

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО
ITMO University**

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 3

По дисциплине Web-программирование

Тема работы Создание сайта по отправлению обратной связи

Обучающийся Гаврилов Кирилл Станиславович

Факультет Факультет инфокоммуникационных технологий

Группа K3322

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	<u>10.12.2024</u> (дата)	<u> </u> (подпись)	<u>Гаврилов К. С.</u> (Ф.И.О.)
Руководитель	<u> </u> (дата)	<u> </u> (подпись)	<u>Марченко Е.В.</u> (Ф.И.О.)

Санкт-Петербург
2024 г.

Цель

Создать сайт для отправки обратной связи, используя php-скрипт и инструментарий для отладки проектов. Ознакомиться с параллельными и последовательными задачами Gulp.

Задачи

1. Ознакомиться с параллельными и последовательными задачами Gulp;
2. Написать форму обратной связи с помощью PHP-скриптов;
3. Установить WordPress и настроить переход на сайт по ссылке test.site.

Ход работы

Задание 1.

В данном задании нужно было создать два таска и настроить их выполнение параллельным и последовательным способами.

Для выполнения задания были созданы две простые задачи Hello и GoodBye, выводящие соответствующие сообщения в консоль. Потом были созданы задачи для выполнения вышеупомянутых задач разными способами, используя `series()` и `parallel()`.

```
var gulp : Gulp | {...} | {...} = require('gulp');
const browserSync = require('browser-sync').create();

gulp.task('Hello', function(done) : void {
  console.log("Hello, world!");
  done();
});
gulp.task('GoodBye', function(done) : void {
  console.log("GoodBye, world!");
  done();
});

gulp.task('order', gulp.series('Hello', 'GoodBye'));
gulp.task('parallel', gulp.parallel('Hello', 'GoodBye'));
```

Рисунок 1 – Параллельное и последовательное исполнение задач

Вызвав в консоли задачи можно увидеть, что в случае «order» задачи исполняются последовательно, а в «parallel» - параллельно.

```
kswboks@MacBook-Air-Kirill-3 task1 % gulp order
[22:51:32] Using gulpfile ~/Desktop/пары/web/lab3/task1/gulpfile.js
[22:51:32] Starting 'order'...
[22:51:32] Starting 'Hello'...
Hello, world!
[22:51:32] Finished 'Hello' after 433 μs
[22:51:32] Starting 'GoodBye'...
GoodBye, world!
[22:51:32] Finished 'GoodBye' after 300 μs
```

Рисунок 2 – Последовательное исполнение задач

```

[22:52:19] Using gulpfile ~/Desktop/напы/web/lab3/task1/gulpfile.js
[22:52:19] Starting 'parallel'...
[22:52:19] Starting 'Hello'...
[22:52:19] Starting 'GoodBye'...
Hello, world!
[22:52:19] Finished 'Hello' after 813 μs
GoodBye, world!
[22:52:19] Finished 'GoodBye' after 876 μs
[22:52:19] Finished 'parallel' after 2.11 ms

```

Рисунок 3 – Параллельное выполнение задач

Далее был написан gulpfile, при выполнении которого в браузере отображаются файлы проекта и происходит обновление при каждом изменении.

Для этого изначально были созданы html и css файлы для запускаемой страницы, которые были помещены в папку source_project. Была определена структура путей для разных задач в объекте path: build, src, watch, clean. Импортированы требуемые библиотеки и определены функции: browserSync – для запуска локального сервера, html – для обработки html-файлов и дальнейшего обновления содержимого в браузере, watchfiles – для отслеживания изменений в html-файлах и запуска функции html. Последующие строки отвечают за запуск задач.

```

let project_folder:string = "dist";
let source_folder:string = "src";

let path:{...} = {
  build: {
    html: project_folder + "/",
    css: project_folder + "/css/",
    js: project_folder + "/js/",
    img: project_folder + "/img/"
  },
  src: {
    html: [source_folder + "/*.html", "! " + source_folder + "/*.html"],
    css: source_folder + "/scss/style.scss",
    js: source_folder + "/js/script.js",
    img: source_folder + "/img/**/*.{jpg, png, svg, gif, ico, webp}"
  },
  watch: {
    html: source_folder + "/*.html",
    css: source_folder + "/scss/**/*.scss",
    js: source_folder + "/js/**/*.js",
    img: source_folder + "/img/**/*.{jpg, png, svg, gif, ico, webp}"
  },
  clean: "." + project_folder + '/'
}

let {src, dest} = require('gulp'),
    gulp = require('gulp'),
    browsersync = require("browser-sync").create(),
    fileinclude : function(any): any | {...} = require("gulp-file-include");

```

Рисунок 4 - Gulpfile для отображения файлов проекта

```

Show usages
function browserSync(params) : void {
  browsersync.init({
    server:{
      baseDir: "." + project_folder + '/'
    },
    port:3000,
    notify:false
  })
}

Show usages
function html() {
  return src(path.src.html)
    .pipe(fileinclude())
    .pipe(dest(path.build.html))
    .pipe(browsersync.stream())
}

Show usages
function watchFiles(params) : void {
  gulp.watch([path.watch.html], html);
}

let build = gulp.series(html);
let watch = gulp.parallel(build, watchFiles, browserSync);

exports.html = html;
exports.build = build;
exports.watch = watch;
exports.default = watch;

```

Рисунок 5 - Gulpfile для отображения файлов проекта

При выполнении задачи открывается страница. При внесении изменений в файлы страница автоматически обновляется.

Шапка сайта

тестовый html для gulp

Рисунок 6 – Открывающаяся страница в браузере

```
-----  
[Browsersync] Serving files from: ./dist/  
[23:08:35] Starting 'html'...  
[Browsersync] 1 file changed (index.html)  
[23:08:35] Finished 'html' after 9.35 ms  
[Browsersync] Reloading Browsers...  
□
```

Рисунок 7 – Логи консоли при изменении файла

Задание 2.

В этом задании было необходимо реализовать форму обратной связи, которую можно было бы отправить и получить информацию с помощью php-скрипта.

Для этого был написан файл `feedback.html`, который содержит в себе форму для заполнения пользователем. В ней пользователю предлагается ввести свои ФИО и почту. Также пользователю предлагается написать развернутую обратную связь в `textarea`, выбрать оценку товара с помощью `radio-input` и указать, откуда он узнал про платформу с помощью `checkbox-input`. В конце была добавлена кнопка для отправления обратной связи соответственно.

```

<!DOCTYPE html>

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Обратная связь</title>
</head>
<body>
  <h1>Ваша обратная связь</h1>
  <form action="feedback.php" method="post">
    <p>Ваше имя:</p>
    <input type="text" name="name">
    <p>Ваша фамилия:</p>
    <input type="text" name="surname">
    <p>Ваш email:</p>
    <input type="email" name="email">
    <p>Ваш комментарий:</p>
    <textarea name="feedback" cols="50" rows="20" placeholder="Введите вашу обратную связь..."></textarea>

    <p>Ваша оценка товара:</p>
    <p> <input type="radio" name="score" value="0"> 0 </p>
    <p> <input type="radio" name="score" value="1"> 1 </p>
    <p> <input type="radio" name="score" value="2"> 2 </p>
    <p> <input type="radio" name="score" value="3"> 3 </p>

    <p>Где вы видели информацию о нашей компании?</p>
    <input type="checkbox" name="service[]" value="www">Интернет<br>
    <input type="checkbox" name="service[]" value="city">Реклама в городе<br>
    <input type="checkbox" name="service[]" value="friend">Посоветовали друзья<br>
    <br>
    <input type="submit" value="Отправить отзыв">
  </form>
</body>

```

Рисунок 8 – HTML-код формы обратной связи

Далее необходимо было написать PHP-скрипт, который обрабатывает post- и get-запросы. В post-запросе сначала проверяется, что все данные были отправлены и после создаются все переменные, которые были получены после отправления формы. Поле со множественным выбором сначала проверяется на пустоту, после чего идет подсчет выбранных аспектов. После этого происходит запись в txt-файл и открывается новая страница с кнопкой для просмотра всех ответов пользователей (get-запрос).

```

function PostRequest() {
    global $file;

    // Проверяем, что данные были отправлены
    if (isset($_POST['name'], $_POST['surname'], $_POST['email'], $_POST['feedback'], $_POST['score'])) {

        $name = $_POST['name'];
        $surname = $_POST['surname'];
        $email = $_POST['email'];
        $feedback = $_POST['feedback'];
        $score = $_POST['score'];
        $services = isset($_POST['service']) ? implode(separator: ' ', $_POST['service']) : 'Не указано';

        $data = "Имя: $name $surname\nEmail: $email\nОценка: $score\nКомментарий: $feedback\nГде нашли нас: $services\n\n";

        file_put_contents($file, $data, flags: FILE_APPEND );

        echo "<h3>Спасибо за ваш отзыв!</h3>";
        echo "<br>";
        echo "<form action='feedback.php' method='get'>";
        echo "<input type='submit' value='Посмотреть ответы'>";
        echo "</form>";
    } else {
        echo "<h3>Пожалуйста, заполните все поля формы.</h3>";
    }
}

```

Рисунок 9 – Код PHP-скрипта

Get-запрос, как было сказано ранее, вызывается при нажатии на кнопку «Посмотреть ответы». В данном случае файлы считываются из txt-файла и выводятся на странице с помощью тэга <pre>.

```

// Функция для обработки GET
1 usage
function GetRequest() {
    global $file;

    if (file_exists($file)) {
        // Читаем все отзывы из файла
        $feedbacks = file_get_contents($file);
        echo "<h3>Ваши отзывы:</h3>";
        echo "<pre>$feedbacks</pre>";
    } else {
        echo "<h3>Пока нет отзывов.</h3>";
    }
}

// Обработка запросов
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    PostRequest();
} else {
    GetRequest();
}
?>

```

Рисунок 10 - Код PHP-скрипта

Результат выполнения упражнения показан на рисунках.

Ваша обратная связь

Ваше имя:

Ваша фамилия:

Ваш email:

Ваш комментарий:

Введите вашу обратную связь...

Ваша оценка товара:

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3

Где вы видели информацию о нашей компании?

- ☐ Интернет
- ☐ Реклама в городе
- ☐ Посоветовали друзья

Отправить отзыв

Рисунок 11 – Страница сбора обратной связи

Спасибо за ваш отзыв!

Посмотреть ответы

Рисунок 12 – Страница после отправления ответов

Ваши отзывы:

Имя: Кирилл
Email: kravnik.03@gmail.com
Оценка: 0
Комментарий: thhth
Где нашли нас: Не указано

Имя: fgf fdgdfgf
Email: dsds@mail.ru
Оценка: 1
Комментарий: vdvvd
Где нашли нас: www

Рисунок 13 – Страница с просмотром ответов пользователя

Задание 3.

Для выполнения данного задания нужно установить WordPress, для его установки будем использовать MAMP – локальный серверный инструмент, который позволяет запускать веб-приложения на MacOS.

Скачиваем с официального сайта MAMP и производим установку.

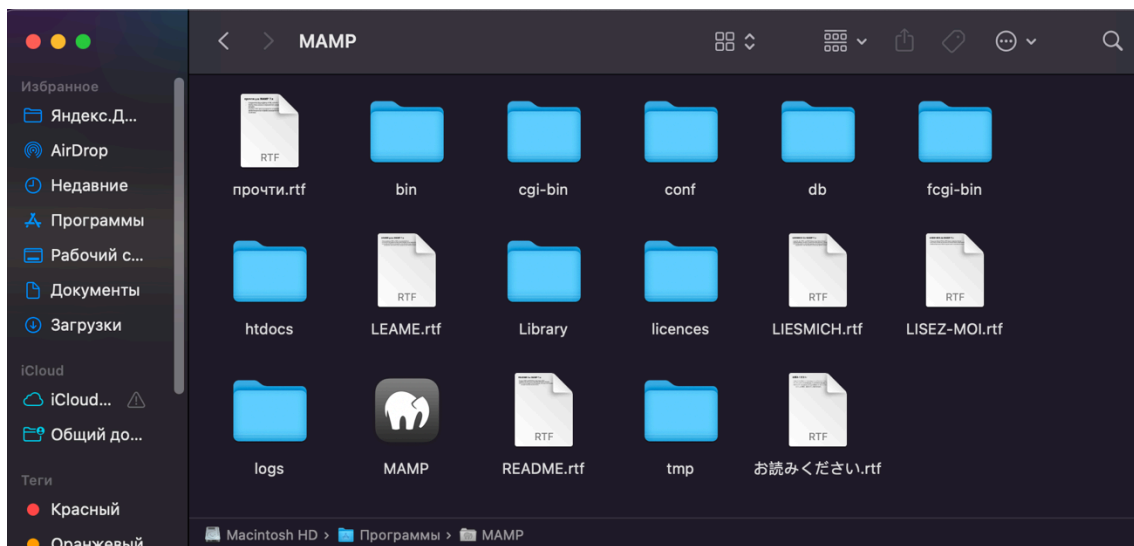


Рисунок 14 – Папка установленного инструментария

Далее скачиваем движок wordpress и перемещаем соответствующую папку в папку htdocs.

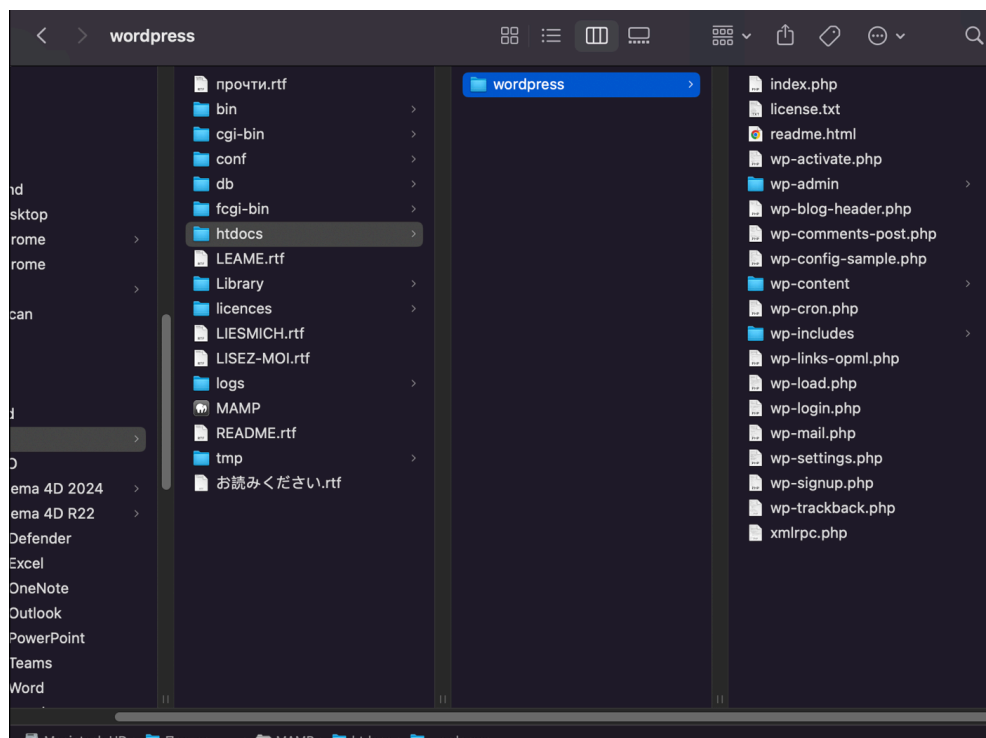


Рисунок 15 – Распаковка папки wordpress

Произведем запуск сервера и создадим базу данных MySQL.

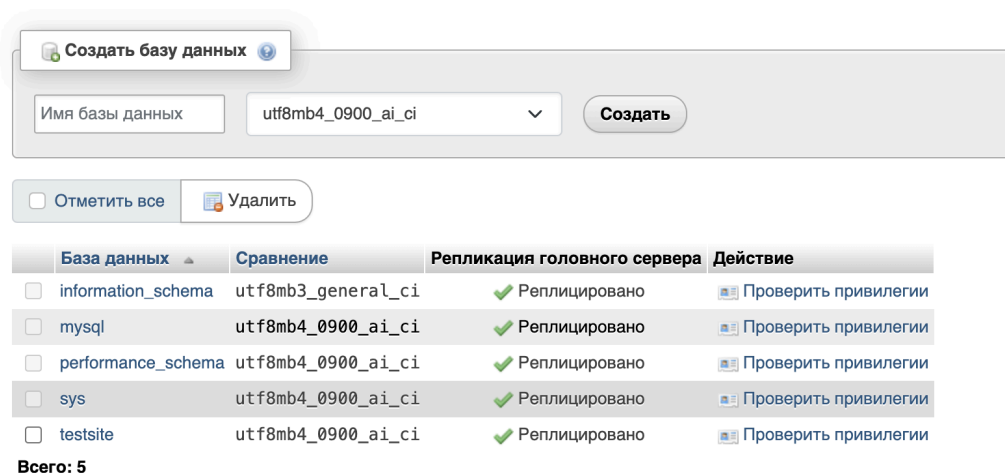



Рисунок 16 – Создание базы данных

Произведем настройку wordpress, перейдя по ссылке <http://localhost/wordpress>.



Введите здесь информацию о подключении к базе данных. Если вы в ней не уверены, свяжитесь с поддержкой вашего хостинга.

Имя базы данных
Имя базы данных, в которую вы хотите установить WordPress.

Имя пользователя
Имя пользователя базы данных.

Пароль [Показать](#)
Пароль пользователя базы данных.

Сервер базы данных
Если localhost не работает, нужно узнать правильный адрес в службе поддержки хостинг-провайдера.

Префикс таблиц
Если вы хотите запустить несколько копий WordPress в одной базе, измените это значение.

[Отправить](#)

Рисунок 17 – Настройка wordpress

После успешной установки мы попадем в консоль разработчика только что созданного сайта.

Тестовый сайт

Пример страницы

Блог

Привет, мир!

Добро пожаловать в WordPress. Это ваша первая запись. Отредактируйте или удалите ее, затем начинайте создавать!

17.12.2024

Тестовый сайт

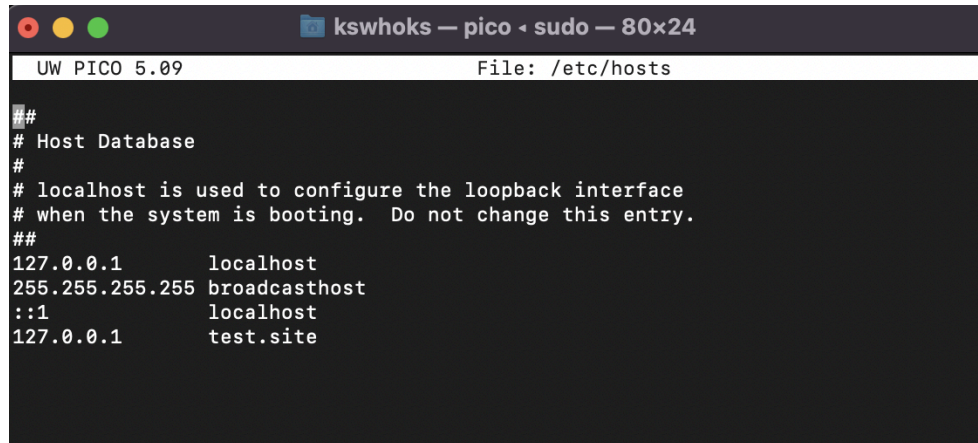
Блог

Мероприятия

Рисунок 18 – Сайт

Далее необходимо сделать так, чтобы при переходе по ссылке <http://test.site>, открывался сайт.

Для этого в hosts файл системы добавим следующее описание адреса.



```
kswhoks — nano — 80x24
UW NANO 2.9.24 File: /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1    localhost
255.255.255.255 broadcasthost
::1         localhost
127.0.0.1    test.site
```

Рисунок 19 – Редактирование файла hosts

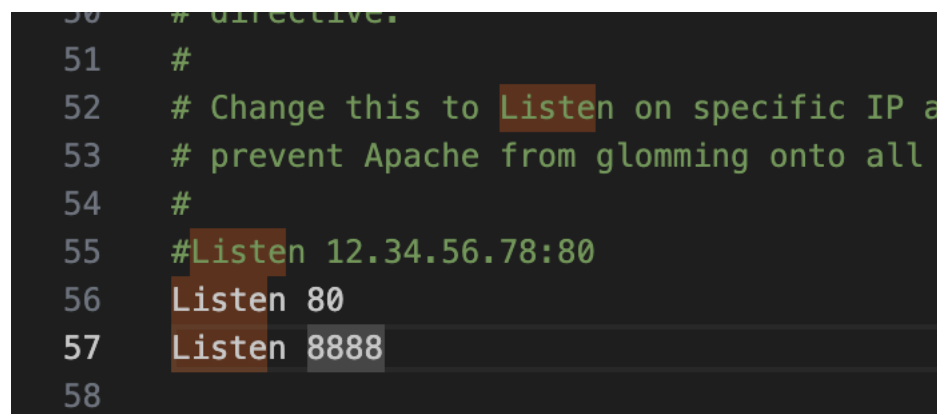
После этого нужно было настроить виртуальный хост в Apache, для этого был отредактирован файл httpd-vhosts.conf следующим образом:



```
<VirtualHost *:80>
    DocumentRoot "/Applications/MAMP/htdocs/wordpress"
    ServerName test.site
</VirtualHost>
```

Рисунок 20 – Файл httpd-vhosts.conf

Далее были добавлены следующие строки в файле apache – httpd.conf



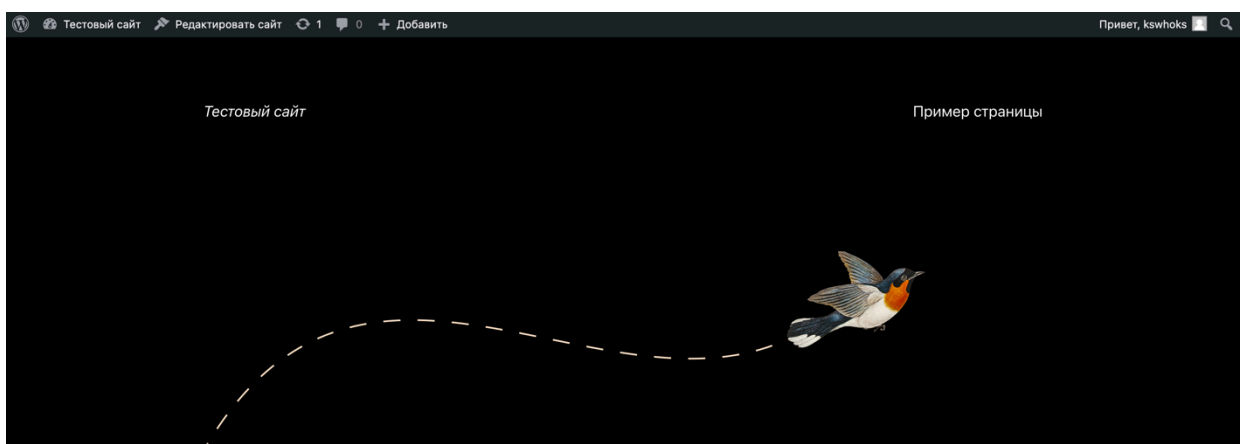
```
50 # directive.
51 #
52 # Change this to Listen on specific IP a
53 # prevent Apache from glomming onto all
54 #
55 #Listen 12.34.56.78:80
56 Listen 80
57 Listen 8888
58
```

Рисунок 21 – Файл httpd.conf

```
666
667 # Virtual hosts
668 Include /Applications/MAMP/conf/apache/extra/httpd-vhosts.conf
669
670 # Local access to the Apache HTTP Server Manual
671 #Include /Applications/MAMP/conf/apache/extra/httpd-manual.conf
672
```

Рисунок 22 – Файл httpd.conf

В конце в настройках на сайте wordpress заменим адрес сайта на <http://test.site> и перезапустим сервер. Теперь при вводе нашего адреса мы попадаем на тестовую страницу сайта wordpress.



Привет, мир!

Добро пожаловать в WordPress. Это ваша первая запись. Отредактируйте или удалите ее, затем начинайте создавать!

17 декабря, 2024

Рисунок 23 – Приветственная страница WordPress

Вывод

В ходе выполнения лабораторной работы цель была достигнута. Было произведено знакомство с PHP-скриптами и инструментарием для отладки проектов MAMP.

В течение выполнения работы были изучены использование параллельных и последовательных задач Gulp, после этого были изучены post- и get-запросы с помощью PHP-скриптов, был написан сайт с приемом и отображением обратной связи. Также, был установлен инструментарий для локальных серверов MAMP, с помощью которого была произведена работа с базой данных, скачан WordPress, произведена его установка и настроен переход на сайт по ссылке <http://test.site>.