**ПРИЛОЖЕНИЕ А**

**Листинг кода**

<?php

// Конфигурация безопасности и аутентификации

class SecurityConfig {

private $db;

private $userService;

public function \_\_construct($db, $userService) {

$this->db = $db;

$this->userService = $userService;

}

/\*\*

\* Проверка аутентификации пользователя

\* @param string $email Email пользователя

\* @param string $password Пароль пользователя

\* @return array|bool Данные пользователя или false при неудаче

\*/

public function authenticate($email, $password) {

$stmt = $this->db->prepare("SELECT \* FROM users WHERE email = ?");

$stmt->execute([$email]);

$user = $stmt->fetch();

if ($user && password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['user\_role'] = $user['role'];

return $user;

}

return false;

}

/\*\*

\* Регистрация нового пользователя

\* @param array $userData Данные пользователя

\* @return bool Результат регистрации

\*/

public function register($userData) {

$hashedPassword = password\_hash($userData['password'], PASSWORD\_BCRYPT);

$stmt = $this->db->prepare("

INSERT INTO users (email, name, password, role, registration\_date)

VALUES (?, ?, ?, 'user', NOW())

");

return $stmt->execute([

$userData['email'],

$userData['name'],

$hashedPassword

]);

}

/\*\*

\* Проверка прав доступа

\* @param string $requiredRole Требуемая роль

\* @return bool Результат проверки

\*/

public function checkAccess($requiredRole) {

if (!isset($\_SESSION['user\_role'])) {

return false;

}

return $\_SESSION['user\_role'] === $requiredRole;

}

/\*\*

\* Получение разрешенных URL для неавторизованных пользователей

\* @return array Массив разрешенных URL

\*/

public function getPublicUrls() {

return [

'/',

'/login',

'/register',

'/catalog',

'/book',

'/search',

'/assets/css',

'/assets/js',

'/assets/images'

];

}

}

/\*\*

\* Класс для работы с корзиной покупок

\*/

class ShoppingCart {

private $db;

public function \_\_construct($db) {

$this->db = $db;

}

/\*\*

\* Добавление книги в корзину

\* @param int $userId ID пользователя

\* @param int $bookId ID книги

\* @param int $quantity Количество

\* @return bool Результат операции

\*/

public function addToCart($userId, $bookId, $quantity = 1) {

// Проверка наличия книги

$stmt = $this->db->prepare("

SELECT quantity FROM books WHERE id = ?

");

$stmt->execute([$bookId]);

$book = $stmt->fetch();

if ($book['quantity'] < $quantity) {

return false;

}

// Добавление в корзину

$stmt = $this->db->prepare("

INSERT INTO cart\_items (user\_id, book\_id, quantity)

VALUES (?, ?, ?)

ON DUPLICATE KEY UPDATE quantity = quantity + ?

");

return $stmt->execute([$userId, $bookId, $quantity, $quantity]);

}

/\*\*

\* Оформление заказа

\* @param int $userId ID пользователя

\* @param array $orderData Данные заказа

\* @return int|bool ID заказа или false при неудаче

\*/

public function checkout($userId, $orderData) {

try {

$this->db->beginTransaction();

// Создание заказа

$stmt = $this->db->prepare("

INSERT INTO orders (user\_id, total\_amount, status, order\_date)

VALUES (?, ?, 'pending', NOW())

");

$stmt->execute([$userId, $orderData['total\_amount']]);

$orderId = $this->db->lastInsertId();

// Перенос товаров из корзины в заказ

$stmt = $this->db->prepare("

INSERT INTO order\_items (order\_id, book\_id, quantity, price)

SELECT ?, book\_id, quantity,

(SELECT price FROM books WHERE books.id = cart\_items.book\_id)

FROM cart\_items WHERE user\_id = ?

");

$stmt->execute([$orderId, $userId]);

// Очистка корзины

$stmt = $this->db->prepare("

DELETE FROM cart\_items WHERE user\_id = ?

");

$stmt->execute([$userId]);

$this->db->commit();

return $orderId;

} catch (Exception $e) {

$this->db->rollBack();

return false;

}

}

}

?>

Этот код демонстрирует реализацию основных классов безопасности и работы с корзиной покупок в онлайн библиотеке. Код включает механизмы аутентификации, регистрации пользователей, управления доступом и обработки покупок. Использованы современные практики безопасности, такие как хеширование паролей и защита от SQL-инъекций через подготовленные запросы.