

Министерство образования Республики Беларусь

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

ОТЧЕТ

По индивидуальной практической работе №1

**«СОЗДАНИЕ ВЕБ-СЕРВИСА НА *RНР* ДЛЯ ОБРАБОТКИ ДАННЫХ  
ФОРМЫ»**

Выполнил:

ст. гр. 220604

Д. В. Рахманько

Проверил:

Н. В. Хаджинова

Минск 2025

## СОДЕРЖАНИЕ

Введение .....	3
1 Постановка задачи.....	4
2 Теоретическая часть.....	5
3 Ход работы.....	6
Заключение .....	14
Приложение а (обязательное) Программный код приложения.....	15

## ВВЕДЕНИЕ

Веб-разработка в современном цифровом мире требует обязательного владения фундаментальными технологиями - *HTML* и *CSS*. Язык гипертекстовой разметки *HTML* (*HyperText Markup Language*) формирует структурную основу веб-страниц, определяя такие ключевые элементы как заголовки, текстовые блоки, медиа-контент, табличные данные и гиперссылки. Каскадные таблицы стилей *CSS* (*Cascading Style Sheets*) отвечают за визуальное представление этих элементов, предоставляя разработчикам инструменты для управления цветовой палитрой, типографикой, пространственным расположением компонентов и другими аспектами дизайна.

*PHP* (*PHP: Hypertext Preprocessor*) представляет собой серверный язык программирования, специально разработанный для веб-разработки. В отличие от клиентских технологий *HTML* и *CSS*, *PHP* выполняется на сервере, что позволяет создавать динамические веб-страницы с возможностью обработки пользовательских данных, взаимодействия с базами данных, управления сессиями и реализации сложной бизнес-логики. *PHP* обеспечивает связь между пользовательским интерфейсом и серверными ресурсами, включая системы управления базами данных *MySQL*, *PostgreSQL* и другие.

Настоящая практическая работа направлена на освоение базовых принципов работы с *HTML*, *CSS* и *PHP* в рамках создания полнофункционального веб-приложения для управления заказами печати документов.

Основная цель лабораторной работы - формирование практических навыков создания статических веб-страниц (*HTML/CSS*) и их интеграции с серверной логикой (*PHP*) для обработки пользовательских данных и взаимодействия с базой данных. Выполнение практических заданий поможет понять взаимосвязь между клиентской частью приложения (*frontend*) и серверной логикой (*backend*), включая процессы валидации данных, их сохранения в базе данных *MySQL* и последующего извлечения для отображения пользователю. Особое внимание уделяется вопросам безопасности веб-приложений, включая защиту от *SQL*-инъекций через использование подготовленных запросов, валидацию пользовательского ввода и безопасное отображение данных.

# 1 ПОСТАНОВКА ЗАДАЧИ

В рамках данной практической работы требуется разработать полнофункциональный веб-сервис для управления заказами печати документов, который продемонстрирует практическое владение современным стеком веб-технологий: *HTML* для создания семантической структуры документов, *CSS* для визуального оформления и обеспечения адаптивности интерфейса, *PHP* для реализации серверной логики обработки данных, а также *MySQL* для организации надежного хранения информации о заказах.

Практическая работа начинается с проектирования архитектуры веб-приложения, включая создание схемы базы данных для хранения информации о заказах печати, определение структуры файлов проекта и планирование пользовательских сценариев взаимодействия с системой. Следующим этапом становится настройка локальной среды разработки с использованием серверного программного обеспечения (*XAMPP*, *OpenServer* или аналогичного), создание базы данных *MySQL* с соответствующими таблицами и проверка корректности подключения к системе управления базами данных.

После подготовки инфраструктуры необходимо реализовать клиентскую часть приложения, включающую главную страницу с описанием сервиса, *HTML*-форму для оформления заказов с валидацией полей на стороне клиента, а также страницу просмотра всех заказов с табличным представлением данных и элементами статистики. Каждая страница должна быть оформлена с использованием современного *CSS*-фреймворка *Bootstrap* для обеспечения адаптивности и профессионального внешнего вида, включая использование иконочных шрифтов, цветовых схем и анимационных эффектов.

Серверная часть приложения предполагает создание *PHP*-скриптов для обработки *POST*-запросов от форм, реализацию комплексной валидации пользовательских данных (проверка типов, длины строк, диапазонов значений и корректности дат), безопасное взаимодействие с базой данных через подготовленные *SQL*-запросы для предотвращения инъекций, а также формирование информативных сообщений об успешном выполнении операций или возникших ошибках.

Важной составляющей работы является обеспечение безопасности веб-приложения, для чего требуется реализовать защиту от основных типов атак (*XSS*, *SQL*-инъекции, *CSRF*), настроить корректную обработку ошибок с логированием критических событий, а также организовать валидацию данных как на клиентской, так и на серверной стороне с соответствующими уведомлениями пользователя.

Завершающий этап включает комплексное тестирование всех компонентов системы: проверку корректности работы форм с различными наборами входных данных, тестирование адаптивности интерфейса на мобильных устройствах и планшетах, валидацию *HTML*-кода и *CSS*-стилей, а также проверку производительности при работе с большими объемами данных.

## 2 КОНТРОЛЬНЫЙ ВОПРОС

Как загрузить файл на сервер через форму и обработать его на *PHP*?

Для загрузки файла на сервер через форму и его обработки на *PHP* необходимо создать *HTML*-форму с атрибутом *enctype="multipart/form-data"* и элементом *input type="file"*, затем на сервере использовать глобальную переменную *\$\_FILES* для получения информации о загруженном файле, проверить код ошибки через *\$\_FILES['имя\_поля']['error']*, валидировать тип и размер файла, создать уникальное имя файла для предотвращения конфликтов, переместить временный файл в целевую директорию с помощью функции *move\_uploaded\_file()*, установить соответствующие права доступа на директорию загрузок и обработать возможные ошибки загрузки, включая проверку максимального размера файла в *php.ini* (*upload\_max\_filesize* и *post\_max\_size*), а также реализовать дополнительные меры безопасности такие как проверка *MIME*-типа, расширения файла, сканирование на вирусы для загружаемых исполняемых файлов и ограничение доступа к загруженным файлам через веб-сервер.

### 3 ХОД РАБОТЫ

Первым этапом работы стало проектирование структуры базы данных для хранения заказов на печать документов. Была создана база данных с именем *user\_form\_db\_rakhmanko*, содержащая таблицу *print\_orders* со следующими полями:

- *id* (*INT*, *AUTO\_INCREMENT*, *PRIMARY KEY*) - уникальный идентификатор заказа
- *document\_name* (*VARCHAR(255)*, *NOT NULL*) - название документа для печати
- *print\_format* (*VARCHAR(50)*, *NOT NULL*) - формат печати (*A4*, *A3*, *A5*, *Letter*, *Legal*)
- *copies* (*INT*, *NOT NULL*) - количество копий
- *pickup\_date* (*DATE*, *NOT NULL*) - дата получения заказа
- *created\_at* (*TIMESTAMP*, *DEFAULT CURRENT\_TIMESTAMP*) - дата и время создания заказа

Для создания базы данных был подготовлен *SQL*-запрос:

```
database.sql
1  -- Создание базы данных для веб-сервиса приема заказов на печать
2  CREATE DATABASE IF NOT EXISTS user_form_db_rakhmanko CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
3
4  USE user_form_db_rakhmanko;
5
6  -- Создание таблицы для хранения заказов на печать
7  CREATE TABLE IF NOT EXISTS print_orders (
8      id INT AUTO_INCREMENT PRIMARY KEY,
9      document_name VARCHAR(255) NOT NULL,
10     print_format VARCHAR(50) NOT NULL,
11     copies INT NOT NULL,
12     pickup_date DATE NOT NULL,
13     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
14     INDEX idx_pickup_date (pickup_date),
15     INDEX idx_created_at (created_at)
16 );
17
18 -- Проверка созданной таблицы
19 DESCRIBE print_orders;
```

Рисунок 1 – *SQL*-запрос

Структура таблицы была спроектирована с учетом требований к хранению данных о заказах печати, включая индексы для оптимизации поиска по датам получения и создания заказов. Структура таблицы была спроектирована с учетом требований к хранению данных о заказах печати, включая индексы для оптимизации поиска по датам получения и создания заказов.

Для обеспечения безопасности и удобства сопровождения был создан отдельный файл конфигурации *config.php*, содержащий параметры подключения к базе данных и функцию для установления соединения:

```

<?php
/**
 * Конфигурационный файл для подключения к базе данных
 */
define(constant_name: 'DB_HOST', value: 'localhost');
define(constant_name: 'DB_USERNAME', value: 'root');
define(constant_name: 'DB_PASSWORD', value: '1111');
define(constant_name: 'DB_NAME', value: 'user_form_db_rakhmanko');
define(constant_name: 'DB_CHARSET', value: 'utf8mb4');

2 references
function getDatabaseConnection(): bool|mysqli {
    $connection = new mysqli(hostname: DB_HOST, username: DB_USERNAME, password: DB_PASSWORD, database: DB_NAME);
    if ($connection->connect_error) {
        error_log(message: "Ошибка подключения к БД: " . $connection->connect_error);
        return false;
    }
    $connection->set_charset(charset: DB_CHARSET);

    return $connection;
}
?>

```

Рисунок 2 – Содержимое *config.php*

Файл содержит константы для хоста, имени пользователя, пароля и имени базы данных, а также функцию *getDatabaseConnection()*, которая возвращает объект подключения *mysqli* с установленной кодировкой *UTF-8*.

Следующим этапом стала разработка клиентской части приложения - *HTML*-формы для ввода данных о заказе. Форма была создана с использованием современного *CSS*-фреймворка *Bootstrap* для обеспечения адаптивности и профессионального внешнего вида.

Основные элементы формы включают:

- Поле для ввода названия документа с валидацией длины;
- Выпадающий список для выбора формата печати (*A4*, *A3*, *A5*, и т.д.);
- Числовое поле для указания количества копий (от 1 до 1000);
- Поле выбора даты получения с ограничением на прошедшие даты;
- Кнопки для очистки формы и отправки заказа;

```

<form action="process.php" method="POST" id="printOrderForm">
  <!-- Название документа -->
  <div class="mb-3">
    <label for="document_name" class="form-label">
      <i class="fas fa-file-alt me-2"></i>Название документа *
    </label>
    <input
      type="text"
      class="form-control"
      id="document_name"
      name="document_name"
      placeholder="Введите название документа"
      required
      maxlength="255"
    >
    <div class="form-text">Максимум 255 символов</div>
  </div>

```

Рисунок 3 - Часть разметки формы

Форма настроена для отправки данных методом *POST* на обработчик *process.php*. Добавлена клиентская валидация с помощью *JavaScript* для предварительной проверки корректности введенных данных.

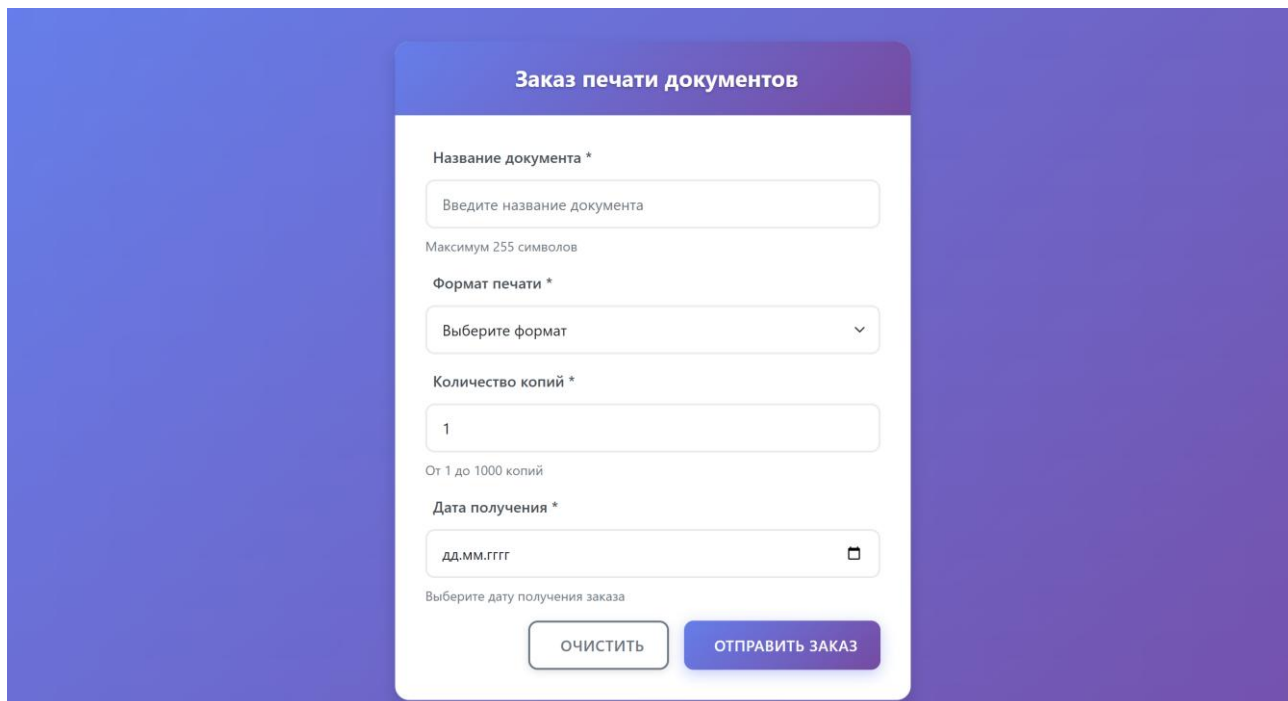


Рисунок 4 – Внешний вид страницы с формой

Серверная часть приложения представлена файлом *process.php*, который выполняет функции описанные ниже.

Проверка метода запроса. Скрипт проверяет, что данные поступили методом *POST*. При попытке прямого обращения через *GET*-запрос пользователь автоматически перенаправляется на форму заказа.

Валидация входных данных. Реализована комплексная валидация всех полей формы:

- Проверка названия документа на пустоту, длину и допустимые символы;
- Валидация формата печати по списку разрешенных значений;
- Проверка количества копий в диапазоне от 1 до 1000;
- Валидация даты получения (не в прошлом, корректный формат);

Для обеспечения безопасности используются подготовленные запросы (*Prepared Statements*), которые предотвращают *SQL*-инъекции.

Создана функция *displayResult()* для генерации *HTML*-страницы с результатом обработки заказа, включающей информативные сообщения об успехе или ошибках.



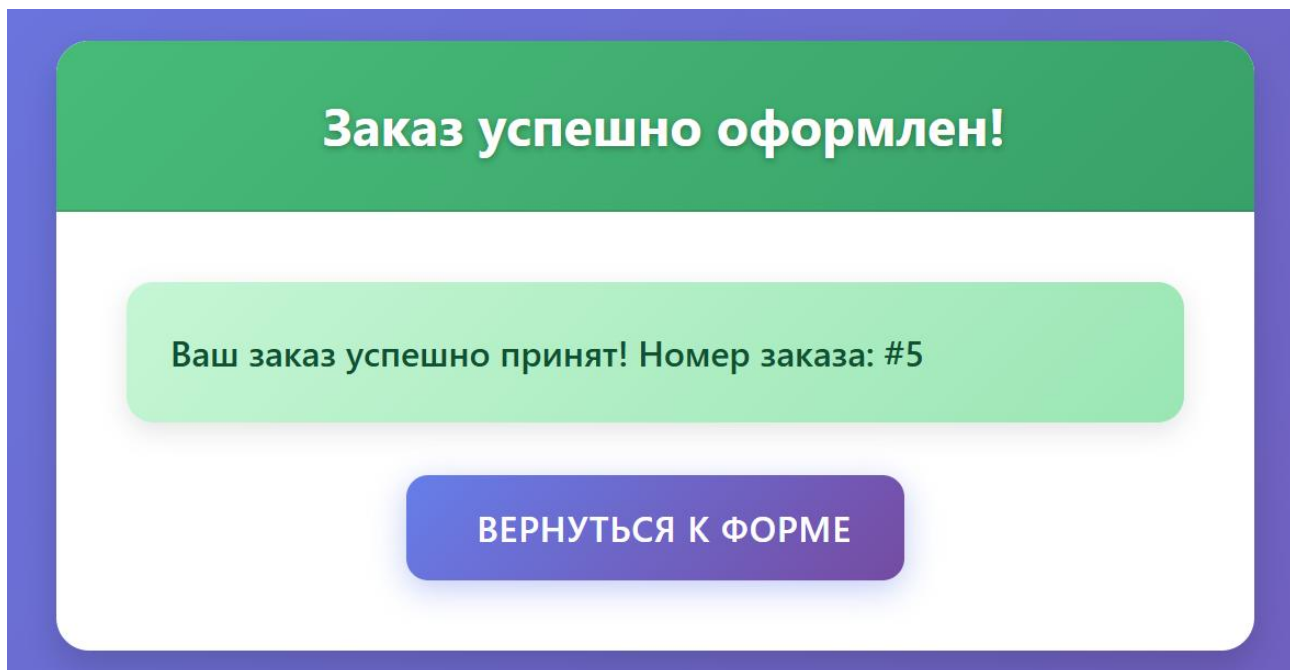


Рисунок 5 – Внешний вид страницы с результатом

Для удобства администрирования системы была разработана дополнительная страница *view\_orders.php*, позволяющая просматривать все оформленные заказы в табличном виде с элементами статистики.

Функциональные возможности страницы:

- Отображение всех заказов в хронологическом порядке;
- Статистические данные;
- Адаптивная таблица с информацией о каждом заказе;
- Навигационные элементы для перехода к форме заказа;

Все заказы на печать

НОВЫЙ ЗАКАЗ

ГЛАВНАЯ

Всего заказов

5

Всего копий

478

Популярный формат

A4

Последний заказ

19.09.2025

Список всех заказов

#	НАЗВАНИЕ ДОКУМЕНТА	ФОРМАТ	КОПИИ	ДАТА ПОЛУЧЕНИЯ	ДАТА СОЗДАНИЯ
5	test	A4	12	20.09.2025	19.09.2025 14:40:20
4	567890987654	Letter	186	22.02.2030	18.09.2025 19:31:15
3	2345678	A4	177	31.03.2066	15.09.2025 16:53:32
2	321123	Legal	100	11.11.2025	12.09.2025 16:46:56
1	lab1	A4	1	11.02.2222	12.09.2025 16:00:35

Рисунок 6 – Страница для просмотра заказов

Была разработана информативная главная страница, содержащая описание сервиса печати документов, навигационные элементы для доступа к основным функциям, информацию о поддерживаемых форматах печати и ограничениях.

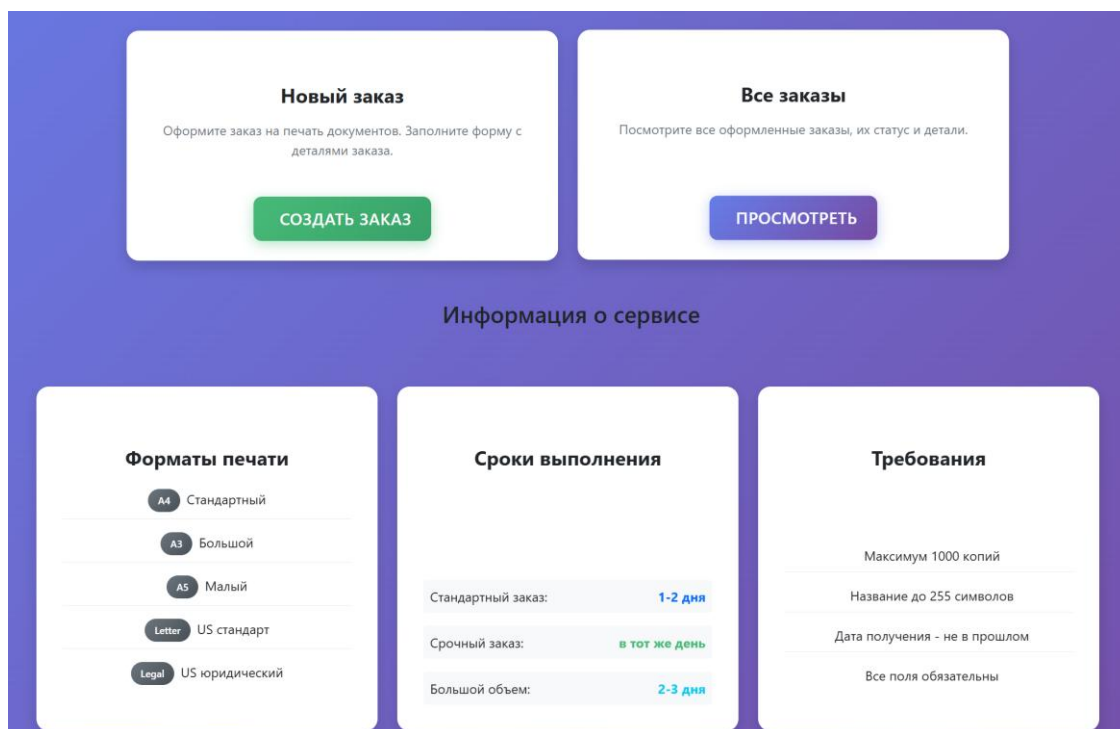


Рисунок 7 – Внешний вид главной страницы

Рисунок 8 – Некоторые примеры валидации

## ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы разработан веб-сайт туристического агентства с использованием *PHP* и *MySQL*. Сайт содержит следующие веб-страницы: главная страница с информацией о компании, страница каталога туров с возможностью просмотра доступных предложений, страница бронирования с формой для оформления заказа и страница просмотра заказов для администратора. Итогом выполнения данной работы стало получение практических навыков создания динамических веб-приложений с использованием серверного языка программирования *PHP*, работы с реляционной базой данных *MySQL* через *SQL*-запросы, обработки *HTML*-форм и валидации пользовательских данных, а также реализации системы аутентификации и управления пользователями. Были изучены основы веб-безопасности, включая защиту от *SQL*-инъекций, валидацию и санитизацию входных данных, а также правильную обработку файловых загрузок с проверкой типов и размеров файлов. Для разработки и тестирования использовался локальный веб-сервер *XAMPP*, что позволило создать полнофункциональное приложение без необходимости внешнего хостинга. В результате был создан интерактивный сайт турагентства с возможностью просмотра туров, оформления бронирования, загрузки документов и административного управления заказами с распределением прав доступа между обычными пользователями и администраторами системы.

Ссылка на проект *github*: <https://github.com/le5n1k/ipr1>

## ПРИЛОЖЕНИЕ А

### Листинг кода

```
<?php
require_once 'config.php';
function htmlEscape($string) {
    return htmlspecialchars($string, ENT_QUOTES, 'UTF-8');
}
function displayResult($success, $message, $errors = []) {
    ?>
    <!DOCTYPE html>
    <html lang="ru">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Результат обработки заказа</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <div class="container mt-5">
            <div class="row justify-content-center">
                <div class="col-md-8 col-lg-6">
                    <div class="card shadow">
                        <div class="card-header <?php echo $success ? 'bg-success' : 'bg-danger'; ?> text-white">
                            <h2 class="card-title mb-0 text-center">
                                <i class="fas <?php echo $success ? 'fa-check-circle' : 'fa-exclamation-triangle'; ?> me-
2"></i>
                                <?php echo $success ? "Заказ успешно оформлен!" : "Ошибка обработки заказа"; ?>
                            </h2>
                        </div>
                        <div class="card-body">
                            <div class="alert <?php echo $success ? 'alert-success' : 'alert-danger'; ?>" role="alert">
                                <strong><?php echo htmlEscape($message); ?></strong>
                            </div>

                            <?php if (!empty($errors)): ?>
                                <div class="alert alert-warning" role="alert">
                                    <h5><i class="fas fa-exclamation-triangle me-2"></i>Обнаружены следующие
ошибки:</h5>
                                    <ul class="mb-0">
                                        <?php foreach ($errors as $error): ?>
                                            <li><?php echo htmlEscape($error); ?></li>
                                        <?php endforeach; ?>
                                    </ul>
                                </div>
                            <?php endif; ?>

                            <div class="text-center mt-4">
                                <a href="form.html" class="btn btn-primary">
                                    <i class="fas fa-arrow-left me-2"></i>Вернуться к форме
```

```

        </a>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>

<script src="https://kit.fontawesome.com/your-font-awesome-kit.js"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
<?php
}

if($_SERVER['REQUEST_METHOD'] !== 'POST') {
    //Если это GET-запрос, перенаправляем на форму
    if($_SERVER['REQUEST_METHOD'] === 'GET') {
        header('Location: form.html');
        exit;
    }
    displayResult(false, 'Неверный метод запроса. Используйте POST.');
```

*exit;*

```

}

$errors = [];

$document_name = isset($_POST['document_name']) ? trim($_POST['document_name']) : '';
$print_format = isset($_POST['print_format']) ? trim($_POST['print_format']) : '';
$copies = isset($_POST['copies']) ? (int)$_POST['copies'] : 0;
$pickup_date = isset($_POST['pickup_date']) ? trim($_POST['pickup_date']) : '';

if(empty($document_name)) {
    $errors[] = 'Название документа не может быть пустым';
} elseif(strlen($document_name) > 255) {
    $errors[] = 'Название документа не может превышать 255 символов';
} elseif(!preg_match('/^[a-zA-Za-яА-ЯёЁ0-9\s\-\_\.\,()]+\$/u', $document_name)) {
    $errors[] = 'Название документа содержит недопустимые символы';
}

$allowed_formats = ['A4', 'A3', 'A5', 'Letter', 'Legal'];
if(empty($print_format)) {
    $errors[] = 'Формат печати должен быть выбран';
} elseif(!in_array($print_format, $allowed_formats)) {
    $errors[] = 'Недопустимый формат печати';
}

if($copies <= 0) {
    $errors[] = 'Количество копий должно быть больше 0';
} elseif($copies > 1000) {
    $errors[] = 'Количество копий не может превышать 1000';
}

```

```

if (empty($pickup_date)) {
    $errors[] = 'Дата получения должна быть указана';
} else {
    $pickup_timestamp = strtotime($pickup_date);
    $today_timestamp = strtotime(date('Y-m-d'));

    if ($pickup_timestamp === false) {
        $errors[] = 'Неверный формат даты получения';
    } elseif ($pickup_timestamp < $today_timestamp) {
        $errors[] = 'Дата получения не может быть в прошлом';
    }
}

if (!empty($errors)) {
    displayResult(false, 'Проверьте правильность заполнения формы.', $errors);
    exit;
}

$connection = getDatabaseConnection();
if (!$connection) {
    displayResult(false, 'Ошибка подключения к базе данных. Попробуйте позже.');
```

exit;

```

}

try {
    $stmt = $connection->prepare("INSERT INTO print_orders (document_name, print_format, copies,
pickup_date) VALUES (?, ?, ?, ?)");

    if (!$stmt) {
        throw new Exception('Ошибка подготовки запроса: ' . $connection->error);
    }

    $stmt->bind_param("ssis", $document_name, $print_format, $copies, $pickup_date);

    if ($stmt->execute()) {
        $order_id = $connection->insert_id;
        displayResult(true, "Ваш заказ успешно принят! Номер заказа: #{$order_id}");
    } else {
        throw new Exception('Ошибка выполнения запроса: ' . $stmt->error);
    }

    $stmt->close();

} catch (Exception $e) {
    error_log("Ошибка при сохранении заказа: " . $e->getMessage());
    displayResult(false, 'Произошла ошибка при сохранении заказа. Попробуйте позже.');
```

finally {

```

    if ($connection) {
        $connection->close();
    }
}
?>

```

