Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №8 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ
на тему
«Архитектура *MVC*»

Выполнил ст. гр. 820601 А.Р. Шведов
Проверил преп. каф. ИТАС А.Л. Гончаревич

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

PHP — язык программирования, который наиболее распространён в сфере веб-разработки. Язык *PHP* работает на удаленном сервере, поэтому он и называется серверный язык программирования.

Любой скрипт *PHP* состоит из последовательности операторов. Оператор может быть присваиванием, вызовом функции, циклом, условным выражением или пустым выражением. Операторы обычно заканчиваются точкой с запятой. Также операторы могут быть объединены в группу заключением группы операторов в фигурные скобки. Группа операторов также является оператором.

Интернет — это множество компьютеров по всему миру, соединённых между собой проводами в единую сеть. Все компьютеры делятся на две большие группы: клиенты и сервера. Клиенты инициируют запросы на сервера, а те, в свою очередь, их принимают, обрабатывают и отправляют клиенту ответ.

PHP позволяет решить множество задач связанных с клиент-серверной архитектурой, например:

1 С помощью *HTML* можно только создать форму. А обработать то, что ввёл пользователь, может лишь *PHP*.

2 Если делать блог на на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. PHP позволяет обойтись с помощью одного файла, а статьи хранить в базе данных. Благодаря этому, можно сделать админку, из которой можно будет добавлять и редактировать контент.

3 *PHP* позволяет реализовать механизм авторизации на сайте.

1 ПОСТАНОВКА ЗАДАЧИ

Изучить семантику, синтаксис и возможности языка *PHP*. Изучение базового синтаксиса в языке *PHP*, работа с БД: чтение, запись, добавление информации и полей, создание таблиц. Ознакомление с основными концепциями модели архитектуры *MVC*.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Базы данных и СУБД

БД – база данных. Под этим термином понимается информация, которую вы храните.

СУБД – система управления базой данных. Это программа, которая предоставляет доступ внешним приложениям к базе данных, обеспечивает ее работу.

Существуют различные популярные СУБД: *Oracle, Microsoft SQL Server, MySQL, Sybase, PostgreSQL* итд.

SQL (*Structured Query Language*) – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

2.2 *Docker*

*Docker (*Докер*)* — программное обеспечение с открытым исходным кодом, применяемое для разработки, тестирования, доставки и запуска веб-приложений в средах с поддержкой контейнеризации. Он нужен для более эффективного использование системы и ресурсов, быстрого развертывания готовых программных продуктов, а также для их масштабирования и переноса в другие среды с гарантированным сохранением стабильной работы.

В данной работе СУБД *MySQL* открывается именно как докер-контейнер, пример команды запуска:

```
docker run --name mysql \
   -e MYSQL_ROOT_PASSWORD=$MYSQL_PASSWORD \
   -e MYSQL_DATABASE="lab8" \
   -p 3306:3306 \
   -d mysql:latest
```

2.3 Архитектура *MVC*

Архитектура *MVC* является самым распространённым шаблоном проектирования веб-приложений. Её суть заключается в разделении всех компонент системы на три группы: контроллеры, модели и представления (или вьюшки, от англ. *view*). Отсюда и аббревиатура: *Model-View-Controller*.

Архитектура *MVC* может принимать различный вид в зависимости от языка программирования и решаемой задачи.

Основные компоненты архитектуры составляют:

1 Контроллер принимает запрос, анализирует, чего вообще хочет пользователь и управляет логикой работы всей программы.

2 Модель – библиотека с функциями. Она изолирована от внешнего мира, т.е., сама не копается в массивах *GET* и *POST* и, что очень важно, ничего не выводит на экран. Задача модели – просто принять от контроллера команду и вернуть результат.

3 Представление – *HTML*-код с небольшими вставками *PHP*.

В сложных системах очень важно разделять основную логику работы и решение мелких попутных задач. Модель становится набором решений таких задач, и, благодаря этому, в контроллере остаются только те действия, которые определяют логику работы страницы. *HTML*-код вынесен в представления также с целью разгрузки контроллера от кода.

2.4 Используемые в работе функции, методы и объекты

В работе используются следующие функции:

1 mysql_connect($server, $username, $password) — устанавливает соединение с сервером *MySQL*. Следующие значения по умолчанию установлены для отсутствующих параметров: *server = 'localhost:3306'*, *username* = имя пользователя владельца процесса сервера и *password* = пустой пароль.

2 *mysql_select_db ($db_name)* — выбирает для работы указанную базу данных на сервере, на который ссылается переданный указатель. Если параметр указателя опущен, используется последнее открытое соединение. Если нет ни одного открытого соединения, функция попытается соединиться с сервером аналогично функции mysql_*connect*(), вызванной без параметров.

3 *mysql_query()* — посылает запрос активной базе данных сервера, на который ссылается переданный указатель. Если параметр *link_identifier* опущен, используется последнее открытое соединение. Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции *mysql_connect*() без параметров. Результат запроса буфферизируется.

*4 finfo_*open() — функция, открывающая файл в как «магическую базу данных» и возвращая ее экземпляр.

5 finfo_close() — функция, которая закрывает экземпляр *finfo*.

6 move_uploaded_file($from, $to) — проверяет, является ли файл *from* загруженным на сервер (переданным по протоколу *HTTP POST*). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе *to*.

7 getimagesize($file) — определит размер любого заданного, поддерживаемого изображения и вернёт этот размер вместе с типом файла и текстовой строкой *height/width*, которую можно будет использовать внутри тега *HTML IMG*, а также вернёт соответствующий тип содержимого *HTTP*.

8 imagecreatetruecolor() — возвращает объект, представляющий чёрное изображение заданного размера.

9 imagecopyresampled() — копирует прямоугольную часть одного изображения на другое изображение, интерполируя значения пикселов таким образом, чтобы уменьшение размера изображения не уменьшало его чёткости.

10 *require()* — включает и выполняет указанный файл. В случае возникновения ошибки он также выдаст фатальную ошибку уровня *E_COMPILE_ERROR*. Другими словами, он остановит выполнение скрипта, тогда как *include()* только выдал бы предупреждение *E_WARNING*, которое позволило бы скрипту продолжить выполнение.

3 ХОД РАБОТЫ

3.1 Постановка задачи

Работа выполняется с помощью редактора кода – *Visual Studio Code*. Работа с программой начинается с создания *PHP* файла, в который будут помещаться скрипты.

Согласно заданию необходимо создать галерею фотографий. Она должна состоять всего из одной странички, на которой пользователь видит все картинки в уменьшенном виде и форму для загрузки нового изображения. При клике на фотографию она должна открыться в браузере в новой вкладке.

При загрузке изображения необходимо делать проверку на тип и размер файла. ри загрузке изображения на сервер должна создаваться его уменьшенная копия. А на странице *index.php* должны выводиться именно копии. На реальных сайтах это активно используется для экономии трафика. При клике на уменьшенное изображение в браузере в новой вкладке должен открываться оригинал изображения на странице *photo.php*.

Приведём структуру проекта на рисунке 3.1.

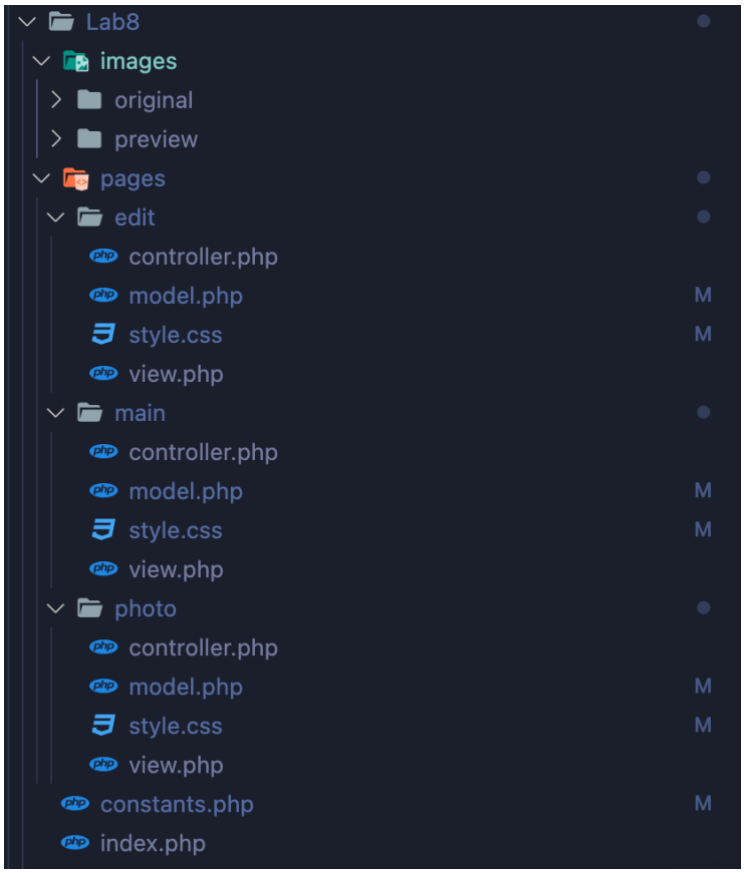

Рисунок 3.1 — Структура проекта

3.2 Техническое выполнение

Приведём основной фрагмент кода, выполняющий работу с СУБД и загрузку копий картинок локально. Заметим, что сохраненные в директории *preview/* изображения уже правильного размера.

```
<?php
function upload_file($file, $connection) {
    $fileName = basename($file['name']);
    $uploadFilePath = ORIGINAL_DIR . $fileName;
    $cropFilePath = PREVIEW_DIR . $fileName;
    if (!str_contains($file['type'], 'image')) {
        echo 'Данный тип файла не поддерживается! ';
        return;
    }
    if ($file['size'] > 5E+6) {
        echo 'Размер файла на должен превышать 5 мегабайт!';
        return;
    }
    if (copy($file['tmp_name'], $uploadFilePath)) {
        resize($uploadFilePath, $cropFilePath, 150,150);
        echo "Файл корректен и был успешно загружен.\n";
        mysqli_query($connection, "INSERT INTO lab7.Gallery (name, clicks) VALUES ('".$fileName."', 0)");
    } else {
        echo 'Ошибка загрузки файла';
    }
}
if (isset($_FILES['userfile'])) {
    upload_file($_FILES['userfile'], $connection);
}
$result = mysqli_query($connection, "SELECT * FROM lab7.Gallery ORDER BY clicks DESC", MYSQLI_USE_RESULT);
$imagesInfoArray = mysqli_fetch_all($result, MYSQLI_BOTH);
?>
```

Приведём фрагмент кода, который отвечает за соединение с БД:

```
$connection = mysqli_connect("localhost:3306", "root", "password", "lab7");
```

Приведём фрагмент кода, который отвечает за отображение страницы *photo.php:*

```
<img class="original" src='images/original/<?php echo $imageName ?>' alt='<?php echo $imageName ?>' />
<div class="info">
    <h3>Количество нажатий: <?php echo $imageClicks + 1 ?></h3>
    <a href="./index.php">Назад</a>
</div>
```

Фрагмент кода, отвечающий за обновление метаданных изображений:

```
<?php
    require $_SERVER['DOCUMENT_ROOT'] . '/Lab8' . '/constants.php';
    function fetchImageInfo($connection, $name) {
        $result = mysqli_query($connection, "SELECT * FROM Gallery WHERE name = '$name'", MYSQLI_USE_RESULT);
        $fetchedInfo = mysqli_fetch_all($result, MYSQLI_BOTH);

        return $fetchedInfo[0];
    }
    function updateImageTitle($connection, $name, $title) {
        mysqli_query($connection,
        "UPDATE Gallery
        SET title = '$title'
        WHERE name = '$name'");
    }
    function updateImageAlt($connection, $name, $alt) {
        mysqli_query($connection,
        "UPDATE Gallery
        SET alt = '$alt'
        WHERE name = '$name'");
    }
```

На рисунке 3.2 приведен скриншот базы данных.

MySQL 8.0.28 : TLSv1.2 : local test : lab8 : Gallery

name	clicks	title	alt
cube-19...	2	image	image
tiger.jpg	0	changed	changed

Рисунок 3.2 — Содержание БД

3.3 Руководство пользователя

Приведём описание для пользователя. Сайт состоит из основной страницы *index.php*. Видно, что изображения отсортированы по количеству кликов, также им заданы базовые значения *Title: image и Alt: image*. Данная страница предоставлена на рисунке 3.3.

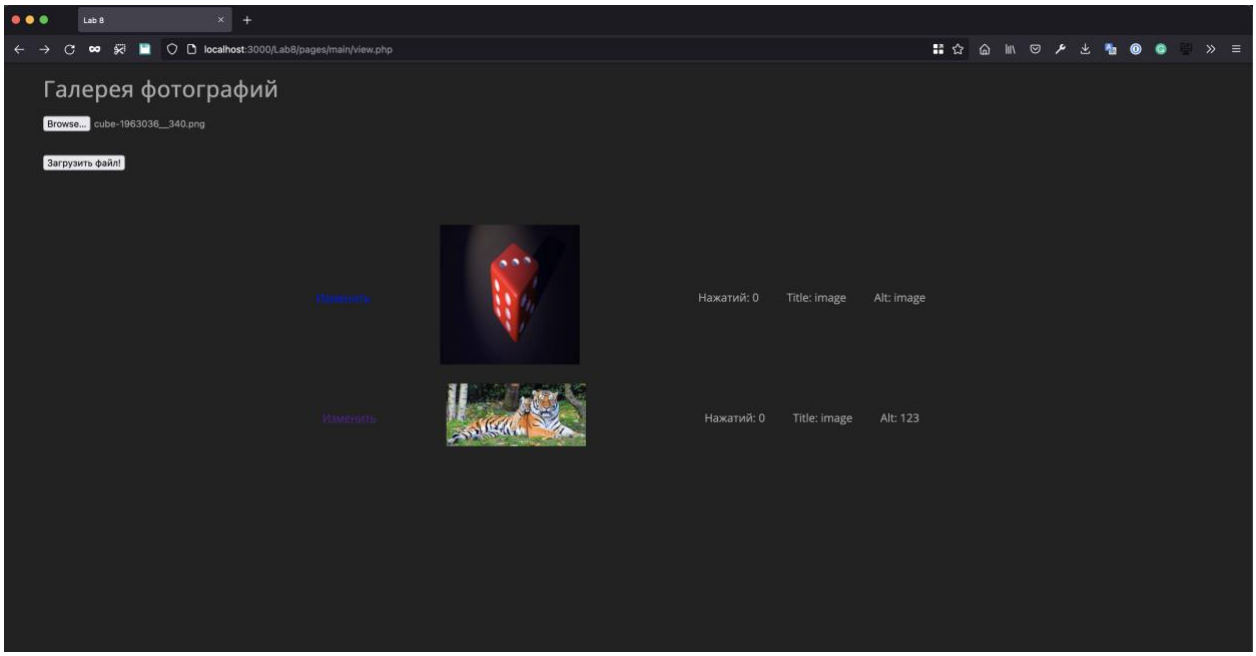

Рисунок 3.3 — Страница *index.html*

При нажатии на изображение пользователь перенаправляется на страницу *photo.php*, откуда он может вернуться назад в галерею. Пример этого приведен на рисунке 3.4.

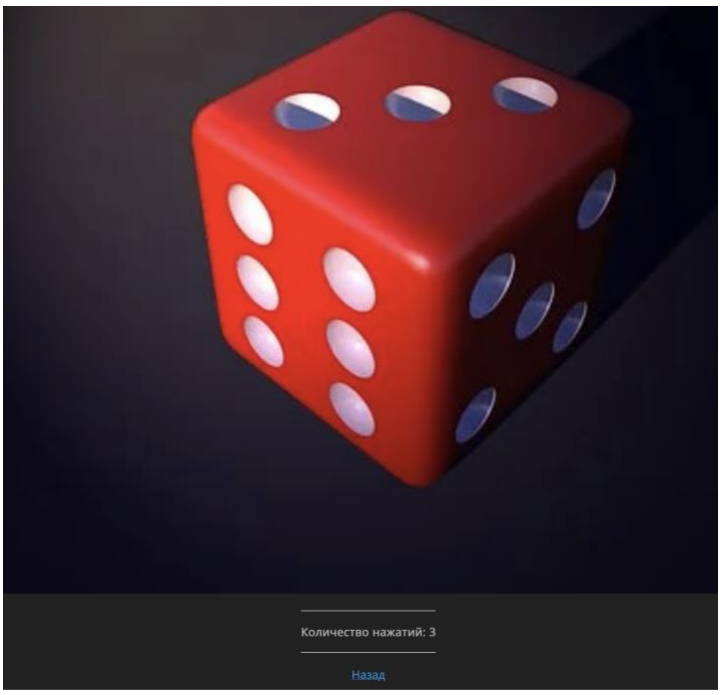

Рисунок 3.4 — Страница *photo.php*

На данной странице пользователь видит уменьшенные копии загруженных им изображений. Кнопка «*upload image*» не может быть нажата без загрузки файла. Так же при загрузке файла в браузере файлов будут подсвечиваться (если ОС и приложение браузера это позволяют) только файлы того расширения, которое поддерживает форма загрузки. Пример этого приведен на рисунке 3.5.

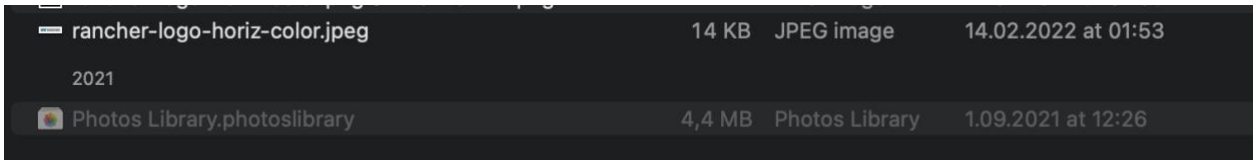

Рисунок 3.5 — Подсветка файлов поддерживаемого расширения

Если пользователь загрузит файл, который нельзя будет обработать или загрузить, выведется ошибка, пример которой приведен на рисунке 3.6.

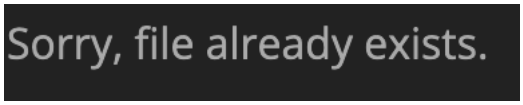

Рисунок 3.6 — Уведомление о неуспешной загрузке

При нажатии на кнопку «изменить» пользователь перенаправляется на страницу *edit*, что показано на рисунке 3.7. После изменения атрибутов пользователь перенаправится на домашнюю страницу.

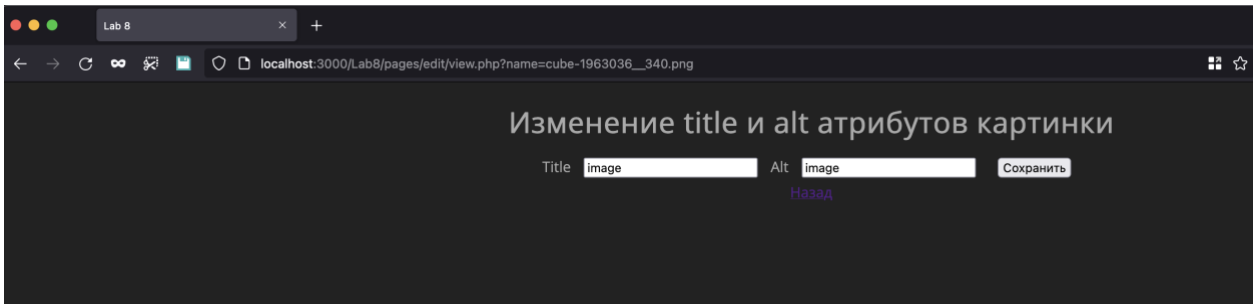

Рисунок 3.7 — Изменение атрибутов изображения

Пример изменения атрибутов приведен приведен на рисунке 3.8. Видно, что при открытии информации об изображении в *debugger* браузера, атрибуты картинки «Тигр» — *changed*, *changed*.

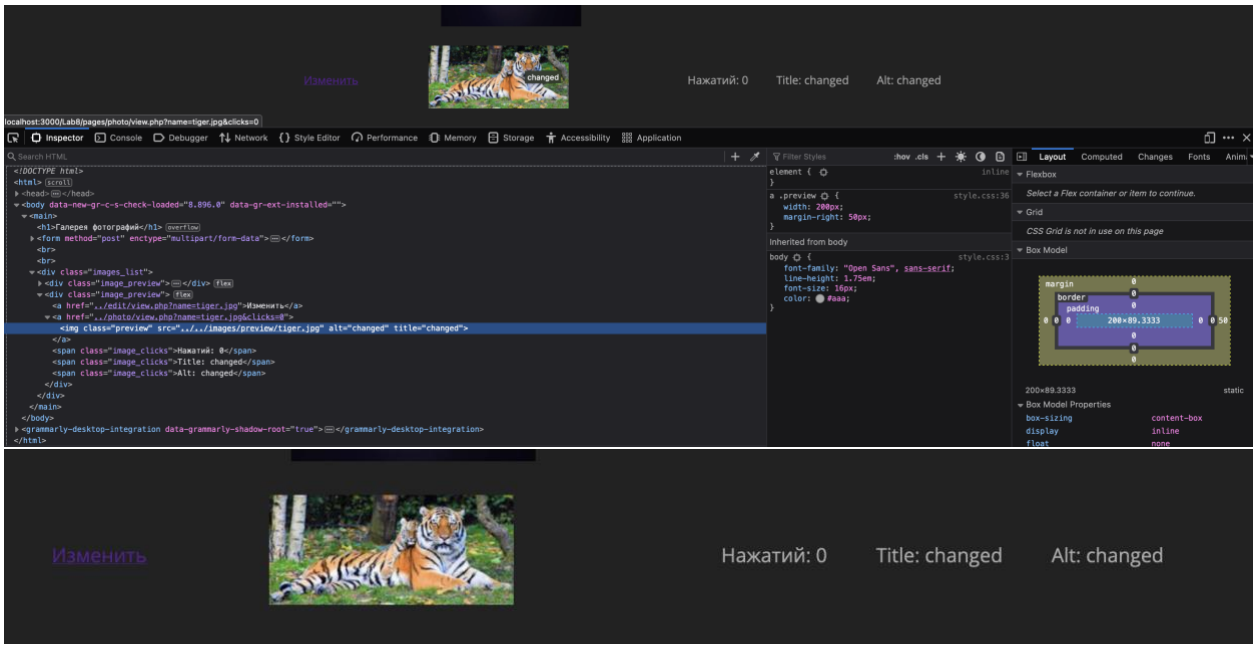

Рисунок 3.8 — Изменённые атрибуты изображения

ЗАКЛЮЧЕНИЕ

HTTP — лёгкий в использовании расширяемый протокол. Структура клиент-сервера, вместе со способностью к простому добавлению заголовков, позволяет *HTTP* протоколу продвигаться вместе с расширяющимися возможностями Сети.

PHP позволяет создавать качественные *web*-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

*Я*зык *PHP* постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков *web*-программирования, по крайней мере, в ближайшее время.

В ходе выполнения лабораторной работы я изучил использование функций работы с СУБД *MySQL* в *PHP*, разработал пример приложения с загрузкой и чтением изображений из локальной директории с сохранением данных об этих изображениях в БД, изучил основы архитектуры *MVC*.