РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

В.С. Якуничев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

МДК.04.01 «Бэкенд-разработка (серверная часть)»

для специальности
09.02.09 Веб-разработка

Ростов-на-Дону
РГУПС
2025

СОДЕРЖАНИЕ

Лабораторная работа № 1

Настройка рабочего окружения и базовый синтаксис

Цель лабораторной работы: освоить установку локального веб-сервера, изучить базовый синтаксис языка PHP, научиться создавать и выполнять простые PHP-скрипты, понять принципы работы серверного программирования.

Методические указания

1. Установка локального сервера

Для установки Web-сервера и сервера СУБД необходимо скачать Denwer с сайта разработчика.

Denwer – набор дистрибутивов и программная оболочка, предназначенные для создания и отладки сайтов на локальном ПК под управлением ОС Windows. Базовый пакет содержит большинство необходимых программ и утилит:

- Инсталлятор.
- Apache, SSL, SSI, mod_rewrite, mod_php.
- PHP5 с поддержкой GD, MySQL, sqLite.
- MySQL5 с поддержкой транзакций.
- Система управления виртуальными хостами, основанная на шаблонах.
- Система управления запуском и завершением всех компонентов Denwer.
- phpMyAdmin – система управления MySQL через Web-интерфейс.
- Эмулятор sendmail и SMTP-сервера.

Установка Denwer:

1) Запустить скачанный установщик Denwer.
2) Появится сообщение о выборе каталога для установки комплекса (по умолчанию используется C:\WebServers, для выбора нажать Enter). 4
3) Далее необходимо ввести имя виртуального диска, который будет свя зан с только что указанной директорией. Рекомендуется использовать значение по умолчанию (Z:). Важно, что диска с этим именем еще не должно содержаться в системе.
4) После этого начнется копирование файлов дистрибутива, а под конец

будет задан вопрос, как именно запускать и останавливать комплекс. Есть два варианта:

а) Создавать виртуальный диск при загрузке машины, а при остановке серверов диск не отключать.

б) Создавать виртуальный диск только по явной команде старта ком плекса (при щелчке по ярлыку запуска на Рабочем столе). Отключать диск от системы при остановке серверов.

Итак, после выполнения выше описанных пунктов, будет совершена уста новка web-сервера и сервера СУБД.

Для проверки работы web-сервера и сервера СУБД необходимо запустить на рабочем столе ярлык Start Denwer, а затем, дождавшись, когда все консоль ные исчезнут, открыть браузер и набрать в нем адрес: http://localhost/denwer/.

2. Введение в PHP

PHP (рекурсивный акроним *PHP: Hypertext Preprocessor*) — это скриптовый язык общего назначения с открытым исходным кодом, специально разработанный для веб-разработки. В отличие от клиентских языков (JavaScript), PHP-код выполняется **на стороне сервера**. Это означает, что браузер получает уже готовый HTML-документ, сгенерированный PHP-скриптом.

Принцип работы

1. Пользователь запрашивает страницу (например, index.php).
2. Веб-сервер (Apache/Nginx) передаёт файл интерпретатору PHP.
3. Интерпретатор выполняет PHP-код, подставляя результаты вычислений.
4. Сервер отправляет клиенту чистый HTML.

Пример встраивания PHP в HTML

```
<!DOCTYPE html>
<html>
<body>
    <h1><?= "Заголовок страницы"; ?></h1>
</body>
</html>
```

3. Структура PHP-скрипта

– Файлы имеют расширение .php.

– Код обрамляется тегами <?php ... ?> (короткий вариант <? ... ?> не рекомендуется).

– Инструкции разделяются точкой с запятой ;.

– Регистр в названиях функций не имеет значения, но в переменных — имеет.

4. Переменные и типы данных

Переменные в PHP начинаются со знака $ и могут содержать буквы, цифры и подчёркивание (первый символ — буква или подчёркивание).

```
$name = "Алексей";  // Строка (string)
$age = 25;          // Целое число (integer)
$price = 99.99;     // Число с плавающей точкой (float)
$is_active = true;  // Логический тип (boolean)
```

PHP — язык с динамической типизацией: тип переменной определяется в момент присваивания значения.

5. Основные операторы

– Арифметические: +, -, *, /, % (остаток от деления), ** (возведение в степень).

– Конкатенация строк: оператор . (точка).

– Присваивания: =, +=, -=, .= и др.

– Инкремент/декремент: $a++, ++$a, $a--, --$a.

Пример:

```
$a = 10;
$b = 3;
echo $a + $b;       // 13
echo $a % $b;       // 1
echo "Результат: " . ($a * $b); // "Результат: 30"
```

6. Вывод данных

– echo — выводит одну или несколько строк.

– print — выводит строку, всегда возвращает 1.

– var_dump() — выводит информацию о переменной (тип, значение,

структура).

– print_r() — выводит информацию в удобочитаемом виде (массивы, объекты).

Задания для выполнения лабораторной работы

Задание 1: Базовые настройки

– Установите Denwer.

– Настройте Denwer для работы с PHP 7.4+ и Apache 2.4.

– Создайте домен studentXX (где XX — ваш номер по журналу).

– Проверьте работоспособность, создав файл test.php с выводом phpinfo().

Задание 2: Основы синтаксиса и переменные

Часть 1: Работа с переменными

1. Объявите переменные с вашими личными данными:

– Имя, фамилия, отчество (строки)

– Возраст (целое число)

– Средний балл за сессию (число с плавающей точкой)

– Являетесь ли старостой (логическое значение)

2. Выведите эти данные в виде форматированной HTML-таблицы:

```
<table class="personal-data">
    <tr><th>Параметр</th><th>Значение</th><th>Тип данных</th></tr>
    <!-- Данные вставляются через PHP -->
</table>
```

Часть 2: Вычисления и преобразования

1. Выполните вычисления с вашими данными:

– Через сколько лет вам будет 30?

– На сколько ваш средний балл отличается от 4.0?

– Сколько дней вы уже живете (примерно)?

2. Преобразуйте типы данных:

– Преобразуйте возраст в строку

– Преобразуйте средний балл в целое число

– Преобразуйте имя в верхний регистр

Часть 3: Динамическое содержимое

1. Создайте переменную $current_year с текущим годом.
2. Вычислите год вашего поступления в колледж (предположим, в 17 лет).
3. Создайте строку-биографию, используя конкатенацию:

Меня зовут [Имя Фамилия]. Мне [возраст] лет. Я учусь на [курс] курсе. Мой средний балл: [балл].

Задание 3: Арифметические и строковые операторы

Часть 1: Калькулятор арифметических операций

1. Создайте форму с двумя числовыми полями и выбором операции (+, -, *, /, %, **).
2. Реализуйте обработку формы с выполнением выбранной операции.
3. Добавьте валидацию:

– Проверка на пустые поля

– Проверка деления на ноль

– Проверка на числовой тип

Часть 2: Строковые манипуляции

1. Создайте генератор случайных паролей:

– Длина пароля: 8-12 символов

– Должен содержать буквы (верхний и нижний регистр), цифры и спецсимволы

– Используйте конкатенацию и строковые функции

2. Реализуйте анализ текста:

$text = "PHP - это популярный язык программирования для веб-разработки.";

Вычислите:

– Количество символов

- Количество слов
- Самые частые слова
- Замените "популярный" на "мощный"

Лабораторная работа № 2

Операции с базовыми типами данных

Цель лабораторной работы: Освоить работу с базовыми типами данных в PHP: выполнять арифметические и строковые операции, практиковать преобразование типов и конкатенацию строк.

Методические указания

1. Базовые типы данных в PHP

PHP поддерживает восемь основных типов данных, которые делятся на три категории:

1.1. Скалярные типы (простые):

- **integer (целое число)** — числа без дробной части: -10, 0, 42;
- **float/double (числа с плавающей точкой)** — дробные числа: 3.14, -2.5, 1.2e3;
- **string (строка)** — последовательность символов: "Привет", 'мир';
- **boolean (логический)** — true или false.

1.2. Составные типы:

- **array (массив)** — упорядоченная карта значений
- **object (объект)** — экземпляр класса

1.3. Специальные типы:

- **resource (ресурс)** — ссылка на внешний ресурс
- **NULL** — переменная без значения

2. Динамическая типизация

PHP — язык с динамической типизацией. Тип переменной определяется в момент присваивания значения и может меняться:

```
$var = 10;        // integer
$var = "текст";   // теперь string
$var = 3.14;      // теперь float
```

3. Арифметические операции

```
$a + $b     // Сложение
$a - $b     // Вычитание
```

```
$a * $b    // Умножение
$a / $b    // Деление
$a % $b    // Остаток от деления
$a ** $b   // Возведение в степень
++$a       // Преинкремент
$a++       // Постинкремент
--$a       // Предекремент
$a--       // Постдекремент
```

4. Строковые операции

```
$str1 . $str2       // Конкатенация (объединение)
$str1 .= $str2      // Конкатенация с присваиванием
strlen($str)        // Длина строки
strtoupper($str)    // Верхний регистр
strtolower($str)    // Нижний регистр
trim($str)          // Удаление пробелов с краёв
substr($str, 0, 5)  // Подстрока
```

5. Преобразование типов

PHP автоматически преобразует типы, но можно делать это явно:

```
(int) $var     // К целому
(float) $var   // К дробному
(string) $var  // К строке
(bool) $var    // К логическому
(array) $var   // К массиву
```

6. Особенности работы с типами

– **Строки в арифметике:** "10" + "5" = 15 (автоматическое преобразование)

– **Логический контекст:** 0, "", null, false, array() = false

– **Строгое сравнение:** === (сравнивает и значение, и тип)

Задания для выполнения лабораторной работы

Задание 1: Калькулятор базовых операций

1) Объявите переменные: $a = 15, $b = 4.
2) Выполните все арифметические операции и выведите результаты в таблице HTML.
3) Добавьте вычисление среднего арифметического.
4) Реализуйте возведение в степень и извлечение квадратного корня.
5) Продемонстрируйте разницу между пре- и постинкрементом.

```
<?php
$a = 15;
$b = 4;
?>
<table border="1">

<tr><th>Операция</th><th>Пример</th><th>Результат</th></tr>
    <!-- Ваш код здесь -->
</table>
```

Задание 2: Конвертер единиц измерения

1) Реализуйте конвертацию километров в мили (1 км = 0.621371 миль).
2) Реализуйте конвертацию килограммов в фунты (1 кг = 2.20462 фунтов).
3) Реализуйте конвертацию градусов Цельсия в Фаренгейты: °F = °C × 9/5 + 32.
4) Все вычисления проводите с точностью до 2 знаков после запятой.
5) Создайте форму для ввода значений и выбора типа конвертации.

Задание 3: Генератор случайных данных

1) Сгенерируйте случайное целое число от -100 до 100.
2) Сгенерируйте случайное дробное число от 0 до 1 с 4 знаками после запятой.
3) Создайте случайную строку из 8 символов (буквы и цифры).
4) Сгенерируйте случайный логический тип (true/false).
5) Выведите все значения с указанием их типа (используйте gettype()).

Задание 4: Обработка строк

1) Создайте строку: " PHP - это мощный язык программирования ".
2) Выполните последовательно:

– Удалите лишние пробелы по краям
– Приведите к верхнему регистру
– Замените "мощный" на "современный"

– Выделите подстроку "язык программирования"
– Разделите строку на слова

3) Посчитайте количество символов и слов.
4) Создайте "перевернутую" версию строки.

Задание 5: Анализ типа данных

1) Создайте массив с разными типами данных: [42, 3.14, "текст", true, null, [1,2]].
2) Для каждого элемента выведите:

– Значение
– Тип (gettype)
– Является ли числом (is_numeric)
– Является ли целым (is_int)
– Является ли строкой (is_string)

3) Продемонстрируйте явное преобразование типов.
4) Покажите разницу между == и === на примерах.

Задание 6: Калькулятор сложных выражений

1) Реализуйте вычисление выражения: $(a^2 + b^2) / (a - b) + \sqrt{(ab)}$
2) Реализуйте конкатенацию: "Сумма " . a . " и " . b . " равна " . (a+b)
3) Создайте строку-шаблон и подставьте в неё значения переменных.
4) Реализуйте вычисление процента: (часть / целое) * 100%
5) Выведите все результаты с форматированием (округление, добавление единиц измерения).

Лабораторная работа № 3

Реализация алгоритмов с использованием условных конструкций и циклов

Цель лабораторной работы: Научиться реализовывать алгоритмы с использованием условных конструкций и циклов, генерировать HTML-структуры на основе логических условий и итеративной обработки данных.

Методические указания

1. Условные конструкции

1.1. Конструкция if-elseif-else

```
if (условие1) {
    // код, если условие1 истинно
} elseif (условие2) {
    // код, если условие2 истинно
} else {
    // код, если все условия ложны
}
```

1.2. Тернарный оператор

```
$result = (условие) ? значение_если_истина : значение_если_ложь;
```

1.3. Оператор switch

```
switch ($переменная) {
    case значение1:
        // код
        break;
    case значение2:
        // код
        break;
    default:
        // код по умолчанию
}
```

2. Циклы

2.1. Цикл for — когда известно количество итераций

```
for ($i = 0; $i < 10; $i++) {
    // тело цикла
}
```

2.2. Цикл while — пока условие истинно

```
while (условие) {
    // тело цикла
}
```

2.3. Цикл do-while — выполняется хотя бы один раз

```
do {
    // тело цикла
} while (условие);
```

2.4. Цикл foreach — для перебора массивов

```
foreach ($массив as $значение) {
    // работа со значением
}

foreach ($массив as $ключ => $значение) {
    // работа с ключом и значением
}
```

3. Управление выполнением

– break — прерывает выполнение цикла или switch
– continue — пропускает текущую итерацию цикла
– break 2 — прерывает два вложенных цикла

4. Генерация HTML через PHP

PHP позволяет динамически генерировать HTML:

```
echo "<div class='container'>";
for ($i = 1; $i <= 5; $i++) {
    echo "<p>Параграф $i</p>";
}
echo "</div>";
```

Задания для выполнения лабораторной работы

Задание 1: Работа со временем

Напишите PHP-скрипт, который определяет текущий час с помощью функции date('H') и в зависимости от времени суток выводит соответствующее приветствие и оформление.

– **С 05:00 до 11:59:** "Доброе утро!" (вывести в теге <h1 style="color: orange;">)

– **С 12:00 до 17:59:** "Добрый день!" (вывести в теге <h1 style="color: blue;">)

– **С 18:00 до 22:59:** "Добрый вечер!" (вывести в теге <h1 style="color: purple;">)

– **В остальное время:** "Доброй ночи!" (вывести в теге <h1 style="color: darkblue;">)

Задание 2: Конвертор оценок

Создайте скрипт, который преобразует числовую оценку (по 5-балльной системе) в текстовое описание и буквенную систему (A-F).

– Присвойте переменной $grade случайное значение от 1 до 5 (используйте rand(1, 5)).

– Используя switch, определите текстовое описание и букву:

- 5: "Отлично" (A)
- 4: "Хорошо" (B)
- 3: "Удовлетворительно" (C)
- 2: "Неудовлетворительно" (F, требуется пересдача)
- 1: "Провал" (F, необходим повтор курса)

– Сгенерируйте HTML-список (<ul>) и выведите в нем:

- Сама числовая оценка.
- Ее текстовый эквивалент.
- Буквенное значение.

Задание 3: Генератор таблицы умножения

Напишите скрипт, который генерирует HTML-тацу умножения (Пифагора) размером 10x10.

– Используйте два вложенных цикла for.

– Внешний цикл ($i) для строк (от 1 до 10).

– Внутренний цикл ($j) для столбцов (от 1 до 10).

– Результат умножения $i * $j должен находиться в ячейке таблицы <td>.

Задание 4: Анализатор числа

Напишите программу, которая для заданного случайного числа $number = rand(10, 100);

1) Используя do-while: Выведите все целые числа от 1 до $number, пропуская каждое третье число (используйте continue).

2) Используя while: Определите, является ли число простым. Для этого ищите делители в цикле. Если найден делитель (кроме 1 и самого числа), выведите "Число $number - составное" и завершите цикл досрочно (break). Если делителей не найдено, выведите "Число $number - простое".

Выведите оба результата в виде двух отдельных блоков с пояснительным текстом.

Лабораторная работа № 4

Работа с индексированными и ассоциативными массивами

Цель лабораторной работы: Освоить работу с индексированными и ассоциативными массивами: создание, модификация, преобразование массивов и форматированный вывод элементов в виде HTML-разметки.

Методические указания

1. Типы массивов в PHP

1.1. Индексированные массивы — с числовыми индексами

```
$arr = [1, 2, 3, 4, 5];
// или
$arr = array(1, 2, 3, 4, 5);
// Доступ: $arr[0], $arr[1], ...
```

1.2. Ассоциативные массивы — со строковыми ключами

```
$user = [
    'name' => 'Иван',
    'age' => 25,
    'city' => 'Москва'
];
// Доступ: $user['name'], $user['age'], ...
```

1.3. Многомерные массивы — массивы, содержащие другие массивы

```
$students = [
    ['name' => 'Иван', 'grade' => 5],
    ['name' => 'Мария', 'grade' => 4],
    ['name' => 'Петр', 'grade' => 3]
];
```

2. Создание и модификация массивов

```
$arr = [];              // Создание пустого массива
$arr[] = 'элемент';     // Добавление в конец
array_push($arr, 'эл'); // Добавление в конец (функция)
array_pop($arr);        // Удаление последнего элемента
array_unshift($arr, 'эл'); // Добавление в начало
```

```
array_shift($arr);      // Удаление первого элемента
unset($arr[2]);         // Удаление элемента по индексу
```

3. Основные функции для работы с массивами

```
count($arr)             // Количество элементов
in_array('знач', $arr)  // Проверка наличия значения
array_key_exists('key', $arr) // Проверка наличия ключа
array_merge($arr1, $arr2)     // Слияние массивов
array_slice($arr, 2, 3)       // Выбор среза
array_splice($arr, 2, 1)      // Удаление/замена части
array_keys($arr)        // Получение всех ключей
array_values($arr)      // Получение всех значений
```

4. Перебор массивов

4.1. Цикл for (только для индексированных массивов)

```
for ($i = 0; $i < count($arr); $i++) {
    echo $arr[$i];
}
```

4.2. Цикл foreach (для любых массивов)

```
foreach ($arr as $value) {
    echo $value;
}

foreach ($arr as $key => $value) {
    echo "$key: $value";
}
```

4.3. Функции итераторы

```
while (list($key, $value) = each($arr)) {
    echo "$key: $value";
}
```

5. Сортировка массивов

```
sort($arr)              // Сортировка по значениям (индексы
```

```
сбрасываются)
    asort($arr)        // Сортировка по значениям (сохраняет ключи)
    ksort($arr)         // Сортировка по ключам
    rsort($arr)         // Обратная сортировка по значениям
    arsort($arr)             // Обратная сортировка по значениям
(сохраняет ключи)
    krsort($arr)        // Обратная сортировка по ключам
    usort($arr, 'func') // Пользовательская сортировка
```

Задания для выполнения лабораторной работы

Задание 1: Каталог товаров

1) Создайте массив товаров с ассоциативными массивами для каждого товара:

```
$products = [
    [
        'id' => 1,
        'name' => 'Ноутбук Lenovo',
        'price' => 45000,
        'category' => 'Электроника',
        'in_stock' => true
    ],
    // ещё 5-7 товаров
];
```

2) Сгенерируйте HTML-таблицу с товарами.

Задание 2: Расписание занятий

1) Создайте многомерный массив с расписанием на неделю:

```
$schedule = [
    'Понедельник' => [
        ['time' => '9:00', 'subject' => 'Математика', 'room' =>
'101'],
        ['time' => '10:45', 'subject' => 'Физика', 'room' =>
'203']
    ],
    // остальные дни
];
```

2) Сгенерируйте HTML-таблицу расписания.
3) Добавьте подсчёт количества пар в день и за неделю.

Задание 3: Конвертер данных в HTML-списки

1) Создайте одномерный массив имен и наполните его значениями.
2) Выведите данные из массива в виде списков: ul (маркированный), ol (нумерованный).
3) Реализуйте поддержку многомерных массивов (вложенные списки).
4) Добавьте возможность стилизации элементов списка.

Задание 4: Статистика оценок студентов

1) Создайте массив студентов с их оценками по предметам:

```
$students = [
    'Иванов' => ['Математика' => 5, 'Физика' => 4, 'Информатика' => 5],
    'Петров' => ['Математика' => 3, 'Физика' => 4, 'Информатика' => 4],
    // ещё 5-8 студентов
];
```

2) Рассчитайте и выведите:

– Средний балл каждого студента
– Средний балл по каждому предмету
– Лучшего студента по среднему баллу

3) Сгенерируйте сводную таблицу с результатами.

Задание 5: Манипуляции с массивами

1) Выполните следующие операции с массивами:

```
// Исходные данные
$numbers = [5, 2, 8, 1, 9, 3, 7, 4, 6];
$users = [
    ['id' => 1, 'name' => 'Иван', 'age' => 25],
    ['id' => 2, 'name' => 'Мария', 'age' => 30],
    ['id' => 3, 'name' => 'Петр', 'age' => 22]
];
```

2) Задачи:

– Отсортируйте массив чисел по возрастанию и убыванию

– Отфильтруйте пользователей старше 25 лет

– Преобразуйте массив пользователей в ассоциативный массив [id => user]

– Найдите максимальный и минимальный возраст

– Сгруппируйте пользователей по первой букве имени

– Преобразуйте массив чисел в строку через запятую

3) Каждую операцию сопровождайте выводом результата в читаемом виде.

Лабораторная работа № 5

Структурирование приложения с помощью пользовательских функций

Цель лабораторной работы: Научиться создавать пользовательские функции, организовывать код в модули, создавать библиотеки вспомогательных функций для выполнения типовых задач обработки данных.

Методические указания

1. Пользовательские функции в PHP

Пользовательские функции позволяют структурировать код, избегать повторений и создавать переиспользуемые блоки логики.

1.1. Синтаксис объявления функции:

```
Function имя_функции($параметр1, $параметр2 = 'значение_по_умолчанию') {
    // Тело функции
    return $результат;
}
```

1.2. Параметры функции:

– Обязательные параметры: function test($param);

– Необязательные параметры со значениями по умолчанию: function test($param = 'default');

– Переменное количество параметров: function sum(...$numbers).

1.3. Возврат значений:

– return — завершает выполнение функции и возвращает значение;

– Функция может возвращать любой тип данных;

– Если return не указан, функция возвращает null.

2. Область видимости переменных

– **Локальные переменные** — существуют только внутри функции;

– **Глобальные переменные** — доступны во всем скрипте;

– **Статические переменные** — сохраняют значение между вызовами функции.

```
function counter() {
```

```
    static $count = 0;
    $count++;
    return $count;
}
```

3. Анонимные функции и замыкания

```
$multiply = function($a, $b) {
    return $a * $b;
};
echo $multiply(5, 3); // 15
```

4. Рекурсивные функции

Функции, которые вызывают сами себя:

```
function factorial($n) {
    if ($n <= 1) return 1;
    return $n * factorial($n - 1);
}
```

5. Библиотеки функций

Создание отдельных файлов с функциями для повторного использования:

```
// functions.php
function formatPrice($price) {
    return number_format($price, 2, ',', ' ') . ' руб.';
}

// main.php
require_once 'functions.php';
echo formatPrice(1500.5); // 1 500,50 руб.
```

Задания для выполнения лабораторной работы

Задание 1: Математические функции

Реализуйте следующие функци.

- sum($a, $b) — сложение двух чисел
- roundTo($number, $decimals) — округление до указанного знака

– circleArea($radius) — площадь круга

Задание 2: Библиотека строковых операций

Реализуйте следующие функци.

– truncate($text, $length) — обрезает текст до указанной длины
– countVowels($text) — считает количество гласных букв
– camelCase($text) — преобразует в camelCase

Задание 3: Библиотека работы с датами

Реализуйте следующие функци.

– formatDate($date, $format) — форматирует дату по шаблону
– isWeekend($date) — проверяет, выходной ли день
– getSeason($date) — определяет время года

Лабораторная работа № 6

Обработка данных, передаваемых через GET-запрос

Цель лабораторной работы: Научиться обрабатывать данные, передаваемые через GET-запрос, читать параметры из URL, реализовывать фильтрацию и вывод данных на основе переданных пользователем критериев.

Методические указания

1. Метод GET в HTTP

GET — один из основных методов HTTP, используется для запроса данных с сервера.

1.1. Особенности GET-запросов:

- Параметры передаются в URL-строке;
- Ограничение длины (обычно 2048 символов);
- Данные видны пользователю;
- Кэшируются браузером;
- Безопасны (идемпотентны).

1.2. Структура URL с GET-параметрами:

```
http://example.com/page.php?параметр1=значение1&параметр2=значение2
```

2. Работа с GET-параметрами в PHP

2.1. Суперглобальный массив $_GET:

```
// URL: page.php?name=John&age=25
$name = $_GET['name']; // "John"
$age = $_GET['age'];   // "25"
```

2.2. Проверка наличия параметров:

```
if (isset($_GET['param'])) {
    // параметр существует
}
if (!empty($_GET['param'])) {
    // параметр не пустой
}
```

3. Безопасная обработка GET-данных

3.1. Фильтрация данных:

```
$id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
$email = filter_input(INPUT_GET, 'email', FILTER_VALIDATE_EMAIL);
```

3.2. Экранирование вывода:

```
echo htmlspecialchars($_GET['search'], ENT_QUOTES, 'UTF-8');
```

4. Построение URL с параметрами

4.1. Функция http_build_query():

```
$params = ['page' => 2, 'sort' => 'price', 'order' => 'desc'];
$url = 'products.php?' . http_build_query($params);
// products.php?page=2&sort=price&order=desc
```

4.2. Добавление параметров к текущему URL:

```
$current_params = $_GET;
$current_params['page'] = 2;
$new_url = $_SERVER['PHP_SELF'] . '?' . http_build_query($current_params);
```

5. Пагинация и фильтрация

Типичный пример использования GET-параметров:

```
search.php?query=php&category=books&price_min=100&price_max=1000&page=2
```

Задания для выполнения лабораторной работы

Задание 1: Система поиска товаров

1) Создайте массив товаров:

```
$products = [
    ['id' => 1, 'name' => 'Ноутбук Lenovo', 'category' => 'электроника', 'price' => 45000],
    ['id' => 2, 'name' => 'Смартфон iPhone', 'category' => 'электроника', 'price' => 80000],
```

```
        ['id' => 3, 'name' => 'Книга по PHP', 'category' => 'книги',
'price' => 1500],
        // ещё 15-20 товаров
    ];
```

2) Параметры поиска через GET:

   – category — категория товара
   – min_price, max_price — диапазон цен
   – sort — сортировка (price_asc, price_desc, name)
   – page — номер страницы

3) Реализуйте функционал:

   – Поиск по названию товара
   – Фильтрация по категории и цене
   – Сортировка результатов
   – Пагинация (по 5 товаров на странице)

4) Создайте форму поиска с сохранением параметров в URL.

Задание 2: Каталог фильмов с фильтрами

1) Создайте массив фильмов:

```
$movies = [
    [
        'id' => 1,
        'title' => 'Интерстеллар',
        'year' => 2014,
        'genre' => ['фантастика', 'драма'],
        'rating' => 8.6,
        'duration' => 169
    ],
    // ещё 20-30 фильмов
];
```

2) Многоуровневая фильтрация:

   – По жанру (можно выбрать несколько)
   – По году выпуска (диапазон)

- По рейтингу (мин/макс)
- По длительности

3) Особенности:

- Сохранение выбранных фильтров в URL
- Сброс фильтров кнопкой
- Подсчет найденных фильмов

Лабораторная работа № 7

Обработка данных, передаваемых через POST-запрос

Цель лабораторной работы: Научиться обрабатывать данные, передаваемые через POST-запрос, создавать HTML-формы различной сложности, реализовывать валидацию введённых пользователем данных на стороне сервера.

Методические указания

1. Метод POST в HTTP

POST — метод HTTP для отправки данных на сервер для обработки.

1.1. Особенности POST-запросов:

– Данные передаются в теле запроса;

– Нет ограничений на длину (кроме настроек сервера);

– Данные не видны в URL;

– Не кэшируются браузером;

– Используются для создания/изменения данных.

1.2. Когда использовать POST:

– Отправка форм (регистрация, логин, контакты);

– Загрузка файлов;

– Создание новых записей в БД;

– Отправка конфиденциальных данных.

2. Работа с POST-данными в PHP

2.1. Суперглобальный массив $_POST:

```
// Форма с полями name и email
$name = $_POST['name'];
$email = $_POST['email'];
```

2.2. Проверка метода запроса:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Обработка POST-запроса
}
```

3. Создание HTML-форм

3.1. Базовая форма:

```
<form method="POST" action="process.php">
    <input type="text" name="username">
    <input type="submit" value="Отправить">
</form>
```

3.2. Типы полей формы:

- text, password, email, number — текстовые поля;
- textarea — многострочный текст;
- select — выпадающий список;
- checkbox — флажки (множественный выбор);
- radio — переключатели;
- file — загрузка файлов;
- hidden — скрытые поля.

4. Валидация данных на стороне сервера

4.1. Типы валидации:

- Проверка на пустоту;
- Проверка формата (email, URL, телефон);
- Проверка длины;
- Проверка типа данных;
- Проверка уникальности.

4.2. Обработка ошибок:

```
$errors = [];

if (empty($_POST['name'])) {
    $errors['name'] = 'Имя обязательно';
}

if (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    $errors['email'] = 'Неверный формат email';
}
```

5. Защита форм

5.1. CSRF-токены:

```
// Генерация токена
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
// Проверка в форме
<input type="hidden" name="csrf_token" value="<?=
$_SESSION['csrf_token'] ?>">
// Валидация
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('CSRF атака!');
}
```

5.2. Защита от переполнения:

- Ограничение длины полей
- Валидация типа файлов
- Проверка размера загружаемых файлов

Задания для выполнения лабораторной работы

Задание 1: Система регистрации пользователей

1) Сверстайте форму регистрации:

```
<form method="POST" action="process_register.php">
    <!-- Персональные данные -->
    <input type="text" name="first_name" placeholder="Имя"
required>
    <input type="text" name="last_name" placeholder="Фамилия"
required>

    <!-- Контактные данные -->
    <input type="email" name="email" placeholder="Email"
required>
    <input type="tel" name="phone" placeholder="Телефон">

    <!-- Учетные данные -->
    <input type="text" name="username" placeholder="Логин"
required>
```

```
        <input          type="password"          name="password"
placeholder="Пароль" required>
        <input       type="password"       name="confirm_password"
placeholder="Подтверждение пароля" required>

        <!-- Дополнительная информация -->
        <select name="country">
            <option value="">Выберите страну</option>
            <option value="ru">Россия</option>
            <option value="by">Беларусь</option>
        </select>

        <textarea name="bio" placeholder="О себе"></textarea>

        <!-- Соглашения -->
        <input type="checkbox" name="terms" required>
        Я согласен с условиями использования

        <button type="submit">Зарегистрироваться</button>
    </form>
```

2) Валидация на стороне сервера:

- Проверка уникальности email и логина
- Сложность пароля (мин. 8 символов, буквы+цифры)
- Совпадение паролей
- Формат телефона
- Принятие условий использования

2. **Реализуйте:**

- Хэширование паролей (password_hash)
- Сохранение данных в массив

Лабораторная работа № 8

Работа с файловыми загрузками от пользователя

Цель лабораторной работы: Научиться обрабатывать данные, передаваемые через POST-запрос, создавать HTML-формы различной сложности, реализовывать валидацию введённых пользователем данных на стороне сервера.

Методические указания

1. Загрузка файлов через HTTP

Файлы передаются на сервер через форму с методом POST и специальным типом кодирования.

1.1. Конфигурация PHP для загрузки:

```
; php.ini настройки
file_uploads = On
upload_max_filesize = 2M
max_file_uploads = 20
post_max_size = 8M
```

1.2. Форма для загрузки файлов:

```
<form method="POST" action="upload.php"
enctype="multipart/form-data">
    <input type="file" name="userfile">
    <input type="submit" value="Загрузить">
</form>
```

2. Суперглобальный массив $_FILES

После отправки формы данные о файле доступны в массиве $_FILES['userfile']:

```
$_FILES['userfile']['name']     // Исходное имя файла
$_FILES['userfile']['type']     // MIME-тип
$_FILES['userfile']['size']     // Размер в байтах
$_FILES['userfile']['tmp_name'] // Временный путь на сервере
$_FILES['userfile']['error']    // Код ошибки
```

3. Коды ошибок загрузки

```
UPLOAD_ERR_OK (0)        // Успешная загрузка
UPLOAD_ERR_INI_SIZE (1)  // Превышен upload_max_filesize
```

```
UPLOAD_ERR_FORM_SIZE (2) // Превышен MAX_FILE_SIZE из формы
UPLOAD_ERR_PARTIAL (3)   // Файл загружен частично
UPLOAD_ERR_NO_FILE (4)   // Файл не был загружен
UPLOAD_ERR_NO_TMP_DIR (6) // Отсутствует временная директория
UPLOAD_ERR_CANT_WRITE (7) // Ошибка записи на диск
UPLOAD_ERR_EXTENSION (8)  // Остановлено расширением PHP
```

4. Безопасная обработка загрузок

4.1. Валидация файлов:

- Проверка MIME-типа;
- Проверка расширения;
- Проверка размера;
- Проверка на вирусы (антивирусное сканирование);
- Переименование файлов.

4.2. Защитные меры:

- Ограничение типов файлов;
- Проверка содержимого файлов;
- Изоляция загруженных файлов;
- Генерация уникальных имен.

5. Функции для работы с загрузками

```
is_uploaded_file($tmp_name)     // Проверка, что файл загружен через POST
move_uploaded_file($tmp_name, $destination)   // Безопасное перемещение
```

Задания для выполнения лабораторной работы

Задание 1: Система загрузки изображений

Реализуйте систему загрузки изображений.

1) Форма загрузки:

```
<form method="POST" action="process_image.php" enctype="multipart/form-data">
    <input type="file" name="image" accept="image/*" required>
```

```
        <input type="text" name="title" placeholder="Название
изображения">
        <textarea name="description"
placeholder="Описание"></textarea>
        <select name="category">
            <option value="nature">Природа</option>
            <option value="city">Город</option>
            <option value="portrait">Портрет</option>
        </select>
        <input type="checkbox" name="watermark"> Добавить водяной
знак
        <input type="submit" value="Загрузить">
    </form>
```

2) Валидация изображений:

– Максимальный размер: 5MB

– Разрешенные форматы: JPEG, PNG, GIF, WebP

3) Реализуйте галерею с отображением загруженных изображений и их метаданных.

Задание 2: Система загрузки документов

1) Множественная загрузка:

```
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="documents[]" multiple
accept=".pdf,.doc,.docx,.txt">
        <input type="text" name="project_name"
placeholder="Название проекта">
        <input type="submit" value="Загрузить документы">
    </form>
```

2) Валидация документов:

– Максимальный размер каждого файла: 10MB

– Общий размер пакета: 50MB

– Максимальное количество файлов: 10

– Проверка содержимого (не пустые файлы)

3) Реализуйте систему управления документами с поиском по содержимому.

Лабораторная работа № 9

Взаимодействие с файловой системой на стороне сервера

Цель лабораторной работы: Научиться читать из и записывать в текстовые файлы, организовать хранение данных приложения в файловой системе, безопасно работать с файлами и директориями на стороне сервера.

Методические указания

1. Основные функции работы с файлами

1.1. Открытие и закрытие файлов:

```
$file = fopen("file.txt", "r"); // Открытие для чтения
fclose($file); // Закрытие файла
```

1.2. Режимы открытия файлов:

```
r  - чтение (курсор в начале)
r+ - чтение и запись (курсор в начале)
w  - запись (создает/очищает файл)
w+ - чтение и запись (создает/очищает)
a  - запись (курсор в конец, создает файл)
a+ - чтение и запись (курсор в конец)
x  - эксклюзивное создание (ошибка если существует)
x+ - эксклюзивное создание для чтения/записи
```

2. Чтение из файлов

2.1. Различные методы чтения:

```
fread($file, $length)     // Чтение указанного количества байт
fgets($file)              // Чтение строки
fgetc($file)              // Чтение символа
file_get_contents($path) // Чтение всего файла в строку
file($path)               // Чтение файла в массив строк
```

2.2. Навигация по файлу:

```
fseek($file, $offset)     // Перемещение указателя
ftell($file)              // Текущая позиция указателя
rewind($file)             // Сброс указателя в начало
feof($file)               // Проверка конца файла
```

3. Запись в файлы

```
fwrite($file, $data)      // Запись данных
file_put_contents($path, $data) // Запись всего содержимого
fputs($file, $data)       // Алиас для fwrite
```

4. Работа с директориями

```
mkdir($path)              // Создание директории
rmdir($path)              // Удаление директории
is_dir($path)             // Проверка, является ли директорией
scandir($path)            // Список файлов в директории
```

5. Информация о файлах

```
file_exists($path)        // Существует ли файл
filesize($path)           // Размер файла
filemtime($path)          // Время последнего изменения
filectime($path)          // Время создания
fileatime($path)          // Время последнего доступа
is_readable($path)        // Проверка прав чтения
is_writable($path)        // Проверка прав записи
```

6. Безопасность работы с файлами

- Валидация путей;
- Ограничение доступа к системным файлам;
- Экранирование специальных символов;
- Проверка прав доступа.

Задания для выполнения лабораторной работы

Задание 1: Файловый менеджер

Реализуйте файловый менеджер.

1) Базовый функционал:

```
class FileManager {
    private $basePath;
```

```
    public function __construct($basePath = './') {
        $this->basePath = realpath($basePath);
    }

    public function listDirectory($path = '') {
        $fullPath = $this->getFullPath($path);
        $items = scandir($fullPath);

        $result = [
            'files' => [],
            'directories' => []
        ];

        foreach ($items as $item) {
            if ($item === '.' || $item === '..') continue;

            $itemPath = $fullPath . '/' . $item;
            $itemInfo = [
                'name' => $item,
                'path' => $path . '/' . $item,
                'size'      =>      is_file($itemPath)      ?
filesize($itemPath) : null,
                'modified' => filemtime($itemPath),
                'permissions'     =>     substr(sprintf('%o',
fileperms($itemPath)), -4)
            ];

            if (is_dir($itemPath)) {
                $result['directories'][] = $itemInfo;
            } else {
                $result['files'][] = $itemInfo;
            }
        }

        return $result;
    }
```

```
        // Другие методы: create, rename, delete, copy, move
    }
```

2) Расширенный функционал:

   – Загрузка файлов
   – Редактирование текстовых файлов
   – Просмотр изображений
   – Поиск по файлам
   – Пакетные операции

3) Реализуйте полноценный веб-файловый менеджер с интерфейсом, похожим на проводник.

Лабораторная работа № 10

Реализация механизма аутентификации с использованием сессий

Цель лабораторной работы: Научиться реализовывать механизм аутентификации с использованием сессий, организовывать защиту разделов веб-приложения, управлять состоянием пользователя в рамках сессии.

Методические указания

1. Сессии в PHP

1.1. Что такое сессия?

Сессия — механизм сохранения данных между запросами одного пользователя.

1.2. Инициализация сессии:

```
session_start(); // Должно быть первой строкой, до любого вывода
```

1.3. Работа с сессионными данными:

```
$_SESSION['user_id'] = 1;       // Запись в сессию
$userId = $_SESSION['user_id']; // Чтение из сессии
unset($_SESSION['user_id']);    // Удаление из сессии
session_destroy();              // Уничтожение сессии
```

2. Настройки сессий

2.1. Параметры в php.ini:

```
session.save_handler = files
session.save_path = "/tmp"
session.name = PHPSESSID
session.cookie_lifetime = 0
session.gc_maxlifetime = 1440
```

2.2. Настройки через код:

```
ini_set('session.cookie_httponly', 1);
ini_set('session.use_strict_mode', 1);
ini_set('session.cookie_secure', 1); // Только для HTTPS
```

3. Безопасность сессий

3.1. Защита от угроз:

– Фиксация сессии (session fixation);
– Перехват сессии (session hijacking);
– XSS-атаки через сессии.

3.2. Меры защиты:

```
// Регенерация ID сессии
session_regenerate_id(true);

// Проверка user-agent
$_SESSION['user_agent'] = $_SERVER['HTTP_USER_AGENT'];

// Проверка IP (осторожно с динамическими IP)
$_SESSION['ip_address'] = $_SERVER['REMOTE_ADDR'];
```

4. Хранение данных пользователя

4.1. Базовая структура:

```
$_SESSION['user'] = [
    'id' => 1,
    'username' => 'john',
    'email' => 'john@example.com',
    'role' => 'user',
    'last_login' => time()
];
```

5. Управление доступом

5.1. Middleware для проверки доступа:

```
function requireAuth() {
    if (!isset($_SESSION['user'])) {
        header('Location: /login.php');
        exit;
    }
}

function requireAdmin() {
    requireAuth();
```

```
        if ($_SESSION['user']['role'] !== 'admin') {
            header('HTTP/1.0 403 Forbidden');
            exit;
        }
    }
```

Задания для выполнения лабораторной работы

Задание 1: Создание системы регистрации и аутентификации

1) База пользователей (в файле):

```
// users.json
[
    {
        "id": 1,
        "username": "admin",
        "email": "admin@example.com",
        "password_hash": "$2y$10$...", // password_hash()
        "role": "admin",
        "created_at": "2024-01-01 00:00:00",
        "last_login": "2024-03-15 10:30:00"
    }
]
```

2) Регистрация пользователя:
   - Валидация email и пароля
   - Проверка уникальности username/email
   - Хэширование пароля (password_hash)
3) Аутентификация:
   - Проверка логина/пароля
   - Запоминание сессии
   - Ограничение попыток входа
   - Блокировка при подозрительной активности
4) Реализуйте:
   - Восстановление пароля
   - Изменение пароля

Задание 2: Создание системы ролей и разрешений

1) Структура ролей:

```
$roles = [
    'guest' => [
        'permissions' => ['view_public_pages']
    ],
    'user' => [
        'permissions'   =>   ['view_profile',   'edit_profile',
'create_content'],
        'inherits' => ['guest']
    ],
    'moderator' => [
        'permissions'          =>          ['moderate_content',
'delete_comments'],
        'inherits' => ['user']
    ],
    'admin' => [
        'permissions'  =>  ['manage_users',  'manage_settings',
'view_logs'],
        'inherits' => ['moderator']
    ]
];
```

2) Проверка разрешений:

```
class PermissionChecker {
    public static function can($permission) {
        $userRole = $_SESSION['user']['role'] ?? 'guest';
        $userPermissions                                    =
self::getPermissionsForRole($userRole);

        return in_array($permission, $userPermissions);
    }

    public static function requirePermission($permission) {
        if (!self::can($permission)) {
            header('HTTP/1.0 403 Forbidden');
```

```
                include '403.php';
                exit;
            }
    }
}
```

3) Реализуйте:

- Управление ролями через админ-панель
- Назначение ролей пользователям
- Удаление ролей у пользователя

Задание 3: Управление сессиями пользователя

1) Менеджер сессий:

```
class SessionManager {
    public static function createSession($userId, $remember =
false) {
        session_regenerate_id(true);

        $_SESSION['user_id'] = $userId;
        $_SESSION['created'] = time();
        $_SESSION['last_activity'] = time();
        $_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
        $_SESSION['user_agent'] = $_SERVER['HTTP_USER_AGENT'];

        if ($remember) {
            self::setRememberToken($userId);
        }

        self::logSession($userId, 'login');
    }

    public static function validateSession() {
        if (!isset($_SESSION['user_id'])) {
            return false;
        }

        // Проверка таймаута (30 минут)
```

```
            if (time() - $_SESSION['last_activity'] > 1800) {
                self::destroySession();
                return false;
            }

            // Проверка user-agent
            if             ($_SESSION['user_agent']             !==
$_SERVER['HTTP_USER_AGENT']) {
                self::destroySession();
                return false;
            }

            $_SESSION['last_activity'] = time();
            return true;
        }
    }
```

2) Реализуйте:

- Просмотр активных сессий
- Принудительное завершение сессий
- Ограничение количества сессий

Лабораторная работа № 11

Работа с механизмом cookies

Цель лабораторной работы: Научиться использовать cookies для долговременного хранения пользовательских предпочтений, реализации функционала "запомнить меня" и персонализации веб-приложений.

Методические указания

1. Что такое cookies?

Cookies (куки) — небольшие текстовые файлы, хранящиеся в браузере пользователя и отправляемые серверу при каждом запросе.

1.1. Основные характеристики:

– Хранятся на стороне клиента;

– Ограниченный размер (обычно 4KB на cookie);

– Могут иметь срок жизни;

– Отправляются автоматически с каждым запросом.

2. Установка cookies в PHP

```
// Установка простой cookie
setcookie('username', 'john', time() + 3600, '/');

// Безопасная установка cookie
setcookie(
    'session_id',          // Имя
    'abc123',              // Значение
    time() + 86400 * 30,   // Время жизни (30 дней)
    '/',                   // Путь
    'example.com',         // Домен
    true,                  // Только HTTPS
    true                     // Только HTTP (не доступно через JS)
);
```

3. Чтение cookies

```
// Чтение cookie
if (isset($_COOKIE['username'])) {
```

```
    $username = $_COOKIE['username'];
}

// Суперглобальный массив $_COOKIE содержит все cookies
foreach ($_COOKIE as $name => $value) {
    echo "$name: $value<br>";
}
```

4. Удаление cookies

```
// Удаление cookie (установка времени в прошлое)
setcookie('username', '', time() - 3600, '/');
```

5. Безопасность работы с cookies

5.1. Основные угрозы:

- XSS-атаки (кража cookies через JavaScript);
- Перехват трафика (особенно без HTTPS);
- Подделка cookies.

5.2. Рекомендации по безопасности:

- Использовать флаг HttpOnly;
- Использовать флаг Secure для HTTPS;
- Валидировать содержимое cookies;
- Использовать подписанные cookies.

6. Сравнение cookies и сессий

Параметр	Cookies	Сессии
Место хранения	Браузер клиента	Сервер
Объем данных	Ограничен (4KB)	Большой
Безопасность	Ниже	Выше

Параметр	Cookies	Сессии
Срок хранения	Задается явно	До закрытия браузера
Доступность	Доступны JavaScript	Не доступны JS

Задания для выполнения лабораторной работы

Задание 1: Реализация системы "запомнить меня"

1) Механизм запоминания:

```
class RememberMe {
    private $cookieName = 'remember_token';
    private $tokenLength = 32;

    public function setToken($userId) {
        $token = bin2hex(random_bytes($this->tokenLength));
        $expires = time() + 86400 * 30; // 30 дней

        // Сохраняем в базу данных (файл для демо)
        $this->saveToken($userId, $token);

        // Устанавливаем cookie
        setcookie(
            $this->cookieName,
            $token,
            $expires,
            '/',
            '',
            true,  // Secure
            true   // HttpOnly
        );
    }

    public function validateToken() {
        if (!isset($_COOKIE[$this->cookieName])) {
```

```
            return false;
        }

        $token = $_COOKIE[$this->cookieName];
        $userId = $this->getUserIdByToken($token);

        if ($userId) {
            $_SESSION['user_id'] = $userId;
            $this->refreshToken($userId);
            return true;
        }

        return false;
    }
}
```

2) Реализуйте:

– Автоматический вход при наличии валидного токена

– Обновление токена при каждом успешном входе

– Отзыв всех токенов при смене пароля

– Управление активными сессиями

Задание 2: Реализация система пользовательских предпочтений

1) Хранение настроек:

```
class UserPreferences {
    private $preferences = [];

    public function __construct() {
        $this->loadPreferences();
    }

    public function set($key, $value) {
        $this->preferences[$key] = $value;
        $this->savePreferences();
    }
```

```
        public function get($key, $default = null) {
            return $this->preferences[$key] ?? $default;
        }

        private function loadPreferences() {
            if (isset($_COOKIE['preferences'])) {
                $data    =    json_decode($_COOKIE['preferences'],
true);
                if (is_array($data)) {
                    $this->preferences = $data;
                }
            }
        }

        private function savePreferences() {
            $data = json_encode($this->preferences);
            setcookie('preferences', $data, time() + 86400 * 365,
'/');
        }
    }
```

2) Настройки для реализации:

– Тема оформления (светлая/темная)
– Язык интерфейса
– Количество элементов на странице
– Сортировка по умолчанию
– Уведомления и оповещения

3) Создайте интерфейс настроек с мгновенным сохранением изменений.

Задание 3: Корзина покупок на cookies

1) Класс корзины:

```
class Cart {
    private $cookieName = 'shopping_cart';

    public function addItem($productId, $quantity = 1) {
        $cart = $this->getCart();
```

```
            if (isset($cart[$productId])) {
                $cart[$productId] += $quantity;
            } else {
                $cart[$productId] = $quantity;
            }

            $this->saveCart($cart);
        }

        public function removeItem($productId) {
            $cart = $this->getCart();
            unset($cart[$productId]);
            $this->saveCart($cart);
        }

        public function getCart() {
            if (isset($_COOKIE[$this->cookieName])) {
                return    json_decode($_COOKIE[$this->cookieName],
true);
            }
            return [];
        }

        private function saveCart($cart) {
            $data = json_encode($cart);
            setcookie($this->cookieName, $data, time() + 86400 * 7,
'/');
        }

        public function getTotalItems() {
            $cart = $this->getCart();
            return array_sum($cart);
        }
    }
```

2) Дополнительный функционал:

– Сохранение корзины между сессиями
– Ограничение времени жизни корзины

3) Реализуйте AJAX-обновление корзины без перезагрузки страницы.

Лабораторная работа № 12

Безопасная обработка пользовательского ввода

Цель лабораторной работы: Научиться безопасно обрабатывать пользовательский ввод, применять функции для экранирования и очистки внешних данных перед их выводом в HTML или сохранением в базу данных.

Методические указания

1. Типы угроз безопасности

1.1. XSS (Cross-Site Scripting):

– Внедрение JavaScript-кода через пользовательский ввод;

– Может привести к краже cookies, перенаправлению, модификации страниц.

1.2. SQL Injection:

– Внедрение SQL-кода через формы ввода;

– Позволяет читать, изменять, удалять данные БД.

1.3. CSRF (Cross-Site Request Forgery):

– Выполнение действий от имени авторизованного пользователя;

– Использует доверие браузера к сайту.

2. Функции для очистки данных

2.1. Экранирование HTML:

```
// Преобразование специальных символов в HTML-сущности
htmlspecialchars($string, ENT_QUOTES, 'UTF-8');
htmlentities($string, ENT_QUOTES, 'UTF-8');
```

2.2. Удаление тегов:

```
// Удаление всех HTML/PHP тегов
strip_tags($string, '<p><br>'); // Можно разрешить определенные теги
```

2.3. Очистка строк:

```
trim($string);          // Удаление пробелов по краям
addslashes($string);    // Экранирование специальных символов
```

```
mysqli_real_escape_string($connection, $string); // Для SQL
```

3. Валидация данных

3.1. Проверка типов:

```
filter_var($email, FILTER_VALIDATE_EMAIL);
filter_var($url, FILTER_VALIDATE_URL);
filter_var($ip, FILTER_VALIDATE_IP);
filter_var($int, FILTER_VALIDATE_INT);
```

3.2. Санитизация:

```
filter_var($string, FILTER_SANITIZE_STRING);
filter_var($email, FILTER_SANITIZE_EMAIL);
filter_var($url, FILTER_SANITIZE_URL);
filter_var($int, FILTER_SANITIZE_NUMBER_INT);
```

4. Подготовленные выражения для SQL

```
// С использованием PDO
$stmt = $pdo->prepare("SELECT * FROM users WHERE email =
:email");
$stmt->execute(['email' => $email]);
$user = $stmt->fetch();
// С использованием MySQLi
$stmt = $mysqli->prepare("SELECT * FROM users WHERE email =
?");
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();
```

5. CSRF-токены

```
// Генерация токена
$csrf_token = bin2hex(random_bytes(32));
$_SESSION['csrf_token'] = $csrf_token;
// Проверка в форме
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('CSRF атака обнаружена!');
```

```
}
```

Задания для выполнения лабораторной работы

Задание 1: Система комментариев с безопасным вводом

1) Класс безопасной обработки комментариев:

```
class SafeCommentProcessor {
    private $allowedTags = ['b', 'i', 'u', 'code', 'quote'];
    private $maxLength = 1000;

    public function processComment($input) {
        // 1. Проверка длины
        if (mb_strlen($input) > $this->maxLength) {
            throw new Exception('Комментарий слишком длинный');
        }

        // 2. Базовое экранирование
        $cleaned = htmlspecialchars($input, ENT_QUOTES, 'UTF-
8');

        // 3. Безопасное разрешение некоторых тегов
        $cleaned = $this->safeAllowTags($cleaned);

        // 4. Проверка на запрещенные слова
        $cleaned = $this->filterBadWords($cleaned);

        // 5. Удаление повторяющихся пробелов
        $cleaned = preg_replace('/\s+/', ' ', $cleaned);

        return trim($cleaned);
    }

    private function safeAllowTags($text) {
        // Разрешаем только безопасные теги с закрытием
        $pattern = '/&lt;(\/?(b|i|u|code|quote))&gt;/i';
        $replacement = '<$1>';
        return preg_replace($pattern, $replacement, $text);
```

```
        }

        private function filterBadWords($text) {
            $badWords = ['спам', 'реклама', 'оскорбление'];
            foreach ($badWords as $word) {
                $pattern = '/\b' . preg_quote($word, '/') . '\b/i';
                $text = preg_replace($pattern, '***', $text);
            }
            return $text;
        }
    }
```

2) Реализуйте:

– Предпросмотр комментария

– Модерацию комментариев

– Оповещение о подозрительном контенте

– Систему жалоб на комментарии

Задание 2: Защищенная форма поиска

1) Безопасный поисковой движок:

```
class SecureSearch {
    private $maxQueryLength = 100;
    private $minQueryLength = 2;

    public function sanitizeQuery($query) {
        // 1. Обрезка и проверка длины
        $query = trim($query);
        if (mb_strlen($query) > $this->maxQueryLength) {
            $query = mb_substr($query, 0, $this->maxQueryLength);
        }

        if (mb_strlen($query) < $this->minQueryLength) {
            throw new Exception('Слишком короткий запрос');
        }
```

```
            // 2. Удаление опасных символов
            $query  =  preg_replace('/[^\p{L}\p{N}\s\-_]/u',  '',
$query);

            // 3. Ограничение на SQL-инъекции
            $query = $this->preventSQLInjection($query);

            // 4. Ограничение на XSS
            $query = htmlspecialchars($query, ENT_QUOTES, 'UTF-8');

            // 5. Удаление лишних пробелов
            $query = preg_replace('/\s+/', ' ', $query);

            return $query;
        }

        public function search($query, $data) {
            $sanitizedQuery = $this->sanitizeQuery($query);

            // Безопасный поиск в массиве
            $results = [];
            foreach ($data as $item) {
                if (stripos($item, $sanitizedQuery) !== false) {
                    $results[]   =   $this->highlightMatch($item,
$sanitizedQuery);
                }
            }

            return $results;
        }

        private function highlightMatch($text, $query) {
            // Безопасное выделение найденного текста
            $pattern = '/(' . preg_quote($query, '/') . ')/iu';
            $replacement = '<mark>$1</mark>';
            return      preg_replace($pattern,      $replacement,
```

```
htmlspecialchars($text));
            }
        }
```

2) Реализуйте:

– Подсказки при вводе

– Фильтрацию результатов

– Поиск с учетом морфологии

– Логирование поисковых запросов

Лабораторная работа № 13

Принципы разделения кода и организации простого приложения

Цель лабораторной работы: Научиться разделять код на логические модули, выделять общие элементы разметки в отдельные файлы, организовывать простое веб-приложение с единой структурой.

Методические указания

1. Принципы организации кода

1.1. Разделение ответственности (Separation of Concerns):

– **Модель (Model)** — бизнес-логика и работа с данными;

– **Представление (View)** — отображение данных пользователю;

– **Контроллер (Controller)** — обработка пользовательских действий.

1.2. DRY (Don't Repeat Yourself):

```
// Плохо: дублирование кода
function displayHeader() { echo "<header>...</header>"; }
function displayFooter() { echo "<footer>...</footer>"; }

// Хорошо: повторное использование
function includeTemplate($template) { include "templates/$template.php"; }
```

1.3. KISS (Keep It Simple, Stupid):

– Простые решения лучше сложных;

– Минимальное количество абстракций;

– Понятная структура проекта.

2. Шаблонизация

2.1. Подключение общих элементов:

```
// header.php
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title><?= $title ?? 'Мой сайт' ?></title>
```

```
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <header>
        <nav><?php include 'navigation.php'; ?></nav>
    </header>
    <main>
```

2.2. Основной шаблон (layout):

```
// layouts/main.php
<?php include 'header.php'; ?>
<div class="container">
    <?= $content ?>
</div>
<?php include 'footer.php'; ?>
```

3. Работа со списками и записями

3.1. Паттерн "Список-Детали":

```
// Список
/products           -> ProductController::index()
/products/create    -> ProductController::create()
/products/{id}      -> ProductController::show($id)
/products/{id}/edit -> ProductController::edit($id)
```

3.2. Маршрутизация:

```
$routes = [
    '/' => 'HomeController@index',
    '/products' => 'ProductController@index',
    '/products/{id}' => 'ProductController@show',
];
```

4. Организация конфигурации

4.1. Конфигурационные файлы:

```
// config/app.php
```

```
return [
    'name' => 'Мое приложение',
    'url' => 'http://localhost',
    'timezone' => 'Europe/Moscow',
    'debug' => true,
];
```

4.2. Загрузка конфигурации:

```
function config($key, $default = null) {
    static $config = null;

    if ($config === null) {
        $config = require 'config/app.php';
    }

    return $config[$key] ?? $default;
}
```

Задания для выполнения лабораторной работы

Задание 1: Создание шаблонной системы для блога

Создайте простое приложение "Блог" со следующей структурой:

Часть 1: Базовая структура проекта

1) Создайте следующую структуру папок и файлов:

```
text
blog/
├── index.php              (главная страница)
├── config/
│   └── config.php         (конфигурация)
├── includes/
│   ├── header.php         (шапка сайта)
│   ├── footer.php         (подвал сайта)
│   ├── sidebar.php        (боковая панель)
│   └── functions.php      (вспомогательные функции)
├── templates/
│   ├── post-list.php      (шаблон списка статей)
│   ├── post-single.php    (шаблон отдельной статьи)
│   └── pagination.php     (шаблон пагинации)
├── pages/
```

```
│   ├── home.php           (главная страница)
│   ├── category.php       (страница категории)
│   └── post.php           (страница статьи)
└── css/
    └── style.css          (стили)
```

2) Реализуйте файл конфигурации config/config.php, который должен содержать:

- Название блога
- Описание блога
- Настройки подключения к БД (заглушка)
- Пути к основным директориям
- Функцию для получения абсолютных путей

Часть 2: Шаблоны общих элементов

1) В файле includes/header.php создайте шапку сайта, которая должна содержать:

- Логотип с названием блога (берется из конфигурации)
- Главное меню навигации (Главная, Категории, О нас, Контакты)
- Поисковую форму
- Приветствие пользователя (если авторизован)

2) В файле includes/footer.php создайте подвал сайта, который должен содержать:

- Информацию о копирайте
- Дополнительное меню
- Контактную информацию
- Счетчик посещений (простая реализация)

3) В файле includes/sidebar.php создайте боковую панель с:

- Список категорий (минимум 5)
- Последние статьи (5 штук)
- Архив по месяцам
- Облако тегов

Часть 3: Основные шаблоны контента

1) В файле templates/post-list.php создайте шаблон для отображения списка статей:

– Каждая статья должна отображаться в карточке

– Карточка должна содержать: заголовок, краткое описание, дату публикации, автора, категорию

– Добавьте кнопку "Читать далее"

– Реализуйте чередование стилей для четных/нечетных статей

2) В файле templates/post-single.php создайте шаблон для отдельной статьи:

– Полный заголовок статьи

– Автор и дата публикации

– Полный текст статьи

– Список тегов

– Кнопки социальных сетей для шаринга

3) В файле templates/pagination.php создайте компонент пагинации:

– Номера страниц

– Кнопки "Предыдущая" и "Следующая"

– Индикатор текущей страницы

– Эллипсис для большого количества страниц

Часть 4: Страницы приложения

1) В файле pages/home.php реализуйте главную страницу:

– Подключите шапку, подвал и сайдбар

– Выведите последние 10 статей используя шаблон post-list.php

– Добавьте пагинацию внизу

– Включите специальный блок "Избранные статьи"

2) В файле pages/category.php реализуйте страницу категории:

– Получайте ID категории из GET-параметра

– Фильтруйте статьи по категории

– Выводите название категории в заголовке

– Используйте тот же шаблон списка статей

3) В файле pages/post.php реализуйте страницу отдельной статьи:

– Получайте ID статьи из GET-параметра

– Отображайте полный текст статьи

– Показывайте связанные статьи из той же категории

– Добавьте навигацию "предыдущая/следующая статья"

Часть 5: Главный файл приложения

В файле index.php реализуйте роутинг:

- Определяйте запрашиваемую страницу по GET-параметру page
- Поддерживайте следующие маршруты:
  - / или /index.php → home.php
  - /index.php?page=category&id=X → category.php
  - /index.php?page=post&id=X → post.php
- Обрабатывайте несуществующие страницы (404 ошибка)
- Всегда подключайте конфигурацию и функции

Лабораторная работа № 14

Разработка простого контент-ориентированного приложения

Цель лабораторной работы: Интегрировать изученные технологии PHP для создания полноценного контент-ориентированного приложения: реализовать интерфейс для добавления и управления данными, организовать их хранение и обеспечить отображение на клиентской стороне.

Методические указания

1. Контент-ориентированные приложения

1.1. Определение и особенности:

Контент-ориентированное приложение — веб-приложение, основная цель которого — создание, хранение, управление и отображение контента (текстов, изображений, видео и других материалов).

1.2. Типичные компоненты:

- Система управления контентом (CMS);
- Публичная часть для отображения;
- Административная часть для управления;
- База данных/файловое хранилище;
- Пользовательская система.

1.3. Архитектура:

Пользователь → Интерфейс просмотра → Контроллер → Модель данных → Хранилище

Администратор → Интерфейс управления → Контроллер → Модель данных → Хранилище

2. Интеграция технологий PHP

2.1. Используемые технологии:

- PHP для серверной логики;
- HTML/CSS для интерфейса;
- JavaScript для интерактивности;
- Базы данных/файлы для хранения;
- Сессии для аутентификации.

2.2. Паттерны проектирования:

- MVC (Model-View-Controller);
- Репозиторий для доступа к данным;
- Сервисный слой для бизнес-логики.

3. Безопасность контент-приложений

3.1. Основные угрозы:

- XSS через контент;
- SQL-инъекции;
- Незаконное распространение контента;
- Неавторизованный доступ.

3.2. Меры защиты:

- Валидация и санация контента;
- Контроль доступа (RBAC);
- Логирование действий;
- Резервное копирование.

Задания для выполнения лабораторной работы

Задание 1: Структура проекта и главная страница

1) Создайте следующую структуру папок:

```
text
/blog_project/
├── index.php           (главная страница)
├── config/             (конфигурационные файлы)
├── public/             (публичная часть)
│   ├── css/
│   ├── js/
│   ├── images/
│   └── uploads/        (загруженные файлы)
├── admin/              (административная часть)
│   ├── index.php
│   ├── login.php
│   └── dashboard.php
├── includes/           (общие файлы)
│   ├── header.php
│   ├── footer.php
```

```
├── database.php
└── functions.php
└── data/              (хранилище данных)
    ├── posts.json
    ├── users.json
    └── categories.json
```

2) Реализуйте главную страницу index.php, которая должна:

– Отображать последние 10 статей блога

– Иметь навигационное меню

– Содержать список категорий в боковой панели

– Иметь форму поиска по статьям

– Отображать пагинацию если статей больше 10

Задание 2: Система статей блога

1) Разработайте систему хранения статей в JSON-формате. Каждая статья должна содержать:

– ID (уникальный идентификатор)

– Заголовок

– Краткое описание

– Полный текст

– Автора

– Дата публикации

– Категорию

– Теги (массив)

– Изображение-превью

– Статус (опубликована/черновик)

2) Создайте следующие страницы:

– article.php — страница просмотра отдельной статьи

– category.php — страница категории со списком статей

– archive.php — архив статей по месяцам

– search.php — страница результатов поиска

3) Реализуйте функционал комментариев к статьям:

– Форма добавления комментария

– Модерация комментариев (администратором)

– Вложенные комментарии (ответы)

– Капча для защиты от спама

Задание 3: Система аутентификации и авторизации

1) Создайте систему регистрации и входа для администраторов:

– Форма входа с валидацией

– Защита от брутфорс-атак

– Сессионная аутентификация

– Механизм "запомнить меня"

2) Реализуйте систему ролей с двумя уровнями:

– Администратор (полный доступ)

– Редактор (может создавать/редактировать статьи, но не управлять пользователями)

Задание 4: Управление контентом (CRUD для статей)

Разработайте интерфейс для управления статьями в /admin/articles.php:

– Таблица со списком всех статей с возможностью сортировки и фильтрации

– Форма создания новой статьи с WYSIWYG-редактором (используйте простой textarea)

– Форма редактирования существующих статей

– Возможность удаления статей с подтверждением

– Пакетные операции (удаление нескольких статей)

Лабораторная работа № 15

Расширенная работа с HTTP

Цель лабораторной работы: Научиться обрабатывать различные HTTP-методы (GET, POST, PUT, DELETE) и анализировать заголовки запроса в PHP-приложениях.

Методические указания

1. HTTP-методы

HTTP-методы определяют тип операции, которую клиент хочет выполнить с ресурсом. В RESTful API каждый метод имеет четкую семантику:

– **GET** — безопасный и идемпотентный метод для получения данных без изменения состояния сервера.

– **POST** — используется для создания новых ресурсов, не является ни безопасным, ни идемпотентным.

– **PUT** — идемпотентный метод для полного обновления существующего ресурса или создания, если ресурс не существует.

– **DELETE** — идемпотентный метод для удаления ресурса.

– **PATCH** — используется для частичного обновления ресурса.

2. Суперглобальные массивы PHP для работы с HTTP

PHP предоставляет специальные суперглобальные массивы для доступа к данным HTTP-запроса:

– $_GET — содержит параметры, переданные в строке запроса (после знака ? в URL).

– $_POST — содержит данные, отправленные через формы с методом POST или application/x-www-form-urlencoded.

– $_SERVER — содержит информацию о сервере и текущем запросе, включая заголовки.

– $_REQUEST — объединяет данные из $_GET, $_POST и $_COOKIE (не рекомендуется для использования из-за безопасности).

3. Анализ HTTP-заголовков

Заголовки HTTP содержат метаинформацию о запросе или ответе. В PHP заголовки запроса доступны через массив $_SERVER с префиксом HTTP_:

– $_SERVER['REQUEST_METHOD'] — метод HTTP-запроса.

– $_SERVER['HTTP_USER_AGENT'] — информация о браузере

клиента.

– $_SERVER['HTTP_ACCEPT'] — типы контента, которые принимает клиент.

– $_SERVER['HTTP_ACCEPT_LANGUAGE'] — предпочитаемые языки клиента.

Для получения сырых данных запроса (например, для PUT или DELETE) используется file_get_contents('php://input').

4. Отправка HTTP-ответов

PHP позволяет управлять HTTP-ответами через функции:

– header() — отправка HTTP-заголовков ответа. Должна вызываться до любого вывода.

– http_response_code() — установка кода состояния HTTP.

– Коды ответов: 200 (OK), 201 (Created), 400 (Bad Request), 404 (Not Found), 405 (Method Not Allowed).

Задания для выполнения лабораторной работы

Задание 1: Анализатор HTTP-запросов

1) Определите метод запроса ($_SERVER['REQUEST_METHOD']).

2) Выведите все заголовки запроса.

3) В зависимости от метода:

– Для GET: выведите все GET-параметры.

– Для POST: выведите все POST-данные.

– Для PUT/DELETE: выведите сырые данные запроса (file_get_contents('php://input')).

4) Установите соответствующий код ответа (200, 400, 405).

1) Добавьте возможность отправки тестовых запросов через форму.

2) Задание 2: Простой REST-обработчик

3) Реализуйте обработку всех основных HTTP-методов.

4) Для GET — возвращайте список ресурсов.

5) Для POST — создавайте новый ресурс (сохраняйте в сессии или файле).

6) Для PUT — обновляйте ресурс по ID.

7) Для DELETE — удаляйте ресурс по ID.

8) Все ответы возвращайте в формате JSON.

9) Добавьте проверку на наличие обязательных заголовков (например, Content-Type: application/json).

Задание 3: Логирование запросов

1) Записывайте в файл лога информацию о каждом запросе:

– Время запроса.

– IP-адрес клиента.

– Метод запроса.

– URL.

– Заголовки User-Agent и Referer.

2) Реализуйте ротацию логов (например, новый файл каждый день).

3) Создайте страницу для просмотра логов (с фильтрацией по дате и IP).

Лабораторная работа № 16

Рекурсивный обход директорий

Цель лабораторной работы: Освоить работу с файловой системой в PHP, научиться использовать рекурсивные функции для обхода директорий.

Методические указания

1. Функции для работы с файловой системой

PHP предоставляет набор функций для работы с файловой системой:

- scandir() — возвращает массив файлов и папок в указанной директории.
- is_dir() — проверяет, является ли путь директорией.
- is_file() — проверяет, является ли путь файлом.
- is_link() — проверяет, является ли путь символической ссылкой.
- filesize() — возвращает размер файла в байтах.
- filemtime() — возвращает время последнего изменения файла.
- pathinfo() — возвращает информацию о пути к файлу (имя, расширение, директория).

2. Рекурсивный обход директорий

Рекурсивный обход — это процесс посещения всех вложенных папок и файлов в файловой системе. Основные подходы:

- **Рекурсивная функция** — функция, которая вызывает саму себя для обработки вложенных директорий.
- **Итеративный подход** — использование стека или очереди для обхода без рекурсии (предпочтительнее для глубоких структур).
- **Рекурсивный итератор.**

Важные аспекты:

- Учет глубины рекурсии для предотвращения переполнения стека.
- Обработка символических ссылок во избежание бесконечных циклов.
- Форматирование вывода с отступами согласно уровню вложенности.

3. Безопасность при работе с файловой системой

- Всегда проверяйте существование директорий перед обходом.
- Ограничивайте доступ к системным файлам и директориям.
- Экранируйте специальные символы в именах файлов.
- Используйте realpath() для получения абсолютного пути и

предотвращения атак с использованием относительных путей.

Задания для выполнения лабораторной работы

Задание 1: Дерево файловой системы

1) Реализуйте рекурсивную функцию для обхода директорий.

2) Выводите дерево с отступами согласно уровню вложенности.

3) Для каждой записи указывайте:

– Иконку (🗀 для папок, 🗎 для файлов).

– Размер файла (в удобном формате: Б, КБ, МБ).

– Дату последнего изменения.

4) Добавьте возможность скрывать системные файлы (например, начинающиеся с точки).

5) Реализуйте навигацию по дереву (переход в поддиректорий по клику).

Задание 2: Поиск файлов

1) Реализуйте поиск файлов по:

– Имени (полное совпадение, частичное совпадение, регулярное выражение).

– Расширению.

– Размеру (мин./макс.).

– Дате изменения.

2) Отображайте результаты поиска с указанием полного пути.

3) Добавьте возможность сортировки результатов (по имени, размеру, дате).

4) Реализуйте пагинацию для большого количества результатов.

Задание 3: Анализатор использования диска

1) Рассчитайте общий размер директории.

2) Определите самые большие файлы и папки.

3) Визуализируйте использование диска с помощью CSS-графиков.

4) Добавьте предупреждения при достижении заданных лимитов.

Лабораторная работа № 17

Расширенная работа с сессиями

Цель лабораторной работы: Научиться эффективно работать с сессиями в PHP, реализовать сложные механизмы сохранения состояния пользователя.

Методические указания

1. Сессии в PHP

Сессии позволяют сохранять данные между запросами одного пользователя. Основные понятия:

- session_start() — начинает сессию или возобновляет существующую. Должна вызываться до любого вывода.
- $_SESSION — суперглобальный массив для хранения данных сессии.
- session_id() — возвращает идентификатор текущей сессии.
- session_destroy() — уничтожает все данные сессии (но не удаляет cookie на стороне клиента).
- session_unset() — освобождает все переменные сессии.

2. Безопасность сессий

- **Регенерация ID сессии** — session_regenerate_id(true) для предотвращения атак фиксации сессии.
- **Привязка к IP-адресу** — сохранение IP клиента в сессии и проверка при каждом запросе.
- **Ограничение времени жизни** — настройка session.gc_maxlifetime в php.ini.
- **Использование Secure и HttpOnly флагов** для session cookie.
- **Хранение сессий** в базе данных для большей безопасности и масштабируемости.

3. Хранение сложных данных в сессиях

- Сессии автоматически сериализуют и десериализуют данные.
- Можно хранить массивы, объекты (если класс определен до начала сессии).
- Ограничение размера: по умолчанию сессии хранятся в файлах, размер ограничен настройками сервера.
- Производительность: избегайте хранения больших объемов данных в сессии.

Задания для выполнения лабораторной работы

Задание 1: Корзина покупок

1) Реализуйте добавление товаров в корзину (храните в $_SESSION['cart']).

2) Функциональность корзины:

– Добавление/удаление товаров.

– Изменение количества.

– Очистка корзины.

– Расчет общей стоимости.

3) Сохраняйте в корзине:

– ID товара.

– Название.

– Цену.

– Количество.

– Изображение.

4) Добавьте возможность применения скидочных купонов.

5) Реализуйте восстановление корзины при повторном входе пользователя.

Задание 2: Система управления сессиями

1) Отображайте список всех активных сессий.

2) Для каждой сессии показывайте:

– Время начала.

– Последнюю активность.

– IP-адрес.

– Данные сессии (частично).

3) Реализуйте возможность:

– Принудительного завершения сессии.

– Просмотра полных данных сессии.

– Поиска сессий по пользователю или IP.

4) Добавьте автоматическую очистку устаревших сессий.

Лабораторная работа № 18

Работа с JSON

Цель лабораторной работы: Освоить работу с форматом JSON в PHP, научиться создавать системы хранения данных в JSON-файлах, реализовать CRUD-операции над структурированными данными.

Методические указания

1. Формат JSON

JSON (JavaScript Object Notation) — легковесный формат обмена данными, основанный на подмножестве синтаксиса JavaScript. Основные элементы:

– **Объекты** — неупорядоченные наборы пар "ключ-значение" в фигурных скобках {};

– **Массивы** — упорядоченные списки значений в квадратных скобках [];

– **Значения** — строки (в двойных кавычках), числа, true, false, null, объекты или массивы.

2. Функции PHP для работы с JSON

– json_encode() — преобразует PHP-значение в JSON-строку;

```
$json = json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
Параметры форматирования:
JSON_PRETTY_PRINT – красивое форматирование с отступами;
JSON_UNESCAPED_UNICODE – сохранение Unicode-символов;
JSON_UNESCAPED_SLASHES – не экранировать слеши.
```

– json_decode() — преобразует JSON-строку в PHP-значение;

```
$data = json_decode($json, true); // true – вернуть ассоциативный массив
```

Без второго параметра возвращает объект stdClass

– json_last_error() и json_last_error_msg() — получение информации об ошибках декодирования.

3. Работа с JSON-файлами

– Чтение из файла: file_get_contents() + json_decode();

– Запись в файл: json_encode() + file_put_contents();

– Блокировка файлов для предотвращения конкурентного доступа: flock();

– Валидация JSON-структуры перед сохранением.

4. Безопасность при работе с JSON

– Проверка корректности JSON перед обработкой;

– Ограничение размера загружаемых JSON-данных;

– Экранирование специальных символов при выводе;

– Защита от JSON Injection атак.

Задания для выполнения лабораторной работы

Задание 1: Гостевая книга на JSON

1) Реализуйте добавление записей в гостевую книгу с полями:

– Имя пользователя

– Email (с валидацией)

– Сообщение

– Дата и время добавления

– IP-адрес пользователя

2) Реализуйте функции:

– Добавление новой записи

– Просмотр всех записей с пагинацией

– Удаление записей

– Поиск по тексту сообщений

Задание 2: Менеджер задач (Todo List)

1) Создайте структуру данных для задач:

```
{
  "id": 1,
  "title": "Задача",
  "description": "Описание",
  "status": "pending",
  "priority": "medium",
  "created_at": "2024-01-15 10:30:00",
  "deadline": "2024-01-20",
  "tags": ["работа", "срочно"]
}
```

2) Реализуйте CRUD-операции:

– Создание задачи с валидацией полей

– Чтение списка задач с фильтрацией по статусу/приоритету

– Обновление статуса задачи (pending → in_progress → done)
– Удаление выполненных задач

3) Добавьте функции:

– Сортировка задач по приоритету
– Поиск по тегам
– Статистика: количество задач по статусам
– Напоминание о приближающихся дедлайнах

Лабораторная работа № 19

Создание простого шаблонизатора

Цель лабораторной работы: Научиться разделять логику приложения и представление данных, создать собственный шаблонизатор с поддержкой наследования шаблонов.

Методические указания

1. Принцип разделения логики и представления

MVC (Model-View-Controller) архитектура предполагает разделение:

- **Модель (Model)** — бизнес-логика и работа с данными;
- **Представление (View)** — отображение данных, HTML-шаблоны;
- **Контроллер (Controller)** — обработка запросов, координация модели и представления.

Преимущества разделения:

- Упрощение поддержки кода;
- Возможность изменения дизайна без изменения логики;
- Повторное использование компонентов;
- Улучшенная безопасность (экранирование вывода в шаблонах).

2. Основные концепции шаблонизации

- **Переменные** — передача данных из PHP в шаблон;
- **Условия и циклы** — управляющие конструкции в шаблонах;
- **Подключение частей шаблонов** — include, require;
- **Наследование шаблонов** — базовый шаблон + дочерние шаблоны;
- **Функции и фильтры** — обработка данных в шаблонах.

3. Реализация простого шаблонизатора

Базовый шаблонизатор можно создать с помощью:

- extract() — импорт переменных из массива в текущую область видимости;
- ob_start() / ob_get_clean() — буферизация вывода;
- Регулярные выражения для парсинга специальных конструкций.

4. Безопасность в шаблонах

- Автоматическое экранирование вывода (htmlspecialchars);
- Ограничение выполняемого кода в шаблонах;
- Валидация данных перед выводом;

– Защита от XSS-атак.

Задания для выполнения лабораторной работы

Задание 1: Простой шаблонизатор

1) Реализуйте базовый класс шаблонизатора с методами:

```
class TemplateEngine {
  private $data = [];
  private $templateDir = 'templates/';

  public function assign($key, $value);
  public function render($template);
  public function display($template);
}
```

2) Добавьте поддержку:
   - Переменных: {{ $variable }}
   - Условных операторов: {% if condition %} ... {% endif %}
   - Циклов: {% foreach array as item %} ... {% endforeach %}
   - Подключения шаблонов: {% include 'header.tpl' %}
3) Реализуйте автоматическое экранирование HTML-сущностей
4) Добавьте кэширование скомпилированных шаблонов

Задание 2: Наследование шаблонов

Создайте систему наследования шаблонов:

1) Реализуйте синтаксис для наследования:

```
<!-- base.tpl -->
<html>
<head><title>{% block title %}Сайт{% endblock %}</title></head>
<body>
  {% block content %}{% endblock %}
</body>
</html>

<!-- page.tpl -->
{% extends 'base.tpl' %}
{% block title %}Заголовок страницы{% endblock %}
{% block content %}Содержимое страницы{% endblock %}
```

2) Добавьте поддержку:

- Множественного наследования
- Вложенных блоков
- Блоков по умолчанию
- Вызова родительского блока: {{ parent() }}

3) Создайте иерархию шаблонов:

- base.tpl — базовый шаблон
- layout.tpl — шаблон с навигацией
- page.tpl, article.tpl, product.tpl — специализированные шаблоны

Лабораторная работа № 20

Реализация простого роутера

Цель лабораторной работы: Научиться создавать систему маршрутизации (роутинг) для PHP-приложений, реализовать паттерн "Фронт-контроллер".

Методические указания

1. Паттерн "Фронт-контроллер"

Фронт-контроллер — это единая точка входа в приложение, которая обрабатывает все запросы и делегирует выполнение соответствующим обработчикам.

Преимущества:

- Централизованная обработка запросов;
- Единая точка для аутентификации, логирования, безопасности;
- Упрощение поддержки и расширения;
- Чистые URL (человекопонятные пути).

2. Маршрутизация (роутинг)

Роутинг — процесс сопоставления URL-адреса с соответствующим обработчиком (контроллером и действием).

Типы маршрутов:

- **Статические маршруты** — фиксированные пути;
- **Динамические маршруты** — с параметрами: /user/{id};
- **Регулярные выражения** — гибкое сопоставление;
- **Маршруты по HTTP-методам** — разные обработчики для GET/POST.

3. Структура роутера

Базовый роутер должен содержать:

- Систему хранения маршрутов;
- Метод для добавления маршрутов;
- Метод для поиска подходящего маршрута;
- Парсинг параметров из URL;
- Обработку ошибок (404, 405).

4. ЧПУ (Человекопонятные URL)

Преобразование URL вида index.php?page=user&action=show&id=5 в /user/5/show

Настройка веб-сервера для поддержки ЧПУ:

- **Apache**: файл .htaccess с mod_rewrite;
- **Nginx**: конфигурация в server block;

– **Встроенный сервер PHP**: маршрутизация через index.php.

Задания для выполнения лабораторной работы

Задание 1: Базовый роутер

1) Реализуйте класс роутера:

```
class Router {
  private $routes = [];

  public function add($method, $path, $handler);
  public function match($method, $path);
  public function dispatch($method, $path);
}
```

2) Поддержите типы маршрутов:

– Статические: '/about'

– С параметрами: '/user/{id}'

– С необязательными параметрами: '/blog/{slug?}'

– С регулярными выражениями: '/post/{id:\d+}'

3) Реализуйте парсинг параметров:

```
// Для маршрута '/user/{id}/edit/{action}'
// и URL '/user/42/edit/update'
// Вернуть: ['id' => '42', 'action' => 'update']
```

4) Добавьте поддержку middleware (промежуточного ПО)

Задание 2: RESTful роутер

Создайте RESTful-ориентированный роутер:

1) Реализуйте ресурс-ориентированную маршрутизацию:

```
$router->resource('users', 'UserController');
// Создает маршруты:
// GET    /users           → UserController::index
// GET    /users/create    → UserController::create
// POST   /users           → UserController::store
// GET    /users/{id}      → UserController::show
// GET    /users/{id}/edit → UserController::edit
// PUT    /users/{id}      → UserController::update
// DELETE /users/{id}      → UserController::destroy
```

2) Добавьте поддержку вложенных ресурсов:

```
$router->resource('users.posts', 'PostController');
// GET /users/5/posts
```

3) Реализуйте группировку маршрутов:

```
$router->group('/admin', function($router) {
  $router->get('/dashboard', 'AdminController@dashboard');
  $router->resource('/users', 'AdminUserController');
})->middleware('auth');
```

4) Создайте систему именованных маршрутов для генерации URL

Лабораторная работа № 21

Обработка ошибок и исключений

Цель лабораторной работы: Освоить современные методы обработки ошибок в PHP, научиться использовать механизм исключений (exceptions), создавать пользовательские исключения для валидации данных и построения надежных приложений.

Методические указания

1. Типы ошибок в PHP

– **Fatal Errors** — критические ошибки, останавливающие выполнение скрипта;

– **Warnings** — предупреждения, не останавливающие выполнение;

– **Notices** — уведомления о потенциальных проблемах;

– **Exceptions** — исключения, которые можно перехватывать и обрабатывать.

2. Механизм исключений (Exceptions)

Исключения позволяют отделить нормальный поток выполнения от обработки ошибок. Основные компоненты:

– **throw** — генерация исключения;

– **try** — блок кода, в котором могут возникнуть исключения;

– **catch** — блок для перехвата и обработки исключений;

– **finally** — блок, выполняемый всегда (в PHP 5.5+).

3. Пользовательские исключения

Создание собственных классов исключений для лучшей семантики:

```
class ValidationException extends Exception {
    private $errors = [];

    public function __construct($message, $errors = [], $code = 0) {
        parent::__construct($message, $code);
        $this->errors = $errors;
    }
    public function getErrors() {
        return $this->errors;
    }
}
```

4. Глобальная обработка ошибок

– set_error_handler() — установка пользовательского обработчика ошибок;

– set_exception_handler() — установка глобального обработчика исключений;

– register_shutdown_function() — регистрация функции для выполнения при завершении скрипта.

Задания для выполнения лабораторной работы

Задание 1: Базовая обработка исключений

1) Реализуйте систему валидации пользовательских данных с исключениями:

```
class UserValidator {
    public function validate($data) {
        $errors = [];

        if (empty($data['username'])) {
            throw new ValidationException('Имя пользователя обязательно');
        }

        if (!filter_var($data['email'], FILTER_VALIDATE_EMAIL)) {
            throw new ValidationException('Некорректный email');
        }

        return true;
    }
}
```

2) Создайте иерархию пользовательских исключений:

– AppException (базовое)

– ValidationException (для ошибок валидации)

– DatabaseException (для ошибок БД)

– AuthException (для ошибок аутентификации)

3) Реализуйте глобальный обработчик исключений с логированием в файл

Задание 2: Расширенная система валидации

Создайте комплексную систему валидации с цепочками исключений:

1) Реализуйте валидатор с поддержкой правил:

```
$validator = new Validator();
$validator->validate([
    'username'    =>    ['required',    'min:3',    'max:20',
'alphanumeric'],
    'email' => ['required', 'email', 'unique:users'],
    'password' => ['required', 'min:8', 'confirmed']
], $data);
```

2) Создайте систему вложенных исключений:

```
try {
    // Основная логика
} catch (ValidationException $e) {
    // Обработка валидации
    throw new AppException('Ошибка обработки данных', 0, $e);
}
```

3) Реализуйте механизм перевода сообщений об ошибках:

```
class TranslatedException extends Exception {
    public function getTranslatedMessage($lang = 'ru') {
        // Возвращает переведенное сообщение
    }
}
```

4) Создайте middleware для автоматической обработки исключений в веб-приложении

Лабораторная работа № 22

Защита от CSRF-атак

Цель лабораторной работы: Изучить механизм CSRF-атак, научиться реализовывать защиту от них с помощью CSRF-токенов, интегрировать систему защиты во все формы приложения.

Методические указания

1. Что такое CSRF-атака?

CSRF (Cross-Site Request Forgery) — атака на веб-приложения, при которой злоумышленник заставляет авторизованного пользователя выполнить нежелательные действия без его ведома. Атака возможна, когда:

- Пользователь авторизован на целевом сайте;
- Злоумышленник знает структуру запросов;
- Не используется дополнительная проверка запросов.

Пример атаки: пользователь посещает вредоносный сайт, который отправляет скрытый запрос на банковский сайт для перевода денег.

2. Принцип защиты CSRF-токенами

CSRF-токен — уникальное случайное значение, которое:

- Генерируется сервером для каждой сессии пользователя;
- Передается клиенту (обычно в скрытом поле формы или meta-теге);
- Отправляется обратно на сервер при каждом запросе, изменяющем состояние;
- Проверяется сервером на соответствие сохраненному значению.

3. Реализация CSRF-защиты

Ключевые компоненты:

- **Генерация токена:** использование криптографически безопасных генераторов
- **Хранение токена:** в сессии пользователя или в cookie (Double Submit Cookie)
- **Валидация токена:** сравнение отправленного токена с сохраненным
- **Срок жизни токена:** ограничение времени действия токена

4. Дополнительные меры защиты

- Проверка заголовка Origin/Referer;
- Использование SameSite cookies;
- Ограничение времени жизни сессии;
- Подтверждение действий для критических операций.

Задания для выполнения лабораторной работы

Задание 1: Базовая защита CSRF

1) Реализуйте класс для работы с CSRF-токенами:

```
class CSRFProtection {
    private $tokenName = 'csrf_token';
    private $tokenLength = 32;

    public function generateToken();
    public function getToken();
    public function validateToken($token);
    public function getTokenField(); // возвращает HTML скрытого поля
}
```

2) Реализуйте методы:

– Генерация токена с помощью random_bytes() или openssl_random_pseudo_bytes()

– Хранение токена в сессии с временной меткой

– Проверка токена (совпадение + время жизни)

3) Создайте защищенную форму с автоматической вставкой токена

4) Реализуйте middleware для автоматической проверки POST-запросов

Задание 2: Расширенная система защиты

Создайте улучшенную систему CSRF-защиты:

1) Реализуйте поддержку нескольких токенов одновременно (для нескольких вкладок)

2) Добавьте привязку токена к конкретному действию:

```
$csrf->generateToken('user_delete');
$csrf->validateToken('user_delete', $_POST['token']);
```

3) Создайте систему одноразовых токенов (токен удаляется после использования)

4) Реализуйте защиту AJAX-запросов через заголовки:

```
// JavaScript
fetch('/api/data', {
    method: 'POST',
```

```
    headers: {
        'X-CSRF-Token': csrfToken
    }
});
```

5) Добавьте мониторинг попыток CSRF-атак с логированием

Лабораторная работа № 23

Отправка email

Цель лабораторной работы: Научиться отправлять электронные письма из PHP-приложений, обрабатывать вложения, реализовать систему обратной связи с защитой от спама.

Методические указания

1. Методы отправки email в PHP

- **Функция mail()** — встроенная функция PHP (ограниченные возможности)
- **PHPMailer** — популярная библиотека с расширенными функциями
- **Swift Mailer** — современная библиотека для работы с почтой
- **API почтовых сервисов** — SendGrid, Mailgun, Amazon SES

2. Структура email-сообщения

- Заголовки (Headers): From, To, Subject, Content-Type, MIME-Version
- Тело сообщения: текстовая и HTML-версии
- Вложения (Attachments): файлы, встроенные изображения
- Альтернативные версии: plain text + HTML

3. Безопасность при отправке email

- Валидация и фильтрация входящих данных
- Защита от header injection атак
- Ограничение размера вложений
- Проверка типов файлов
- Капча или другие методы защиты от спама

4. Форматы и кодировки

- MIME-типы для разных форматов вложений
- Кодировка Base64 для бинарных данных
- Кодировка quoted-printable для текста
- Правильная установка кодировки символов (UTF-8)

Задания для выполнения лабораторной работы

Задание 1: Базовая форма обратной связи

1) Реализуйте форму обратной связи с полями:

- Имя (обязательное)
- Email (с валидацией)

– Тема сообщения

– Текст сообщения (с минимальной/максимальной длиной)

– Загрузка файла (до 5 МБ, ограниченные типы)

2) Используйте PHPMailer для отправки:

```
require 'PHPMailer/PHPMailer.php';

$mail = new PHPMailer\PHPMailer\PHPMailer();
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'your@gmail.com';
$mail->Password = 'password';
$mail->SMTPSecure = 'tls';
$mail->Port = 587;
```

3) Добавьте защиту от спама:

– Капча

– Ограничение по времени между отправками

– Проверка на одинаковое содержимое

Задание 2: Расширенная система отправки

Создайте систему массовой отправки email:

1) Реализуйте класс для работы с почтой:

```
class MailService {
    private $config;
    private $queue = [];

    public function send($to, $subject, $body, $attachments =
[]);
    public function addToQueue($email);
    public function processQueue();
    public function getDeliveryReport();
}
```

2) Добавьте поддержку:

– Шаблонов писем (Twig или простые replace)

- Персонализации (Dear {name},)
- Встроенных изображений
- Отслеживания открытий писем

Лабораторная работа № 24

Создание простого REST API

Цель лабораторной работы: Научиться проектировать и реализовывать RESTful API на PHP, работать с различными HTTP-методами, форматами данных и аутентификацией.

Методические указания

1. Принципы REST

REST (Representational State Transfer) — архитектурный стиль для распределенных систем. Основные принципы:

- Единообразие интерфейса: стандартные HTTP-методы
- Отсутствие состояния: каждый запрос содержит всю информацию
- Кэширование: явное указание возможности кэширования
- Слоистая система: клиент не знает о внутренней структуре
- Код по требованию (опционально): клиент может получать исполняемый код

2. RESTful API дизайн

- Ресурсы: сущности (пользователи, статьи, товары)
- Коллекции: множества ресурсов (/articles)
- Элементы: конкретные ресурсы (/articles/42)
- HTTP-методы:

3. Форматы данных

- основной формат для REST API
- для legacy-систем
- для загрузки файлов
- Content Negotiation: заголовки Accept и Content-Type

4. Аутентификация и авторизация

- логин/пароль в заголовке
- JWT или аналоги
- OAuth 2.0 — делегирование доступа
- API Keys — простые ключи доступа

Задания для выполнения лабораторной работы

Задание 1: Простое API для статей

1) Реализуйте CRUD операции для статей:

```
// GET /api/articles - список статей
```

```
// GET /api/articles/{id} - конкретная статья
// POST /api/articles - создание статьи
// PUT /api/articles/{id} - обновление статьи
// DELETE /api/articles/{id} - удаление статьи
```

2) Поддержите:

– Пагинацию: GET /api/articles?page=2&limit=10

– Фильтрацию: GET /api/articles?category=php

– Сортировку: GET /api/articles?sort=-created_at

– Поиск: GET /api/articles?q=php+8

3) Все ответы в формате JSON с соответствующими HTTP-кодами

4) Добавьте валидацию входных данных

Задание 2: API с аутентификацией

Расширьте API, добавив систему аутентификации:

1) Реализуйте endpoints для аутентификации:

```
// POST /api/auth/register - регистрация
// POST /api/auth/login - вход (возвращает токен)
// POST /api/auth/logout - выход
// POST /api/auth/refresh - обновление токена
```

2) Используйте JWT (JSON Web Tokens) для аутентификации:

– Генерация токена при успешном входе

– Проверка токена в middleware

– Обновление токена по истечении срока

3) Добавьте ролевую модель:

– Анонимные пользователи: только чтение

– Авторизованные: чтение + создание своих статей

– Администраторы: полный доступ

4) Реализуйте rate limiting для защиты от перегрузки

Лабораторная работа № 25

Оптимизация и отладка

Цель лабораторной работы: Освоить методы отладки и оптимизации PHP-кода, научиться анализировать производительность приложений, выявлять и устранять узкие места, рефакторить код для улучшения его структуры и качества.

Методические указания

1. Инструменты отладки

- **Xdebug** — мощный отладчик для PHP с профилированием
- **PHP Debug Bar** — панель отладки для веб-приложений
- **Chrome DevTools** — инструменты разработчика браузера
- **Логирование** — кастомные системы логирования

2. Профилирование кода

- Анализ времени выполнения функций;
- Определение узких мест (bottlenecks);
- Оптимизация запросов к базе данных;
- Анализ использования памяти.

3. Рефакторинг кода

Принципы улучшения кода:

- **DRY** (Don't Repeat Yourself)
- **KISS** (Keep It Simple, Stupid)
- **SOLID** — принципы ООП

4. Оптимизация производительности

- Кэширование данных и результатов;
- Оптимизация запросов к БД;
- Использование opcache;
- Минификация ресурсов.

Задания для выполнения лабораторной работы

Задание 1: Система мониторинга производительности

Создайте систему для мониторинга производительности приложения:

1) Реализуйте сбор метрик:

```
class PerformanceMonitor {
    public function startMeasurement($name);
    public function endMeasurement($name);
    public function getReport();
```

}

2) Добавьте отслеживание:

– Время выполнения запросов

– Использование памяти

– Количество SQL-запросов

– Загрузка CPU

3) Создайте дашборд для визуализации метрик

4) Реализуйте систему алертов при превышении пороговых значений

5) Добавьте рекомендации по оптимизации на основе собранных данных

Лабораторная работа № 26

Итоговая работа: разработка модуля с использованием MVC

Цель лабораторной работы: Применить на практике все полученные знания, создать полноценное веб-приложение с использованием архитектуры MVC, реализовать все необходимые компоненты от базы данных до пользовательского интерфейса.

Методические указания

1. Архитектура MVC

MVC (Model-View-Controller) — архитектурный паттерн, разделяющий приложение на три компонента:

- **Model (Модель)** — бизнес-логика и работа с данными;
- **View (Представление)** — отображение данных пользователю;
- **Controller (Контроллер)** — обработка пользовательского ввода, координация модели и представления.

2. Компоненты MVC-приложения

- **Роутер** — маршрутизация запросов к контроллерам;
- **Контроллеры** — обработка запросов, вызов моделей;
- **Модели** — работа с БД, бизнес-логика;
- **Представления** — шаблоны для отображения;
- **Сервисы** — вспомогательные классы.

3. Структура проекта

```
text
app/
├── Controllers/
├── Models/
├── Views/
├── Services/
├── Middleware/
└── core/
public/
├── index.php (фронт-контроллер)
├── css/
├── js/
└── uploads/
```

Задания для выполнения лабораторной работы

Задание 1: Проектирование приложения

Создайте техническое задание для выбранного приложения (блог или каталог):

1) Определите функциональные требования:

– Для блога: статьи, категории, комментарии, теги

– Для каталога: товары, категории, корзина, заказы

2) Спроектируйте базу данных (ER-диаграмма)

3) Определите список необходимых страниц и их функционал

4) Создайте макеты интерфейса (можно схематично)

5) Составьте план реализации по этапам

Задание 2: Реализация ядра приложения
Создайте базовую структуру MVC-фреймворка:
1) Реализуйте роутер с поддержкой:

– Именованных маршрутов

– Middleware

– Группировки маршрутов

2) Создайте базовый класс контроллера:

```
abstract class Controller {
    protected function render($view, $data = []);
    protected function redirect($url);
    protected function json($data);
}
```

3) Реализуйте систему работы с базой данных:

– Подключение через PDO

– Query Builder или простой ORM

– Миграции базы данных

4) Создайте шаблонизатор с наследованием

Задание 3: Разработка основного функционала
Реализуйте выбранное приложение:
1) Для блога:

– CRUD для статей с WYSIWYG редактором

– Система категорий и тегов

– Комментарии с древовидной структурой

– Поиск по статьям

– RSS-лента

- Административная панель

2) Для каталога:
   - Каталог товаров с фильтрацией
   - Корзина покупок
   - Оформление заказа
   - Личный кабинет пользователя
   - Отзывы и рейтинги товаров
   - Административная панель для управления товарами
3) Общие компоненты для любого варианта:
   - Регистрация и аутентификация пользователей
   - Восстановление пароля
   - Загрузка изображений
   - Пагинация
   - Форма обратной связи
   - SEO-оптимизация (человекопонятные URL, мета-теги)

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. – 5-е изд. – Санкт-Петербург : Питер, 2019. – 816 с. – (Серия «Бестеллеры O'Reily»).

2 **Роббинс, Дж.** HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс ; [пер. с англ. М. А. Райтман]. – 4-е изд. – Москва : Эксмо, 2014. – 528 с.

3 Учебники, задачники, справочники по web языкам [Электронный ресурс]. – Форма доступа: http://code.mu/.

4 **Крокфорд, Д.** JavaScript: сильные стороны / Д. Крокфорд. – Санкт-Петербург : Питер, 2012. – 176 с.

5 **Бенедетти, Р.** Изучаем работу с jQuery / Р. Бенедетти, Р. Крэнли. – Санкт-Петербург : Питер, 2012. – 528 с.

6 **Полуэктова, Н. Р.** Разработка веб-приложений: учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — Москва: Издательство Юрайт, 2022. — 204 с. (образовательная платформа Юрайт https://urait.ru/)

7 **Колисниченко Д.Н.** Разработка веб-приложений. – Спб.: БХВ-Петербург, 2017. – 640 с. [Электронный ресурс]. Форма доступа: https://books.google.ru/books?id=BjExDwAAQBAJ&printsec=frontcover&hl=ru#v=onepage&q &f=false

8 **Ломаш, Д. А.** Интернет-технологии и мультимедиа : учебное пособие / Д. А. Ломаш, О. Г. Ведерникова ; ФГБОУ ВО РГУПС. – Ростов-на-Дону, 2017. – 119 с.

9 **Бибо, Бер.** jQuery. Подробное руководство по продвинутому JavaScript / Бер Бибо, Иегуда Кац ; [пер. с англ.]. – 2-е изд. – Санкт-Петербург : Символ-Плюс, 2011. – 624 с.

10 **Васильев, А. Н.** JavaScript в примерах и задачах / А. Н. Васильев. – Москва : Издательство «Э», 2017. – 720 с. – (Российский компьютерный бестселлер).

Учебное издание

БЭКЕНД-РАЗРАБОТКА
(СЕРВЕРНАЯ ЧАСТЬ)

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Адрес университета:
344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка
Народного Ополчения, д. 2, www.rgups.ru