**Практическая работа №7.**

**Разработка приложения в laravel.**

**Цель работы**: Разработка web - приложения на примере laravel.

**Задачи:**

1. Изучить функции, позволяющие работать с базой данных MySQL средствами РНР.

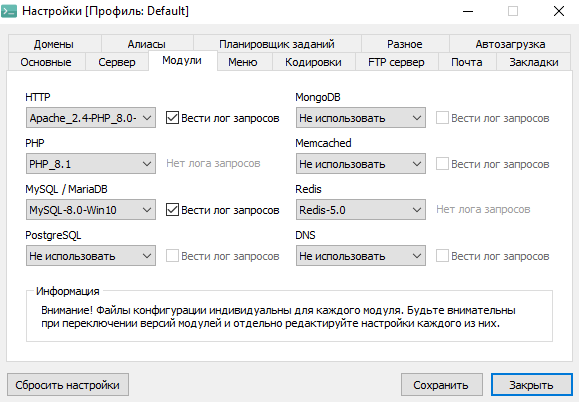
2. Формирование навыков создания и разработки приложения на примере laravel.

3. Изучить возможности и функционал laravel для разработки web – приложений.

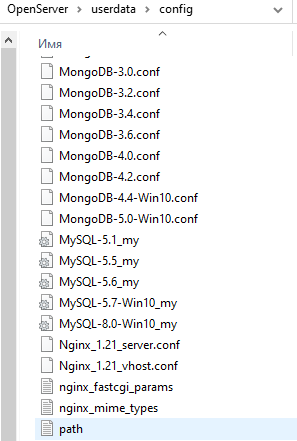
**Порядок работы:**

**Установка и работа с laravel:**

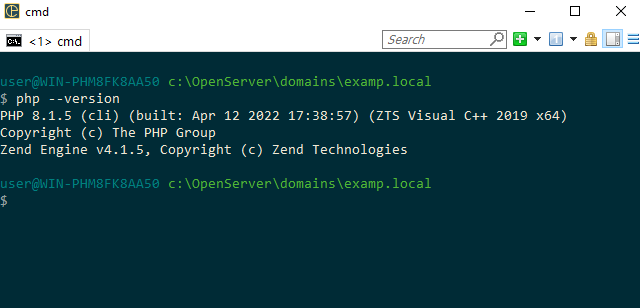
1. Установить Open Server

2. Проверить версии php, MySql, Redis 

3. Дополнительно установить nodejs и для совместимости с OpenServer создать текстовый файл path.txt в котором указать путь: C:\Program Files\nodejs\



4. Проверить актуальность всех версий через командную строку OpenServer (результат проверки представить в виде скриншота через командную строку)

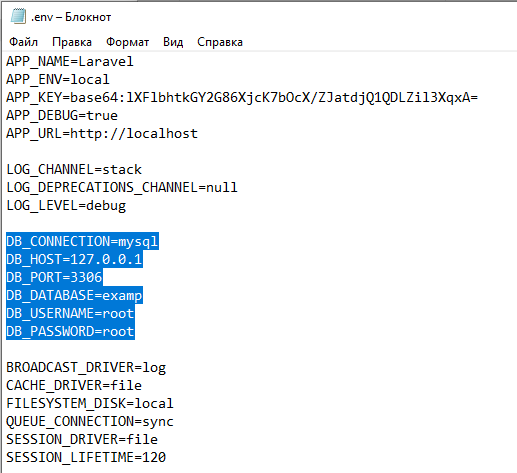


5. Создать проект на laravel

команда создания проекта:

*composer create-project laravel/laravel <название>*

6. Для создания БД проекта и корректной работы, необходимо проверить в файле .env пароль и название БД:



Выполнить команду миграции БД:

*php artisan migration*

Все выполненные действия отразить в отчете в виде команд и скриншотов.

**Требования к приложению:**

В соответствии с разработанной в практической работе №5 картой сайта разработать приложение. Учесть: разграничение прав пользователей, БД должна быть в 3НФ (логическая – физическая модели), все заявленные функции должны быть реализованы.

Предусмотреть:

- описание дизайна разрабатываемого приложения

- авторизацию/регистрацию пользователей (через БД)

- проверку на корректность вводимых данных как в БД так и на формах сайта

- наличие (обязательных/необязательных) полей при заполнении форм

- наличие в листинге комментариев по блокам.

**Этапы разработки:**

- проектирование и разработка базы данных (с учетом установки laravel п.1-6) – 2ч.

- дизайн, макеты страниц – 2ч.

- реализация функций приложения в соответствии с разработанной в работе №5 картой сайта и описанным функционалом с учетом разграничения прав доступа пользователей – 4ч.

- оформление всех этапов реализации приложения (в виде презентации) – 2ч

- защита проектов – 4ч.

**Теоретический материал:**

Язык PHP позволяет работать практически со всеми известными СУБД. Для этого необходимо знать способ подключения к этой СУБД и стандартные функции PHP для работы. Рассмотрим работу в PHP с СУБД MySQL.

MySQL - быстрый многопоточный, многопользовательский надежный SQL-сервер баз данных. (SQL - язык структурированных запросов). Сервер MySQL предназначен для разнообразных систем - от маленьких сайтов до крупных интернет-проектов. Сейчас это самый распространенный сервер баз данных в сети Интернет. MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой, есть возможность объединять при выполнении запроса данные из нескольких таблиц. Для работы с базой данных требуется задать в скрипте следующие настройки:

• имя хоста (сервера баз данных): обычно localhost, но на реальном хостинге возможно и другое значение, указанное провайдером;

• логин пользователя БД: на локальном хосте обычно root, на удаленном сервере - логин, полученный от провайдера;

• пароль пользователя БД: на локальном хосте обычно пуст или также root, на удаленном сервере - пароль, полученный от провайдера;

• имя базы данных: в нашем случае, задано строкой my;

• имя таблицы или общий префикс всех таблиц, если БД достаточно сложна: в нашем случае, соответствует имени таблицы messages, содержащей сообщения.

Обычный порядок работы с этими данными таков:

• подключиться к базе данных стандартной функцией mysqli\_connect, передав ей параметрами имя хоста, логин и пароль;

• выбрать нужную таблицу, передав ее имя параметром функции mysqli\_select\_db;

• сформировать строку SQL-запроса, записав ее, например, в переменную $sql;

• передать эту строку параметром функции mysqli\_query, сохранив результат, возвращенный функцией, например, в переменной $result;

• если переменная $result пуста, запрос не выполнен, в противном случае узнать количество записей базы, возвращенных запросом, можно стандартной функцией mysqli\_num\_rows ($result);

• в цикле поочередно извлечь и обработать записи одной из двух функций: mysqli\_fetch\_assoc ($result), если запись запроса нужно вернуть как ассоциативный массив, или mysqli\_fetch\_array ($result), если достаточно массива с числовыми индексами. Например, при выполнении запроса select \* from messages первая функция будет возвращать записи сообщений в виде элементов массива с индексами id, title и text, а вторая - с индексами 0, 1 и 2. Основой для управления механизмами авторизации служат сессии. Сессии позволяют создавать и использовать переменные, сохраняющие свое значение в течение всего времени работы пользователя с сайтом. При этом у каждого пользователя сайта данные переменные будут собственными, т.е. их область видимости (англ. variable scope) распространяется на все время нахождения на сайте конкретного пользователя, причем для каждого захода пользователя на ваш сайт эти переменные будут различными.

В основе всего механизма сессий лежит решение задачи об идентификации того, от кого именно пришел запрос на сервер. Если это будет точно известно, то уже не возникнет большой проблемы в том, чтобы предоставить скрипту информацию, относящуюся именно к этому конкретному пользователю.

Данная задача решается путем присвоения каждой сессии уникального идентификатора SID (англ. Session IDentifier), который создается в тот момент, когда пользователь заходит на сайт, и уничтожается, когда пользователь уходит с сайта. Он представляет собой строку из 32 символов, например, ac4f4a45bdc893434c95dcaffb1c1811. SID передается на сервер вместе с каждым запросом клиента и возвращается обратно вместе с ответом сервера. Алгоритм генерации SID позволяет гарантировать его уникальность, поэтому исключена возможность того, что две сессии будут иметь один и тот же идентификатор сессии. PHP может использовать два различных механизма в качестве транспортного средства для передачи SID:

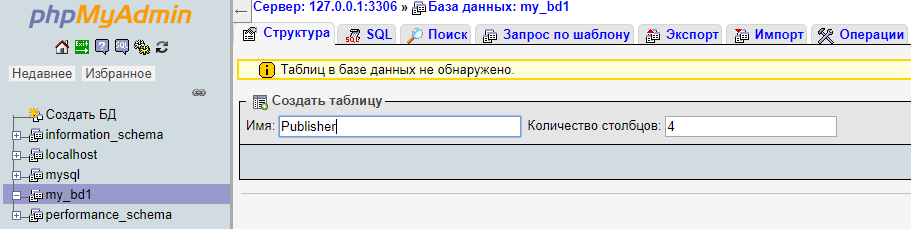
• файлы cookies;

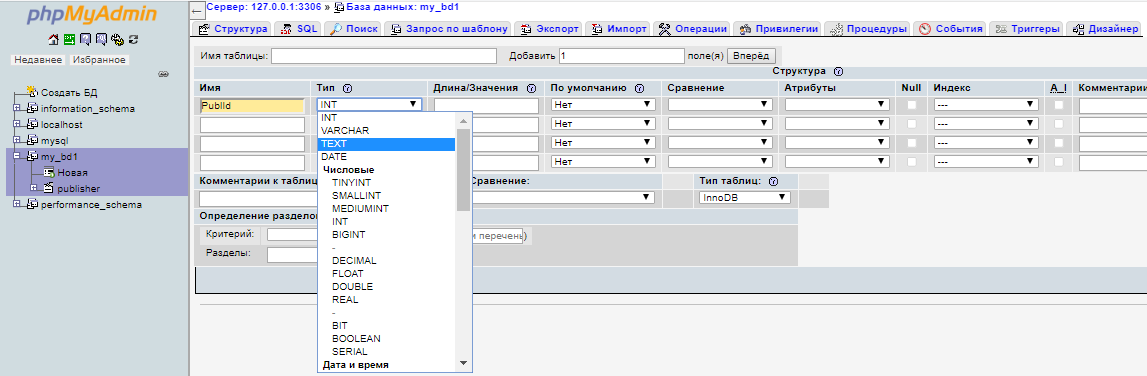
• дополнительный параметр URL-адреса.

Cookies (от англ. cookie - печенье) - это небольшие текстовые файлы с зашифрованной информацией, отправляемые веб-сервером и хранимые на компьютере пользователя в назначенной браузером для этой цели системной папке. Браузер всякий раз при попытке открыть страницу соответствующего сайта пересылает его cookie серверу в составе HTTP-запроса, при этом SID сохраняется "внутри" браузера и остается незаметным для пользователя. Поддержка cookies - необязательное условие для браузера, она может отсутствовать или быть отключена.

Все cookies, как минимум, имеют имя и значение, а отправить их программно можно функцией bool setcookie (string $name [, string $value]), имеющей еще несколько необязательных параметров. Прочитать имеющиеся для данной страницы cookies можно как элементы ассоциативного массива $\_COOKIE.

Функции для удаления cookie не существует, это происходит либо по истечении срока ее хранения, либо по явному вызову setcookie с указанием третьим параметром времени хранения, которое уже истекло: setcookie ('my\_cookie','',time()-14\*24\*3600); Следует также учесть, что вначале сервер направляет cookie клиенту как часть отклика HTTP, потом клиент, если он готов принять cookie, возвращает ее серверу.





**Содержание отчета:**

1. Титульный лист.

2. Цель работы, задание.

3. Схема БД.

4. Исходный текст программы на PHP **к заданию 6.**

5. Скриншот сгенерированной страницы.

6. Выводы.