

Университет ИТМО

Факультет «Инфокоммуникационных технологий»
Направление подготовки «название направления подготовки»

Лабораторная работа №3
«Разработка заданий с использованием Gulp»

Выполнил:
Кузнецов Никита Сергеевич
Группа К33212
Проверил:
Марченко Елена Вадимовна

Санкт-Петербург
2023

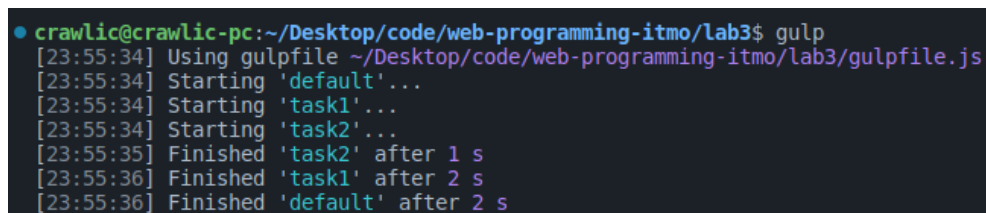
Цель работы: ознакомиться с библиотекой Gulp, настроить базовое выполнение задач, а также реализовать обновление браузера при сохранении файлов активной директории. Разобраться в предназначении GET и POST запросов на примере формы HTML и PHP. Базово ознакомиться с Wordpress и его запуском на локальном компьютере и через домен `http://test.site`.

Ход работы:

1 задание: настроить gulp: а) создать два taska – настроить на последовательное и параллельное выполнение; б) настроить отображение файлов проекта в браузере и автоматическую перезагрузку при изменении одного из контролируемых файлов проекта.

Для выполнения данного задания были использованы те же библиотеки, что и в предыдущем, а именно: gulp и gulp-cli. С помощью пакета gulp для node.js был разработан код JavaScript (приложение 1), который выполняет два taska сначала параллельно, потом последовательно.

Соответственно, если убрать комментарий с кода для параллельного исполнения и посмотреть результат выполнения файла (рис. 1) - то можно заметить, что taskи действительно выполнялись параллельно, так как общее время исполнения составило 2 секунды.



```
● crawllic@crawlic-pc:~/Desktop/code/web-programming-itmo/lab3$ gulp
[23:55:34] Using gulpfile ~/Desktop/code/web-programming-itmo/lab3/gulpfile.js
[23:55:34] Starting 'default'...
[23:55:34] Starting 'task1'...
[23:55:34] Starting 'task2'...
[23:55:35] Finished 'task2' after 1 s
[23:55:36] Finished 'task1' after 2 s
[23:55:36] Finished 'default' after 2 s
```

Рисунок 1 - Параллельное выполнение task1 и task2.

В свою очередь, если откомментировать последовательный запуск gulp, то можно увидеть, что выполнение заняло полные три секунды - время первого taska и второго вместе взятых (рис. 2)

```
● crawllic@crawlic-pc:~/Desktop/code/web-programming-itmo/lab3$ gulp
[00:00:02] Using gulpfile ~/Desktop/code/web-programming-itmo/lab3/gulpfile.js
[00:00:02] Starting 'default'...
[00:00:02] Starting 'task1'...
[00:00:04] Finished 'task1' after 2.01 s
[00:00:04] Starting 'task2'...
[00:00:05] Finished 'task2' after 1 s
[00:00:05] Finished 'default' after 3.01 s
```

Рисунок 2 - Последовательное выполнение task1 и task2.

Для выполнения пункта б) первого задания необходимо было настроить библиотеку BrowserSync, которую необходимо было установить через npm. После установки, был написан код, который создавал таск “serve”, запускающий BrowserSync в корневой директории на хосте 127.0.0.1 и порте 3000 (рис. 3), и который просматривал изменения во всех файлах с расширениями .html, .css, .js, .css.

```
○ crawllic@crawlic-pc:~/Desktop/code/web-programming-itmo/lab3$ gulp serve
[00:06:51] Using gulpfile ~/Desktop/code/web-programming-itmo/lab3/gulpfile.js
[00:06:51] Starting 'serve'...
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
    External: http://192.168.0.7:3000
-----
    UI: http://localhost:3001
    UI External: http://localhost:3001
-----
[Browsersync] Serving files from: ./
```

Рисунок 3 - Запуск BrowserSync с помощью таска “serve” через Gulp.

Для проверки работоспособности кода можно изменить любой файл, например сам исполняемый файл gulpfile.js (приложение 2), или любой другой файл с заданными расширениями. Для проверки кода был создан временный файл test.html, в котором внутри тэга <body> был помещен заголовок с надписью “hello” (рис. 4).

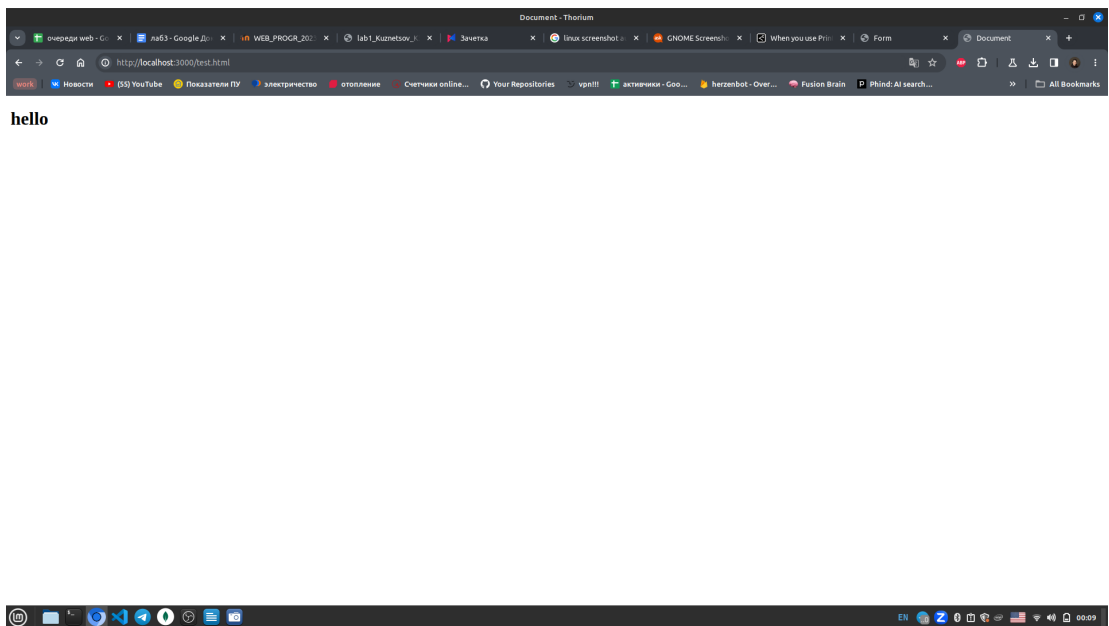


Рисунок 4 - Файл test.html в браузере до изменений.

Далее, этот файл был изменен, и изменения отразились как в консоли (рис. 5), так и в браузере (рис. 6).

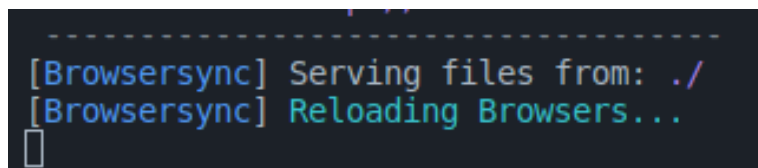


Рисунок 5 - BrowserSync заметил изменения и обновил браузер, отобразив это в консоли.

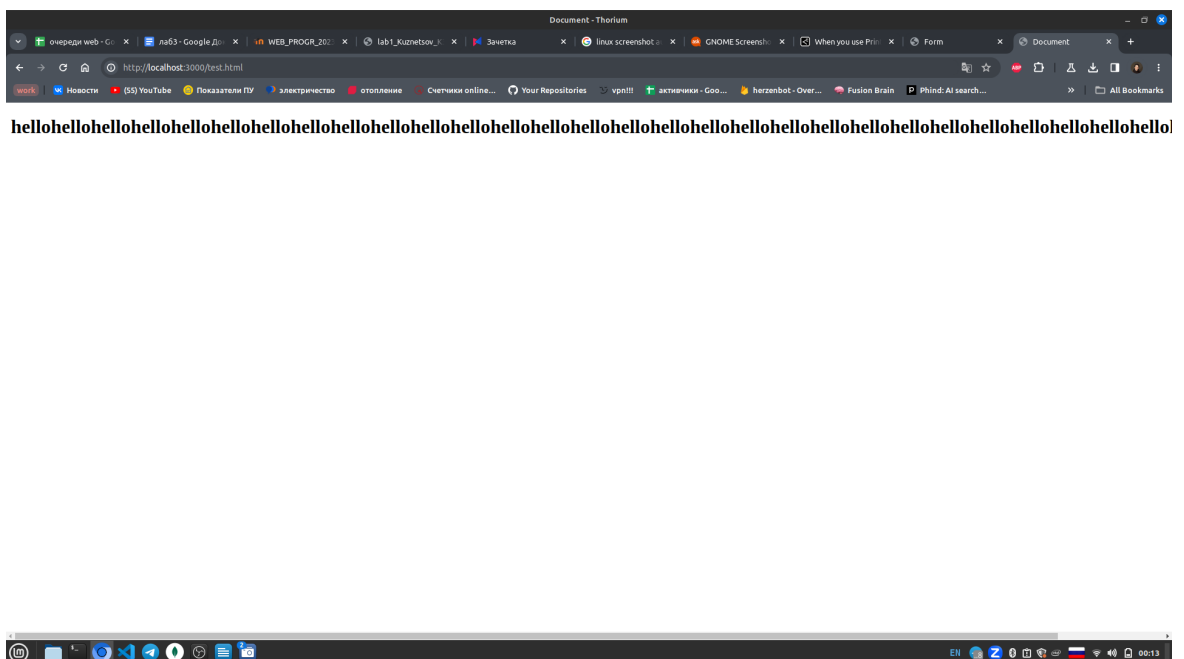


Рисунок 6 - Файл test.html после изменений, без перезагрузки браузера вручную.

2 задание: создать форму для отправки информации по обратной связи от пользователя сайта – передает информацию о себе: имя, фамилия, электронная почта, поле с обратной связью, должны быть радиокнопки (по меньшей мере 2 шт.) и должны быть чекбоксы (не менее трех). Разработать файл с формой и php скрипт по образцу. В отчете привести информацию о использовании методов get и метод post.

Для выполнения данного задания были разработаны файлы HTML (приложение 2) - для разметки формы, PHP (приложение 3) - для обработки запроса, а также CSS (приложение 4) - для добавления стилей к форме.

В файле index.html (приложение 2) находится сама форма с полями о пользователе: имя, фамилия, почта. Также, присутствует поле обратной связи, две радиокнопки “Лайк” и “Дизлайк” и три чекбокса с опциями “1”, “2” и “3” - в качестве примера.

К форме также прикреплен файл style.css через тэг <link> (приложение 4) и по “сохранению” формы отправляется POST-запрос к файлу submit.php (приложение 3), в котором обрабатывается ответ пользователя, заполнившего форму (рис. 7), и выводится сообщение, в зависимости от тех данных, что он отправил (рис. 8-9).

Важная форма

Имя:

Фамилия:

Электронная почта:

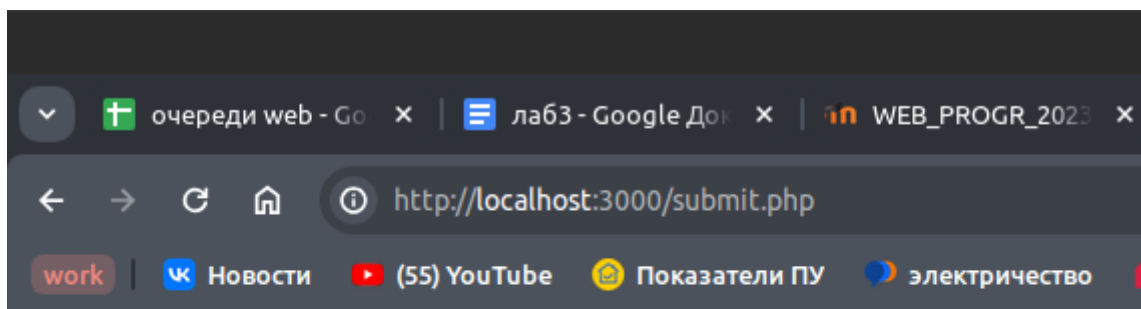
Обратная связь:

☐ Лайк ☐ Дизлайк

☐ 1 ☐ 2 ☐ 3

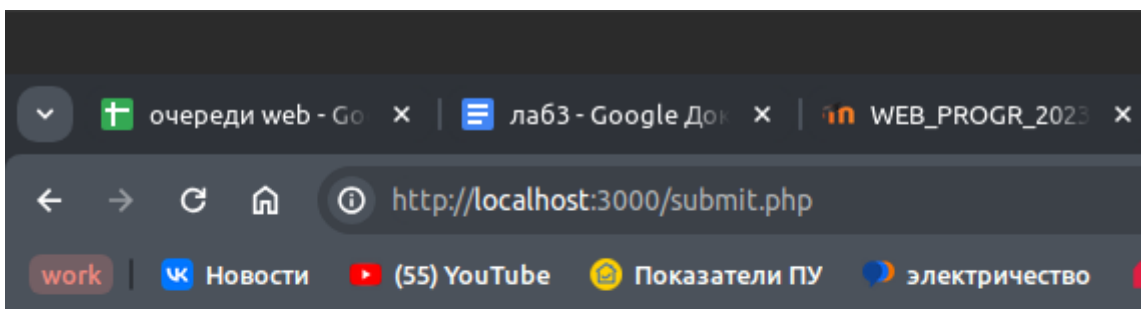
Submit

Рисунок 7 - Внешний вид формы в браузере.



Nikita
Kuznetsov
nik@mail.ru
эээ....
Вы не поставили никакой оценки..... ну ладно...

Рисунок 8 - Ответ пользователю, когда он не выбрал ни одну из радиокнопок или чекбоксов.



Nikita
Kuznetsov
nik@mail.ru
эээ....
Вы выбрали второй чекбокс
Вы выбрали третий чекбокс
Вы поставили лайк! УРА!

Рисунок 9 - Ответ пользователю при выборе радиокнопки и чекбоксов.

В данном задании использовалось два вида запросов - GET и POST. Первый запрос - происходит при открытии страницы в браузере. Так как в корневой директории есть файл index.html (приложение 2), то автоматически делается запрос к нему и показывается содержимое файла. Второй запрос - происходит после отправки формы. Так как в форме определен метод и путь до исполняемого

файла submit.php (приложение 3), то браузер отправляет запрос на него и выполняет тот код, который определен внутри PHP файла.

3 задание: установить инструментарий для отладки проектов (LAMP, Денвер - локальный сервер или что-то другое аналогичное на ваше усмотрение). С портала wordpress берете движок и устанавливаете его. Настраиваете портал `http://test.site` При вводе данного адреса отвечает ваш портал. При желании можете поставить одну из тем, которая Вам понравится.

Для выполнения данной задачи, необходимо было установить программный стек LAMP (Linux, Apache, MySQL, PHP). После установки необходимо было скачать архив Wordpress с официального сайта (рис. 10) и добавить его в директорию с файлами из предыдущих заданий.

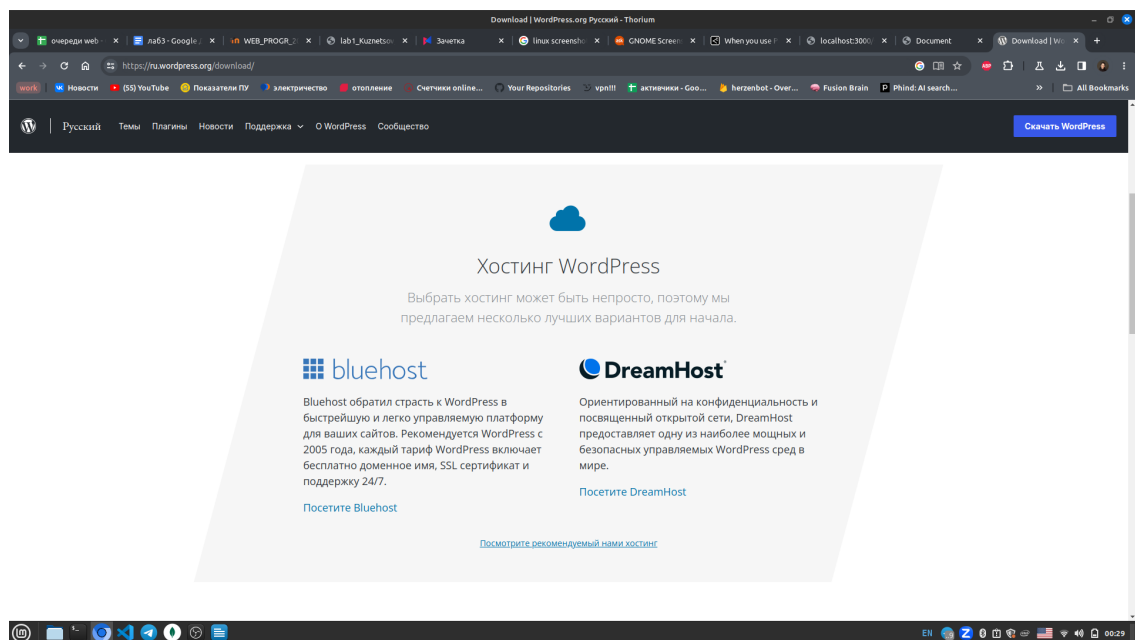


Рисунок 10 - Официальная страница для скачивания Wordpress.

Далее, необходимо было разобраться с MySQL. Для работы данной базы данных в Wordpress требуется создать пользователя и саму базу данных с помощью команд *CREATE DATABASE* и *CREATE USER*. Затем, необходимо записать полученного пользователя и базу данных в файл `wp-config.php` (приложение 5), находящийся в корневой папке Wordpress (рис. 11).

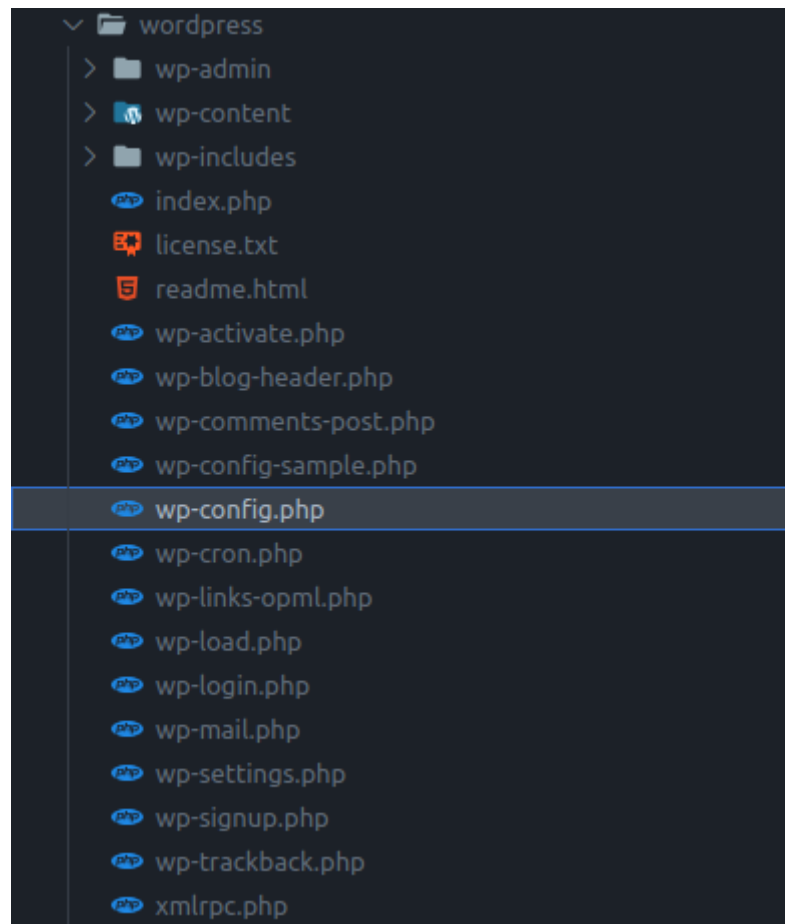


Рисунок 11 - Структура файлов сайта на Wordpress.

После добавления всех нужных параметров MySQL можно запускать сервер и переходить по ссылке `/wordpress/install.php`. В данном окне предлагается создать пользователя и пароль для доступа к админ-панели (рис. 12), а также email и прочие настройки. После добавления администратора, появляется экран успешной установки Wordpress (рис. 13) и перебрасывает на основную страницу панели администратора только что созданного сайта (рис. 14).

Добро пожаловать

Добро пожаловать в знаменитую пятиминутную установку WordPress! Просто заполните поля — и вперёд, к использованию самой мощной и гибкой персональной платформы для публикаций в мире!

Требуется информация

Пожалуйста, укажите следующую информацию. Не переживайте, потом вы всегда сможете изменить эти настройки.

Название сайта

my-site

Имя пользователя

crawlic

Имя пользователя может содержать только латинские буквы, пробелы, подчёркивания, дефисы, точки и символ @.

Пароль

password

Очень слабый

Скрыть

Важно: Этот пароль понадобится вам для входа. Сохраните его в надёжном месте.

Подтвердите пароль

☒ Разрешить использование слабого пароля.

Ваш e-mail

nik@mail.ru

Внимательно проверьте адрес электронной почты, перед тем как продолжить.


Видимость для поисковых систем

☒ Попросить поисковые системы не индексировать сайт

Будет ли учитываться этот запрос — зависит от поисковых систем.

Установить WordPress

Рисунок 12 - Создание администратора для админ-панели Wordpress.



Поздравляем!

WordPress установлен. Желаем успешной работы!

Имя пользователя

crawlic

Пароль

Выбранный вами пароль.

[Войти](#)

Рисунок 13 - Сообщение об успешной установке Wordpress.

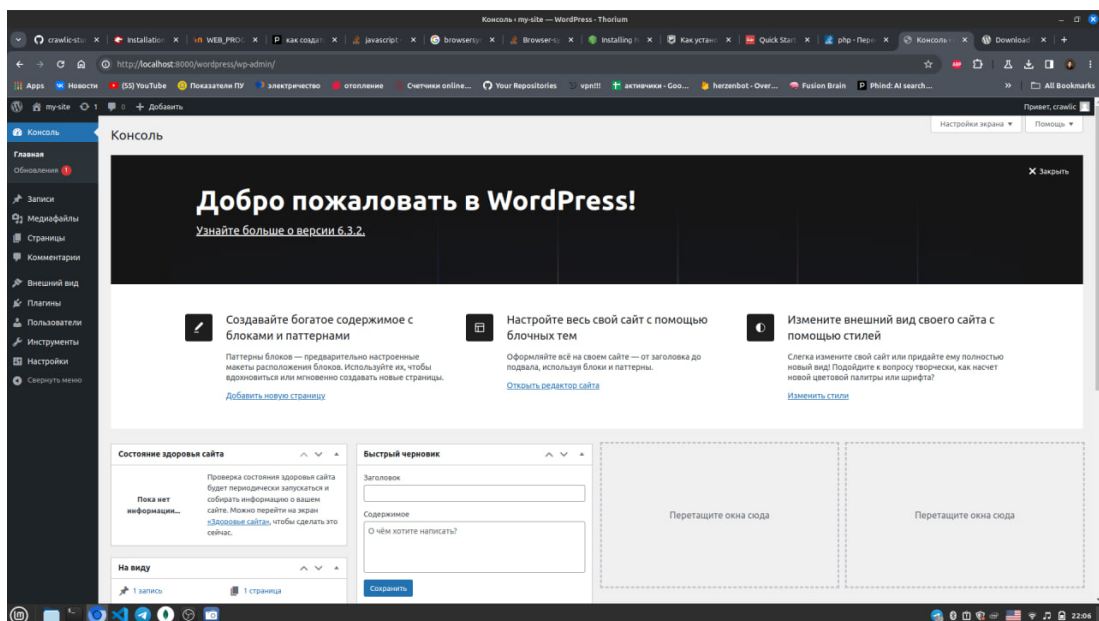


Рисунок 14 - Главная страница админ-панели Wordpress сразу после установки.

Далее, по заданию, необходимо было настроить портал <http://test.site> для доступа к сайту Wordpress. Для решения данной задачи необходимо было добавить данный “домен” в файл /etc/hosts на место адреса 127.0.0.1, который отвечает за localhost (рис. 15).

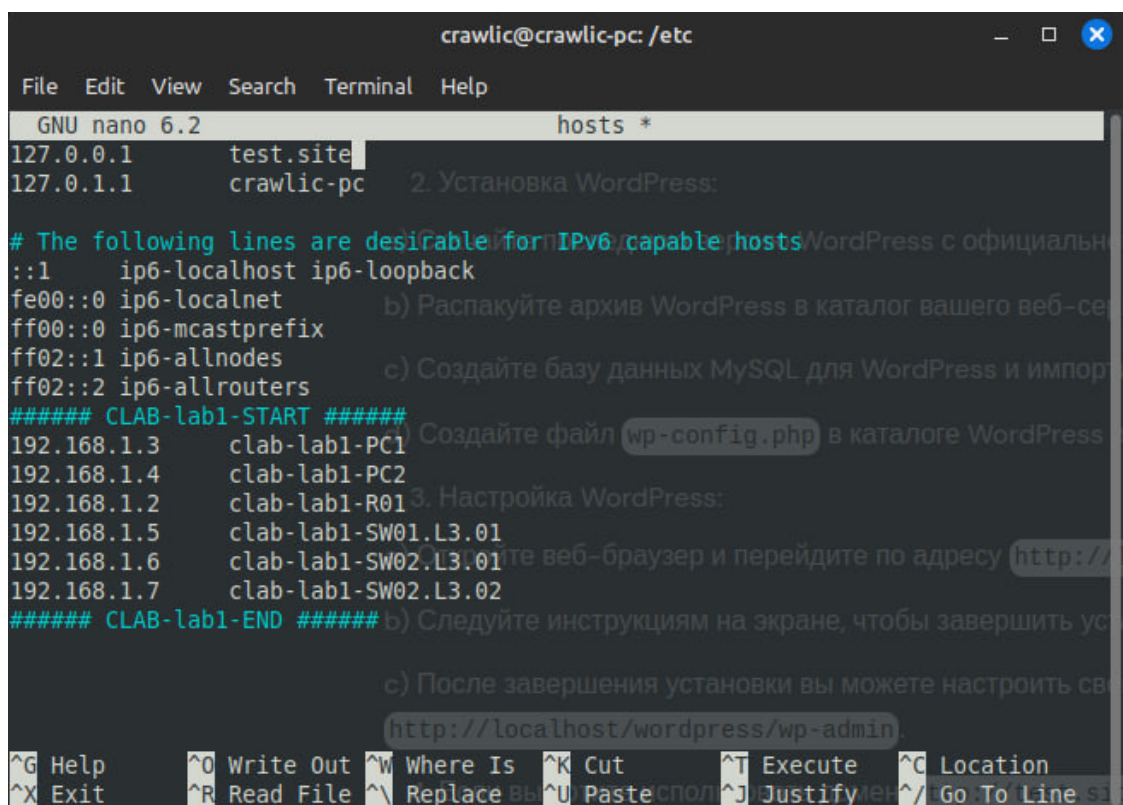


Рисунок 15 - Добавление портала test.site в /etc/hosts.

После изменения этого файла, при переходе по ссылке <http://test.site:8000/wordpress> с ранее запущенным сервером на 8000 порте открывается страница с созданным сайтом Wordpress (рис. 16).

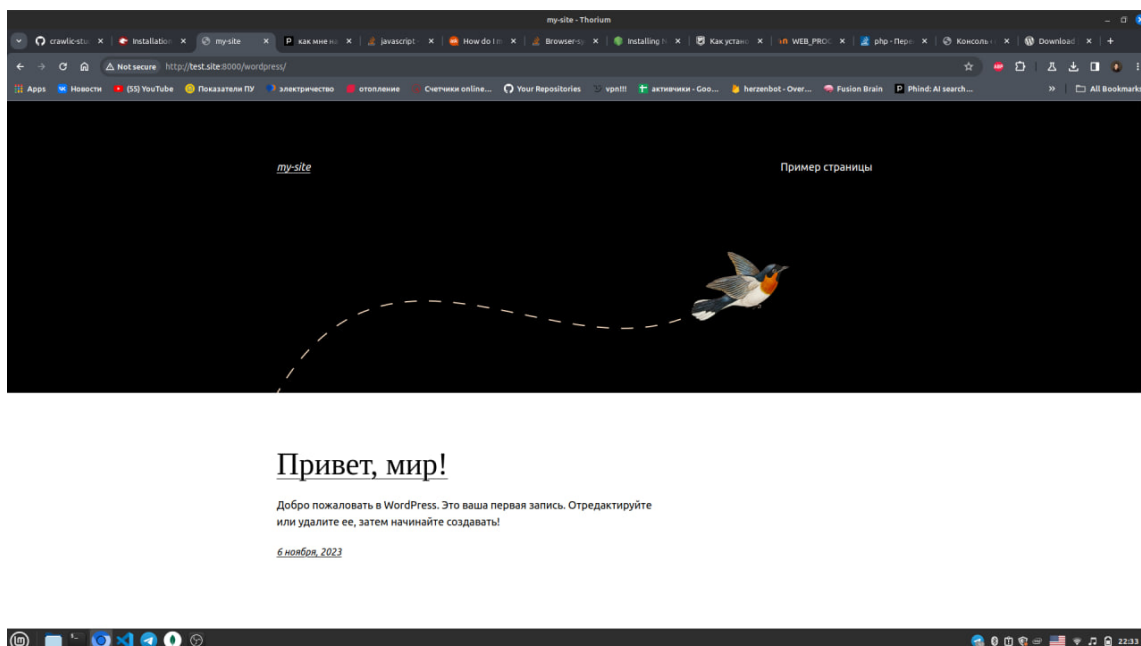


Рисунок 16 - Рабочая страница Wordpress на портале test.site.

Вывод: в ходе работы были разработаны исполнительные файлы для Gulp с разной последовательностью выполнения,. Также был реализован локальный сервер BrowserSync, обновляющий страницу при изменении файла. Была создана форма HTML, скрипт PHP для нее, а также разобраны различия между GET и POST запросами. В конце работы был настроен локальный сайт на Wordpress, а также локальная база данных MySQL. По результату работы были получены знания о разработке страниц с BrowserSync, а также практические навыки работы с Wordpress и MySQL.

Приложение 1. Файл gulpfile.js с кодом с использованием библиотеки Gulp и BrowserSync для JavaScript.

```
const gulp = require("gulp");
const browserSync = require("browser-sync").create();

function task1(cb) {
  setTimeout(cb, 2000);
}

function task2(cb) {
  setTimeout(cb, 1000);
}

gulp.task("serve", function () {
  browserSync.init({
    server: {
      baseDir: "./",
    },
    injectChanges: true,
  });

  gulp.watch("**/*.html").on("change", browserSync.reload);
  gulp.watch("**/*.css").on("change", browserSync.reload);
  gulp.watch("**/*.js").on("change", browserSync.reload);
  gulp.watch("**/*.php").on("change", browserSync.reload);
});

// exports.default = gulp.parallel(task1, task2);
// exports.default = gulp.series(task1, task2);
```

Приложение 2. Файл index.html с кодом для формы отправки пользователем своих данных и обратной связи.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Form</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Важная форма</h1>
  <form method="post" action="submit.php">
    <label for="name">Имя:</label><br>
    <input required type="text" id="name" name="name"><br>
    <label for="surname">Фамилия:</label><br>
    <input required type="text" id="surname" name="surname"><br>
    <label for="email">Электронная почта:</label><br>
    <input required type="text" id="email" name="email"><br>
    <label for="feedback">Обратная связь:</label><br>
    <textarea required id="feedback" name="feedback"></textarea><br>
    <div class="radio-container">
      <input type="radio" id="option1" name="option" value="option1">
      <label for="option1">Лайк</label><br>
      <input type="radio" id="option2" name="option" value="option2">
      <label for="option2">Дизлайк</label><br>
    </div>
    <div class="radio-container">
      <input type="checkbox" id="checkbox1" name="checkbox1" value="Option1">
      <label for="checkbox1">1</label><br>
      <input type="checkbox" id="checkbox2" name="checkbox2" value="Option2">
      <label for="checkbox2">2</label><br>
      <input type="checkbox" id="checkbox3" name="checkbox3" value="Option3">
      <label for="checkbox3">3</label><br>
    </div>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

Приложение 3. Файл submit.php с кодом для обработки POST-запроса от формы в index.html.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST["name"];
    $surname = $_POST["surname"];
    $email = $_POST["email"];
    $feedback = $_POST["feedback"];

    $one = $_POST["checkbox1"];
    $two = $_POST["checkbox2"];
    $three = $_POST["checkbox3"];

    echo $name . '<br>';
    echo $surname . '<br>';
    echo $email . '<br>';
    echo $feedback . '<br>';

    if (isset($one)) {
        print("Вы выбрали первый чекбокс<br>");
    }

    if (isset($two)) {
        print("Вы выбрали второй чекбокс<br>");
    }

    if (isset($three)) {
        print("Вы выбрали третий чекбокс<br>");
    }

    $option = $_POST["option"];

    if (!isset($option)) {
        echo "Вы не поставили никакой оценки..... ну ладно...<br>";
    } else if ($option == "option1") {
        echo "Вы поставили лайк! УРА!<br>";
    } else {
        echo "Вы поставили дизлайк :(<br>";
    }
}
?>
```

Приложение 4. Файл style.css со стилями для формы в index.html.

```
body {
    font-family: Arial, Helvetica, sans-serif;
}
h1 {
    text-align: center;
}
.radio-container {
    display: flex;
    justify-content: space-between;
}
form {
    width: 300px;
    margin: 0 auto;
}
label {
    display: block;
    margin-top: 20px;
}
input[type="text"], textarea {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
textarea {
    height: 100px;
    resize: none;
}
input[type="radio"], input[type="checkbox"] {
    margin-top: 20px;
    position: relative;
    width: 20px;
    margin-right: 5px;
}

input[type="submit"] {
    margin-top: 10px;
    display: block;
    width: 100%;
    padding: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

input[type="submit"]:hover {
    background-color: #45a049;
}
```

Приложение 5. Файл wp-config.php с конфигурацией сайта Wordpress.

```
<?php
/**
 * Основные параметры WordPress.
 *
 * Скрипт для создания wp-config.php использует этот файл в процессе установки.
 * Необязательно использовать веб-интерфейс, можно скопировать файл в "wp-config.php"
 * и заполнить значения вручную.
 *
 * Этот файл содержит следующие параметры:
 *
 * * * Настройки базы данных
 * * Секретные ключи
 * * Префикс таблиц базы данных
 * * ABSPATH
 *
 * @link https://ru.wordpress.org/support/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** Параметры базы данных: Эту информацию можно получить у вашего
хостинг-провайдера ** //
/** Имя базы данных для WordPress */
define( 'DB_NAME', 'wordpress' );

/** Имя пользователя базы данных */
define( 'DB_USER', 'crawlic' );

/** Пароль к базе данных */
define( 'DB_PASSWORD', 'Password_123.' );

/** Имя сервера базы данных */
define( 'DB_HOST', 'localhost' );

/** Кодировка базы данных для создания таблиц. */
define( 'DB_CHARSET', 'utf8' );

/** Схема сопоставления. Не меняйте, если не уверены. */
define( 'DB_COLLATE', '' );

/**#@+
 * Уникальные ключи и соли для аутентификации.
 *
 * Смените значение каждой константы на уникальную фразу. Можно сгенерировать их с
помощью
 * {@link https://api.wordpress.org/secret-key/1.1/salt/ сервиса ключей на
WordPress.org}.
 *
 * Можно изменить их, чтобы сделать существующие файлы cookies недействительными.
 * Пользователям потребуется авторизоваться снова.
 *
 * @since 2.6.0
```



```

*/
define( 'AUTH_KEY',          'впишите сюда уникальную фразу' );
define( 'SECURE_AUTH_KEY',   'впишите сюда уникальную фразу' );
define( 'LOGGED_IN_KEY',     'впишите сюда уникальную фразу' );
define( 'NONCE_KEY',         'впишите сюда уникальную фразу' );
define( 'AUTH_SALT',         'впишите сюда уникальную фразу' );
define( 'SECURE_AUTH_SALT',  'впишите сюда уникальную фразу' );
define( 'LOGGED_IN_SALT',    'впишите сюда уникальную фразу' );
define( 'NONCE_SALT',        'впишите сюда уникальную фразу' );

/**#@-*/

/**
 * Префикс таблиц в базе данных WordPress.
 *
 * Можно установить несколько сайтов в одну базу данных, если использовать
 * разные префиксы. Пожалуйста, указывайте только цифры, буквы и знак подчеркивания.
 */
$table_prefix = 'wp_';

/**
 * Для разработчиков: Режим отладки WordPress.
 *
 * Измените это значение на true, чтобы включить отображение уведомлений при
разработке.
 *
 * Разработчикам плагинов и тем настоятельно рекомендуется использовать WP_DEBUG
 * в своём рабочем окружении.
 *
 * Информацию о других отладочных константах можно найти в документации.
 *
 * @link https://ru.wordpress.org/support/article/debugging-in-wordpress/
 */
define( 'WP_DEBUG', false );

/* Произвольные значения добавляйте между этой строкой и надписью "дальше не
редактируем". */

/* Это всё, дальше не редактируем. Успехов! */

/** Абсолютный путь к директории WordPress. */
if ( ! defined( 'ABSPATH' ) ) {
    define( 'ABSPATH', __DIR__ . '/' );
}

/** Инициализирует переменные WordPress и подключает файлы. */
require_once ABSPATH . 'wp-settings.php';

```