рад са сесијама и колачићима

(session_start(), setcookie()) комбинује са екранирањем излаза htmlspecialchars() ради заштите од XSS-а.

2.4 Сесија

Сесија у PHP-у је механизам за памћење стања корисника између више HTTP захтева. Када позовеш session_start(), PHP креира јединствени ИД сесије и отвори асоцијативни низ $_SESSION у који можеш да упишеш податке попут идентитета корисника или ставки корпе. Те вредности се чувају на серверу и доступне су на свакој наредној страници након session_start(). После успешне пријаве добро је урадити session_regenerate_id(true) да би се спречила фиксација сесије, а за одјаву се користи брисање података и session_destroy(). Сесију треба покретати пре било каквог излаза (пре echo/HTML), а ради безбедности колачић сесије треба да буде HttpOnly, по могућству Secure и са SameSite политиком. У сесији не треба чувати велике објекте ни осетљиве податке у чистом облику; чувај само оно што ти је неопходно.

2.5 Безбедност

Безбедност веб-апликације у PHP-у заснива се на неколико кључних принципа: исправно руковање улазом/излазом, контролисане сесије, безбедно чување тајни и принцип „најмањих привилегија“. Основна идеја је да се сви подаци који долазе од корисника сматрају непоузданим док се не валидирају и не употребе на сигуран начин, а да се све што се шаље назад у HTML буде екранирано како не би дошло до убризгавања злонамерног садржаја.

Најчешћи напади које треба спречити су SQL инјекција, XSS и CSRF. SQL инјекција се елиминише припремљеним упитима (параметризовани упити) уместо конкатенације стрингова. XSS се спречава екранирањем излаза у HTML-у (htmlspecialchars) и забраном небезбедног уноса у атрибуте/скриптове. CSRF захтева токене у формама и проверу токена на серверу, како би само захтеви настали на вашој страници били прихваћени. Поред тога, лозинке никада не треба чувати у чистом облику већ као хешеве са password_hash() и проверавати password_verify(). Сесије треба заштитити регенерацијом ИД-а после пријаве, прикладним подешавањем колачића (HttpOnly, Secure, SameSite) и проверама да само ауторизовани корисници приступају заштићеним страницама. На нивоу базе и сервера примењује се „најмање привилегија“: апликациони корисник базе има само потребне

дозволе, табеле имају ограничења (NOT NULL, CHECK, FOREIGN KEY) и индексе, а конфигурације за продукцију су строже него у развоју.

2.6 HTML

HTML је markup језик који описује структуру веб-странице. Састоји се од елемената записаних у угластим заградама (нпр. <p>, <a>, <img>), који могу имати атрибуте (href, alt, class…). Прегледач чита HTML и од њега гради DOM стабло, на које се затим надовезују CSS (за презентацију) и JavaScript (за понашање), па важи раздвајање улога: структура у HTML-у, стил у CSS-у, логика у JS-у.

Документ почиње декларацијом <!doctype html>, корени елемент је <html lang="sr">, а садржи два кључна дела: <head> (метаподаци, наслов, повезивање стилова и скриптова) и <body> (видљиви садржај). HTML5 је увео семантичке елементе као што су <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>, који дају значење распорету садржаја и побољшавају доступност и SEO. Форме (<form>) са савременим типовима поља (email, number, date) имају основну валидацију атрибутима као што су required, min, max. Приступачност се унапређује употребом исправних заглавља (<h1>–<h6>), алтернативног текста за слике (alt), повезивањем <label> са пољима форме.

Слика4: Лого

2.7 CSS

CSS је језик за стилизовање HTML-а, одређује боје, фонтове, размаке, распоред и анимације. Правила се примењују преко селектора. Кључни појмови су box model (margin, border, padding, content), позиционирање (position), и савремени распореди преко Flexbox-а и Grid-а. За респонзивни дизајн користе се media queries, а стилови се могу уметнути inline, унутар <style>, или као спољни фајл.

Слика5: Лого

2.8 Bootstrap

Bootstrap је фронт-енд оквир који убрзава израду респонзивних веб-интерфејса. Заснован је на модуларном систему компоненти (навигација, дугмад, картице, модали, обрасци) и на мрежном систему (Grid/Flexbox) који омогућава распоред елемената у колонама за различите величине екрана. Поред компоненти, нуди велики сет „utility“ класа за брзо подешавање размака, типографије, поравнања и видљивости без писања сопственог CSS-а.

Савремене верзије се ослањају на CSS променљиве и Flexbox/Grid, а JavaScript компоненти користе мале скриптове за интеракције попут падајућих менија и модалних прозора (у новијим верзијама без зависности од jQuery). Теме и визуелни идентитет могу се прилагодити преко променљивих и SASS променљивих, па је лако задржати конзистентан стил кроз цео сајт, уз добру подршку за приступачност и респонзивно понашање.

3. Опис практичног дела

3.1 База

База се зове umetnickadela, садржи табеле umetnik, korisnik, galerija, prodaja и umetnicka_dela.

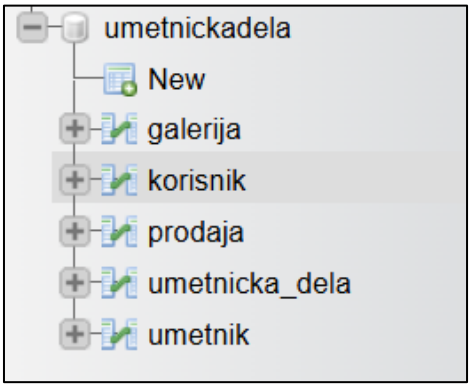
За табелу umetnik, примарни кључ је id_umetnika, а од поља садржи још ime, prezime, datum_рођења и biografiju.

За табелу umetnicka_delaпримарни кључ је id_umetnickogDela, а поља су naziv_dela, godina, tip, cena, id_umetnik(спољни кључ), galerija_id (спољни кључ).

За табелу galerija примарни кључ је id_galerije, а поља су naziv_galerije и adresa.

За табелу prodaja примарни кључ је id_prodaje, а поља су umetnicko_delo_id (спољни кључ), galerija_id (спољни кључ), datum, kupac, cena.

За табелу korisnik примарни кључ је id_korisnika, а поља су korisnicko_ime и lozinka.



Слика6: База и табеле у бази

	id_umetnika	ime	prezime	datum_rodjenja	biografija
Edit Copy Delete	1	Pablo	Picasso	1881-10-25	Пабло Руиз Пикасо (шп. Pablo Ruiz Picasso,[2] Мала...
Edit Copy Delete	2	Vincent	van Gogh	1853-03-30	Винсент Вилем ван Гог био је сликар холандског пор...
Edit Copy Delete	3	Frida	Kahlo	1907-07-06	Фрида Кало је била мексичка сликарка. Рођена је ка...
Edit Copy Delete	4	Claude	Monet	1840-11-14	Моне се родио 14. новембра 1840. Био је други син ...
Edit Copy Delete	5	Salvador	Dali	1904-05-11	Салвадор Фелипе Хасинто Дали, 1. маркиз од Пубола ...

Слика7: Табела уметник

	id_umetnickogDela	naziv_dela	godina	tip	cena	id_umetnik	galerija_id ▲ 1
Edit Copy Delete	1	Guernica	1937	slika	200000.00	1	1
Edit Copy Delete	5	Zvezdana noć	1889	slika	1500000.00	2	1
Edit Copy Delete	4	San	1937	slika	1400000.00	5	2
Edit Copy Delete	9	Vodeni ljiljani	1916	slika	1500000.00	4	2
Edit Copy Delete	3	Upornost pamćenja	1931	slika	180000.00	5	3
Edit Copy Delete	8	Dvostruki portret	1939	slika	14000000.00	3	3
Edit Copy Delete	2	Plava perioda - Stara gitara	1903	slika	1200000.00	1	4
Edit Copy Delete	7	Autoportret sa trnovom krunom	1940	slika	1500000.00	3	4
Edit Copy Delete	6	Suncokreti	1888	slika	1800000.00	2	5
Edit Copy Delete	10	Impresija, izlazak sunca	1872	slika	1200000.00	4	5

Слика8: Табела уметничка дела

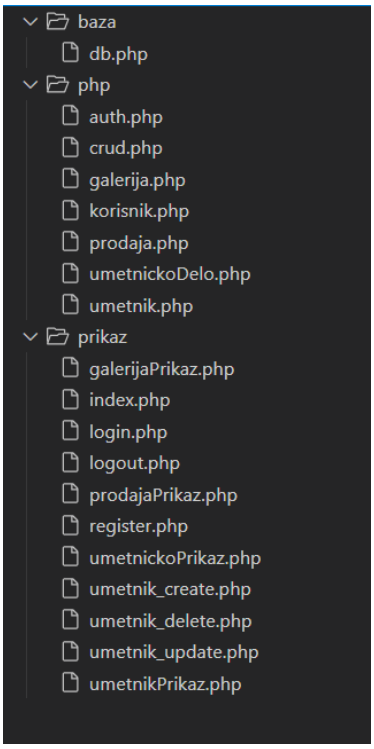
	id_galerije	naziv_galerije	adresa
Edit Copy Delete	1	Louvre	Rue de Rivoli 75001, Paris, Francuska
Edit Copy Delete	2	MOMA	11 W 53rd St, New York, USA
Edit Copy Delete	3	Tate Modern	Bankside, London, UK
Edit Copy Delete	4	Galerija Umetnosti Beograd	Kralja Milana 36, Beograd, Srbija
Edit Copy Delete	5	Prado	Calle de Ruiz de Alarcón 23, Madrid, Španija

Слика9: Табела галерија

	id_prodaje	umetnicko_delo_id	datum	kupac	cena	galerija_id
Edit Copy Delete	1	4	2017-09-12	John Smith	1200000.00	2

Слика10: Табела продаја

3.2 Хијерархија пројекта



Слика11: Хијерархија пројекта

Ова структура је подељена на три нивоа. У директоријуму baza налази се db.php, једино место где се успоставља MySQL конекција и добија заједнички објекат $conn који остале скрипте увозе преко require_once.

Директоријум php садржи пословну логику и моделе који директно раде са базом. Фајл auth.php управља пријавом, одјавом и сесијом. Модели korisnik.php, umetnik.php, umetnickoDelo.php, galerija.php и prodaja.php инкапсулирају CRUD операције за своје табеле. По потреби, crud.php служи као заједнички помоћни слој. Ове скрипте не исписују HTML већ само извршавају упите и враћају резултате.

Директоријум prikaz садржи стране које отвара прегледач и које комбинују контролер и приказ. index.php је почетна табла са навигацијом и провером да ли је корисник пријављен. login.php, register.php и logout.php спроводе процес пријаве/регистрације преко класе Auth. umetnikPrikaz.php, umetnickoPrikaz.php, galerijaPrikaz.php и prodajaPrikaz.php читају податке из модела и приказују их у Bootstrap табелама и модалним формама. Датотеке umetnik_create.php, umetnik_update.php и umetnik_delete.php су „акциони“ ендпоинти који примају захтев, позивају метод модела и затим раде редирекцију назад (PRG образац).

Ток једног захтева изгледа овако: прегледач тражи страницу из prikaz, она учитава baza/db.php и одговарајући модел из php, модел преко заједничке конекције приступа бази и враћа податке, а страница их рендерује у HTML-у. Добра пракса је да директоријум php остане недоступан директно са веба, док prikaz чини јавни слој апликације. Ова подела чува јасноћу кода: конекција на једном месту, логика у моделима, а прикази у засебним страницама.

3.3 Имплементација CRUD-а

У овом пројекту прво је дефинисан CRUD интерфеј који декларише четири основне методе: create(), read(), update() и delete(). Све доменске класе, Umetnik, UmetnickoDelo, Galerija и Prodaja имплементирају тај интерфејс.

```php
<?php

interface Crud{

    public function create($data);
    public function read($id);
    public function update($id,$data);
    public function delete($id);
}

?>
```

Слика12: CRUD интерфејс

Ово је модел класа за табелу umetnik која имплементира CRUD интерфејс и представља јединствен слој преко кога апликација приступа записима у тој табели. Конструктор прима заједничку MySQLi конекцију, а назив табеле је централизован у пољу $table. Све операције користе припремљене упите ради заштите од SQL инјекција: create() уноси новог аутора са пољима ime, prezime и biografija; read($id) враћа један ред по примарном кључу id_umetnika; update($id, $data) ажурира постојећи запис; delete($id) га брише. Везивање параметара преко bind_param одговара типовима у бази (нпр. s за текст, i за цели број), па се подаци сигурно прослеђују серверу. Уколико се у шеми користи и поље за датум рођења, исто се једноставно додаје у INSERT и UPDATE упите, чиме класа остаје јасна, компактна и у потпуности изолована од презентационог слоја.

```php
<?php
require_once __DIR__ . '/crud.php';

class Umetnik implements Crud {
  private $conn;
  private $table="umetnik";
  public function __construct($conn){ $this->conn=$conn; }

  public function create($data){
    $stmt=$this->conn->prepare("INSERT INTO $this->table (ime,prezime,biografija) VALUES (?,?,?)");
    $stmt->bind_param("sss",$data['ime'],$data['prezime'],$data['biografija']);
    return $stmt->execute();
  }
  public function read($id){
    $stmt=$this->conn->prepare("SELECT * FROM $this->table WHERE id_umetnika=?");
    $stmt->bind_param("i",$id); $stmt->execute();
    return $stmt->get_result()->fetch_assoc();
  }
  public function update($id,$data){
    $stmt=$this->conn->prepare("UPDATE $this->table SET ime=?,prezime=?,biografija=? WHERE id_umetnika=?");
    $stmt->bind_param("sssi",$data['ime'],$data['prezime'],$data['biografija'],$id);
    return $stmt->execute();
  }
  public function delete($id){
    $stmt=$this->conn->prepare("DELETE FROM $this->table WHERE id_umetnika=?");
    $stmt->bind_param("i",$id);
    return $stmt->execute();
  }
}
?>
```

Слика13: CRUD методе имплементиране у класи Umetnik

3.4 Имплементација PHP-а

Имплементација PHP-а у пројекту заснована је на јасној подели слојева. У baza/db.php се успоставља једна заједничка MySQLi конекција која се свуда увози. У директоријуму php/ налазе се модел-класе (Umetnik, UmetnickoDelo, Galerija, Prodaja, као и Auth), које имплементирају CRUD интерфејс и инкапсулирају сав рад са базом преко припремљених упита. Презентациони слој је у prikaz/ и садржи странице које примају GET/POST захтеве, позивају моделе, а затим враћају HTML (Bootstrap табеле и модалне форме). После сваког уписа/измене примењује се PRG образац – обрада података, па header("Location: ...") – како би се избегло дупло слање форме.

Аутентификација је реализована преко PHP сесија; након успешне пријаве чува се минималан скуп података о кориснику, а приступ заштићеним странама проверава се преко Auth. Безбедност се постиже параметризованим упитима (спречавање SQL инјекција) и екранирањем излаза htmlspecialchars() у приказима (ублажавање XSS-а), уз базна ограничења и стране кључеве у шеми. Валидирају се обавезна поља и типови, грешке се враћају као кратке поруке кориснику, а структура кода (конекција → модели → прикази) одржава јасан и одвојен ток: захтев се обради у моделу, резултат се проследи приказу, а кориснику се враћа чист, респонзиван HTML.

3.5 Сесија и пријава

Сесија у апликацији служи да запамти стање пријављеног корисника између више захтева. Након session_start() прегледач добија колачић са ИД-јем сесије (PHPSESSID), а на серверу се у $_SESSION чувају минимални подаци, нпр. id и username. После успешне пријаве ради се session_regenerate_id(true) ради заштите од фиксације сесије. Све заштићене странице на почетку проверавају да ли је корисник пријављен; ако није, преусмеравају на форму за логин. Одјава брише податке из $_SESSION и позива session_destroy(). Колачић сесије треба да буде HttpOnly, а по могућству и Secure и са SameSite политиком.

Пријава је реализована кроз класу Auth која користи модел Korisnik. Форма шаље корисничко име и лозинку на login.php; сервер припремљеним упитом добија запис корисника и проверава лозинку (у развоју је дозвољено просто поређење, у продукцији се користе password_hash() и password_verify()). Ако су креденцијали исправни, у сесију се уписује корисников идентификатор, ради се редирекција на почетну страницу и даље приступање системским страницама је условљено сесијом. Уколико провера падне, кориснику се враћа порука о грешци без откривања детаља.

```php
<?php


require_once __DIR__ . '/korisnik.php';


class Auth {
    private Korisnik $korisnik;

    public function __construct(mysqli $conn){
        $this->korisnik = new Korisnik($conn);
        if (session_status() === PHP_SESSION_NONE) session_start();
    }

    public function login(string $u, string $p): bool {
        $user = $this->korisnik->getKorisnik($u);

        if ($user && $p === $user['lozinka']) {
            session_regenerate_id(true);
            $_SESSION['korisnik'] = [
                'id'       => $user['id_korisnika'],
                'username' => $user['korisnicko_ime']
            ];
            return true;
```

```php
        }
        return false;
    }

    public function register(string $u, string $p): bool {
        if ($this->korisnik->getKorisnik($u)) return false;

        return $this->korisnik->createKorisnik($u, $p);
    }

    public function logout(): void {
        if (session_status() !== PHP_SESSION_ACTIVE) session_start();
        $_SESSION = [];
        session_destroy();
    }

    public function isLoggedIn(): bool {
        return isset($_SESSION['korisnik']);
    }

    public function getUser(): ?array {
        return $_SESSION['korisnik'] ?? null;
    }
}
?>
```

Слика14: Auth класа

```php
<?php
class Korisnik {
    private mysqli $conn;
    private string $table = "korisnik";

    public function __construct(mysqli $conn){
        $this->conn = $conn;
    }

    public function getKorisnik(string $username): ?array {
        $sql = "SELECT id_korisnika, korisnicko_ime, lozinka
                FROM {$this->table}
                WHERE korisnicko_ime = ? LIMIT 1";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $rez = $stmt->get_result()->fetch_assoc();
        return $rez ?: null;
    }


    public function createKorisnik(string $username, string $password): bool {
        $sql = "INSERT INTO {$this->table} (korisnicko_ime, lozinka) VALUES (?, ?)";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("ss", $username, $password);
        return $stmt->execute();
    }
}
?>
```

Слика15: класа Korisnik

```php
<?php
require_once __DIR__ . '/../baza/db.php';
require_once __DIR__ . '/../php/korisnik.php';
require_once __DIR__ . '/../php/auth.php';

$auth = new Auth($conn);

if ($auth->isLoggedIn()) {
  header('Location: index.php');
  exit;
}

$msg = '';
$ok  = isset($_GET['ok']) ? 'Uspešna registracija! Sada se prijavi.' : '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $u = isset($_POST['username']) ? trim($_POST['username']) : '';
  $p = $_POST['password'] ?? '';

  if ($u === '' || $p === '') {
    $msg = 'Unesi korisničko ime i lozinku.';
  } else {
    if ($auth->login($u, $p)) {
      header('Location: index.php');
      exit;
    } else {
      $msg = 'Pogrešno korisničko ime ili lozinka.';
    }
  }
}
?>
```

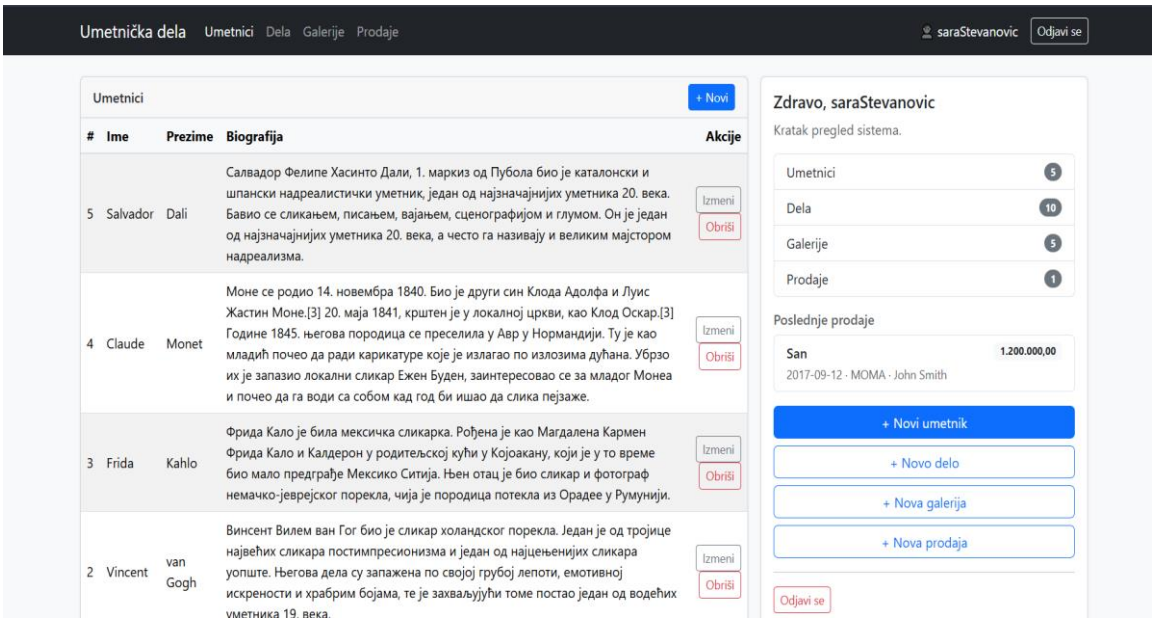
Слика16: класа Login

3.6 Интерфејс

Интерфејс пројекта је једноставан, респонзиван и заснован на Bootstrap-у, тако да се коректно приказује и на телефону и на десктопу. На врху сваке странице налази се тамна навигациона трака са линковима ка четири целине: „Уметници“, „Дела“, „Галерије“ и „Продаје“, као и статусом пријављеног корисника и дугметом за одјаву. Почетна страница након пријаве приказује поздрав и пречице ка главним приказима, па корисник одмах може да оде на рад са подацима.

Сваки приказ има исти шаблон: картица са кратком формом за додавање новог записа (Create) и испод ње табела са постојећим подацима (Read). Табеле су прегледне, са јасним насловима колона и последњом колоном за акције. За измену и брисање коришћена су мала Bootstrap дугмад; измена отвара модални прозор са попуњеним пољима (Update), а брисање има једноставну JS потврду (Delete). Форме користе HTML валидацију и приказују поруке о успеху или грешци изнад садржаја, уз PRG образац да се избегне поновно слање захтева на refresh.

Поља су типизирана (нпр. бројеви и датуми имају одговарајуће input типове), текст се екранира у приказу ради безбедности, а цене се форматирају ради читљивости. Захваљујући униформном распореду (картица за унос + табела + модали), корисник у свим деловима система има исто искуство рада: брз унос, брза измена и јасан преглед података.



Слика17: приказ главне странице

4. Закључак

Резултат пројекта је компактна, функционална и проширива апликација која покрива кључне потребе мале галерије или приватне колекције: централизовано чување података, јасну повезаност између уметника, дела и продаја, као и једноставан интерфејс за свакодневни рад. Применом припремљених упита и екранирања излаза постигнут је добар ниво безбедности за типичне веб нападе, док сесије омогућавају контролисан приступ заштићеним страницама. Релациони модел и уредни идентификатори олакшавају одржавање, анализу и извештавање.