Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ 3

По дисциплине Web-программирование

Тема работы Задание 3

Обучающийся Боженко Мария Александровна

Факультет инфокоммуникационных технологий

Группа К3321

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	17.12.2024 (дата)	_____ (подпись)	Боженко М.А. (Ф.И.О.)
Руководитель	_____ (дата)	_____ (подпись)	Марченко Е.В. (Ф.И.О.)

Санкт-Петербург
2024

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ

Задачи, поставленные в данной лабораторной работе:

1. Настройка Gulp
   (a) Создание двух тасков, настроенных на последовательное и параллельное выполнение
   (b) Настройка отображения файлов проекта в браузере с автоматической перезагрузкой при изменениях файлов
2. Создание формы обратной связи
3. Настройка wordpress

1 Настройка Gulp

1.1 Параллельное и последовательное выполнение

По аналогии с предыдущей лабораторной работой в созданном проекте был настроен Gulp. Далее был создан файл gulpfile.js, в котором прописаны два таска, которые выполняют функции последовательно и параллельно. Код файла представлен ниже:

```
const gulp = require('gulp');

Ссылки: 2
function funcOne(done) {
    console.log('Doing the FIRST function');
    done();
}

Ссылки: 2
function funcTwo(done) {
    console.log('Doing the SECOND function');
    done();
}

const seriesTask = gulp.series(funcOne, funcTwo);

const parallelTask = gulp.parallel(funcOne, funcTwo);

exports.seriesTask = seriesTask;
exports.parallelTask = parallelTask;
```

Рисунок 1.1 — Код gulpfile.js

Далее таски были запущены по очереди. Как видно на рис. 1.2 в первом варианте сначала была запущена первая функция, и только после того, как она завершилась, была запущена вторая, в то время как во втором варианте, вторая функция запустилась сразу после первой, не дожидаясь ее завершения.

```
mi@mashab-laptop MINGW64 ~/Desktop/УНИВЕР/3 КУРС/Web-программирование/Лаба_3/Web
Development_2024-2025/works/K3321/Боженко_Мария/lab3 (lab_3)
$ gulp seriesTask
[18:38:51] Using gulpfile ~\Desktop\УНИВЕР\3 КУРС\Web-программирование\Лаба_3\We
bDevelopment_2024-2025\works\K3321\Боженко_Мария\lab3\gulpfile.js
[18:38:51] Starting 'seriesTask'...
[18:38:51] Starting 'funcOne'...
Doing the FIRST function
[18:38:51] Finished 'funcOne' after 1.51 ms
[18:38:51] Starting 'funcTwo'...
Doing the SECOND function
[18:38:51] Finished 'funcTwo' after 1.42 ms
[18:38:51] Finished 'seriesTask' after 6.94 ms

mi@mashab-laptop MINGW64 ~/Desktop/УНИВЕР/3 КУРС/Web-программирование/Лаба_3/Web
Development_2024-2025/works/K3321/Боженко_Мария/lab3 (lab_3)
$ gulp parallelTask
[18:39:11] Using gulpfile ~\Desktop\УНИВЕР\3 КУРС\Web-программирование\Лаба_3\We
bDevelopment_2024-2025\works\K3321\Боженко_Мария\lab3\gulpfile.js
[18:39:11] Starting 'parallelTask'...
[18:39:11] Starting 'funcOne'...
[18:39:11] Starting 'funcTwo'...
Doing the FIRST function
[18:39:11] Finished 'funcOne' after 1.89 ms
Doing the SECOND function
[18:39:11] Finished 'funcTwo' after 2.45 ms
[18:39:11] Finished 'parallelTask' after 5.55 ms
```

Рисунок 1.2 — Результат запуска тасков

1.2 Автоматическая перезагрузка страницы при изменениях

В данной части задания требовалось настроить отображение файлов проекта в браузере и автоматическую перезагрузку при изменении одного из контролируемых файлов проекта.

Прежде всего была создана простая структура проекта. Добавлена папка src, в которой будут храниться наши исходные файлы: html-страничка и таблица стилей.

А также была добавлена папка dist, в которую будут выводиться html и css-файлы.

Далее были установлены необходимые зависимости: browser-sync для автоматической перезагрузки браузера; gulp-sass для использования sass - расширения css.

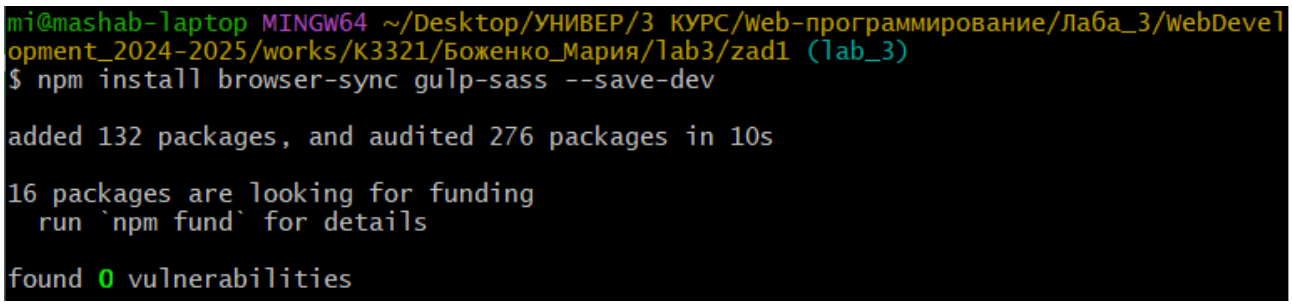

Рисунок 1.3 — Установка зависимостей

Далее был настроен непосредственно gulpfile.js

Были определены пути к исходным и выходным файлам.

Прописаны задачи:

- styles: компилирует sass в css и обновляет браузер
- html: копирует html файлы и обновляет браузер
- serve: инициализирует browser-sync и устанавливает наблюдение за изменениями в файлах

Далее были экспортированы задачи, и задача по умолчанию, которая сначала запускает компиляцию стилей и html, а потом запускает сервер.

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

const paths = {
    styles: {
        src: 'src/scss/**/*.scss',
        dest: 'dist/css'
    },
    html: {
        src: 'src/*.html',
        dest: 'dist/'
    }
};

Ссылки: 3
function styles() {
    return gulp.src(paths.styles.src)
        .pipe(sass().on('error', sass.logError))
        .pipe(gulp.dest(paths.styles.dest))
        .pipe(browserSync.stream());
}

Ссылки: 3
function html() {
    return gulp.src(paths.html.src)
        .pipe(gulp.dest(paths.html.dest))
        .pipe(browserSync.stream());
}

Ссылки: 2
function serve() {
    browserSync.init({
        server: {
            baseDir: 'dist'
        }
    });

    gulp.watch(paths.styles.src, styles);
    gulp.watch(paths.html.src, html);
}

exports.styles = styles;
exports.html = html;
exports.serve = serve;

//const def = gulp.series(gulp.parallel(styles, html), serve);

//exports.def = def;

exports.default = gulp.series(gulp.parallel(styles, html), serve)
```

Рисунок 1.4 — Скрипт gulpfile.js

Для проверки работы тасков, были написаны простые html и css файлы.

```scss
body {
    background-color: lightblue;
    font-family: Arial, sans-serif;
}

h1 {
    color: darkblue;
}
```

Рисунок 1.5 — Файл styles.scss

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="css/styles.css">
        <title>Gulp Test</title>
    </head>
    <body>
        <h1>Hello, Gulp!</h1>
    </body>
</html>
```

Рисунок 1.6 — Файл index.html

Далее был запущен gulp и открыта страница в браузере http://localhost:3000, получившаяся страница представлена ниже.

Рисунок 1.7 — Веб-страница до изменений

Если перейти в файл styles.scss, и поменять, например, цвет для тега h1 на красный и сохранить изменения, они автоматически отобразятся на странице, открытой в браузере.

```
body {
    background-color: lightblue;
    font-family: Arial, sans-serif;
}

h1 {
    color: red;
}
```

Рисунок 1.8 — Измененный scss файл

Рисунок 1.9 — Веб-страница после изменений

Таким образом, все изменения, которые происходят с исходными файлами, автоматически отображаются в браузере.

В терминале можно увидеть, как продолжают выполняться таски при обновлении файлов.

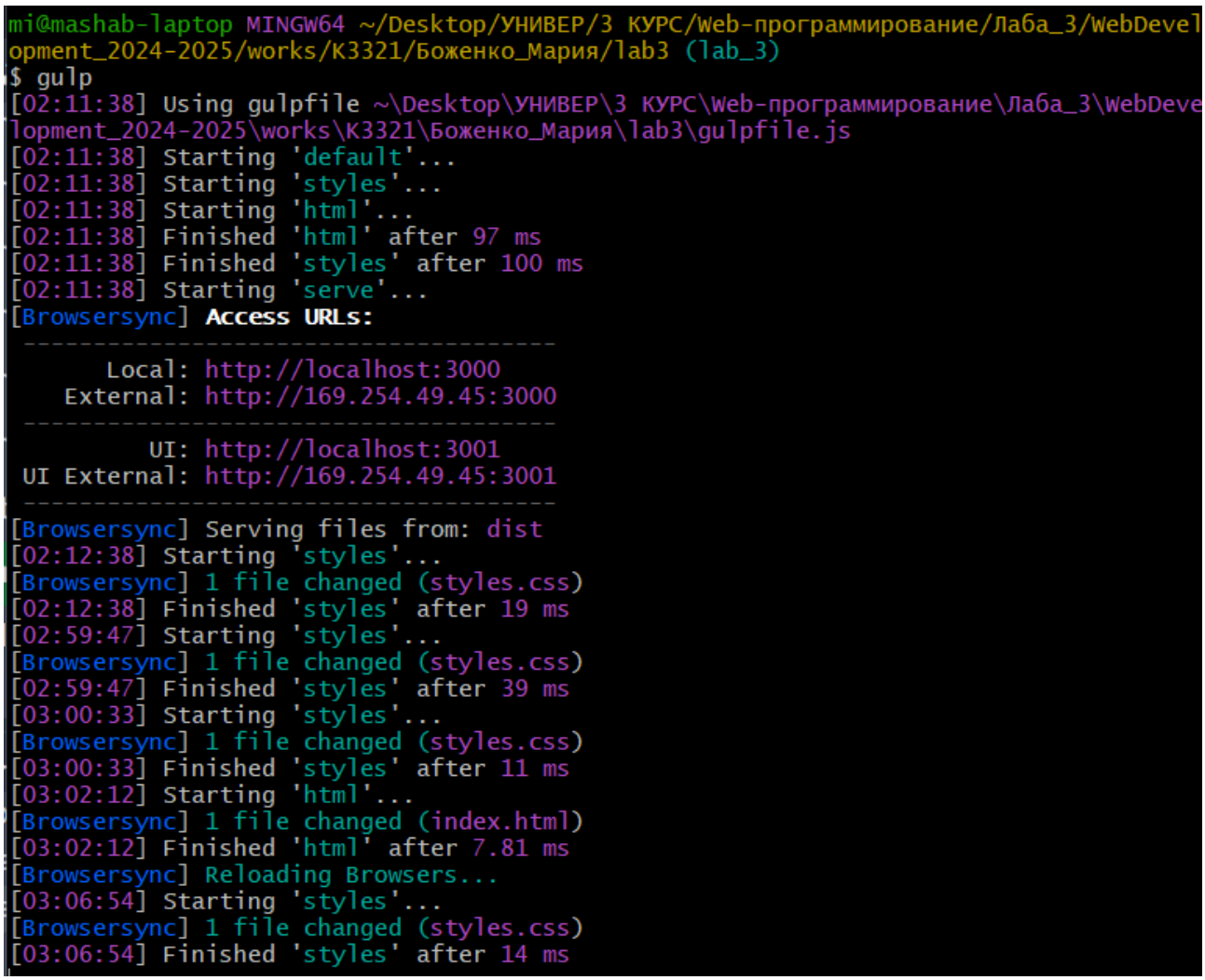

Рисунок 1.10 — Выполнение тасков

2 Форма для отправки обратной связи

В этой части задания нужно было создать форму для отправки информации по обратной связи от пользователя сайта. В форме передаются основные данные пользователя, а также должны присутствовать радиокнопки и чекбоксы.

Для радиокнопок были выбраны темы обращения (жалоба или предложение), а для чекбоксов выбраны источники, в которых пользователь узнал о сервисе.

Ниже представлен html файл с формой.

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Обратная связь</title>
    </head>
    <body>
        <h1>Форма обратной связи</h1>
        <form action="process_feedback.php" method="post">

            <label for="first_name">Имя:</label></br>
            <input type="text" id="first_name" name="first_name" required></br></br>

            <label for="last_name">Фамилия:</label></br>
            <input type="text" id="last_name" name="last_name" required></br></br>

            <label for="email">Электронная почта:</label></br>
            <input type="email" id="email" name="email" required></br></br>

            <label for="first_name">Ваше сообщение:</label></br>
            <textarea id="feedback" name="feedback" rows="4" required></textarea></br></br>

            <label>Выберете тему:</label></br>
            <input type="radio" id="topic1" name="topic" value="Жалоба" required>
            <label for="topic1">Жалоба</label></br>
            <input type="radio" id="topic2" name="topic" value="Предложение" required>
            <label for="topic2">Предложение</label></br>

            <label>Как вы о нас узнали?</label></br>
            <input type="checkbox" id="source1" name="source[]" value="Социальные сети">
            <label for="source1">Социальные сети</label></br>
            <input type="checkbox" id="source2" name="source[]" value="Рекомендации">
            <label for="source2">Рекомендации</label></br>
            <input type="checkbox" id="source3" name="source[]" value="Поиск в интернете">
            <label for="source3">Поиск в интернете</label></br>
            <input type="checkbox" id="source4" name="source[]" value="Реклама">
            <label for="source4">Реклама</label></br>
            <input type="checkbox" id="source5" name="source[]" value="Другое">
            <label for="source5">Другое</label></br>

            <input type="submit" value="Отправить">
        </form>
    </body>
</html>
```

Рисунок 2.1 — Файл feedback_form.html

Для обработки формы был написан php файл, который принимает информацию от пользователя и записывает ее в файл.

```php
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $first_name = htmlspecialchars(trim($_POST['first_name']));
        $last_name = htmlspecialchars(trim($_POST['last_name']));
        $email = htmlspecialchars(trim($_POST['email']));
        $feedback = htmlspecialchars(trim($_POST['feedback']));
        $topic = htmlspecialchars(trim($_POST['topic']));

        $sources = isset($_POST['source']) ? implode(", ", $_POST['source']) : 'Нет источников';

        $file = 'example.txt';
        $data = "Имя: " . $first_name . "\nФамилия: " . $last_name . "\nEmail: " . $email .
        "\nСообщение: " . $feedback . "\nТема: " . $topic . "\nИсточники: " . $sources . "\n";

        $result = file_put_contents($file, $data, FILE_APPEND);

        // Проверяем результат
        if ($result !== false) {
            echo "Данные успешно записаны в файл.";
        } else {
            echo "Произошла ошибка при записи в файл.";
        }
    }
?>
```

Рисунок 2.2 — Файл process_feedback.php

Для тестирования php-кода был установлен веб-сервер XAMPP. В папку xampp/htdocs были добавлен файлы проекта. При запуске html страницы, в браузере появилась написанная форма.

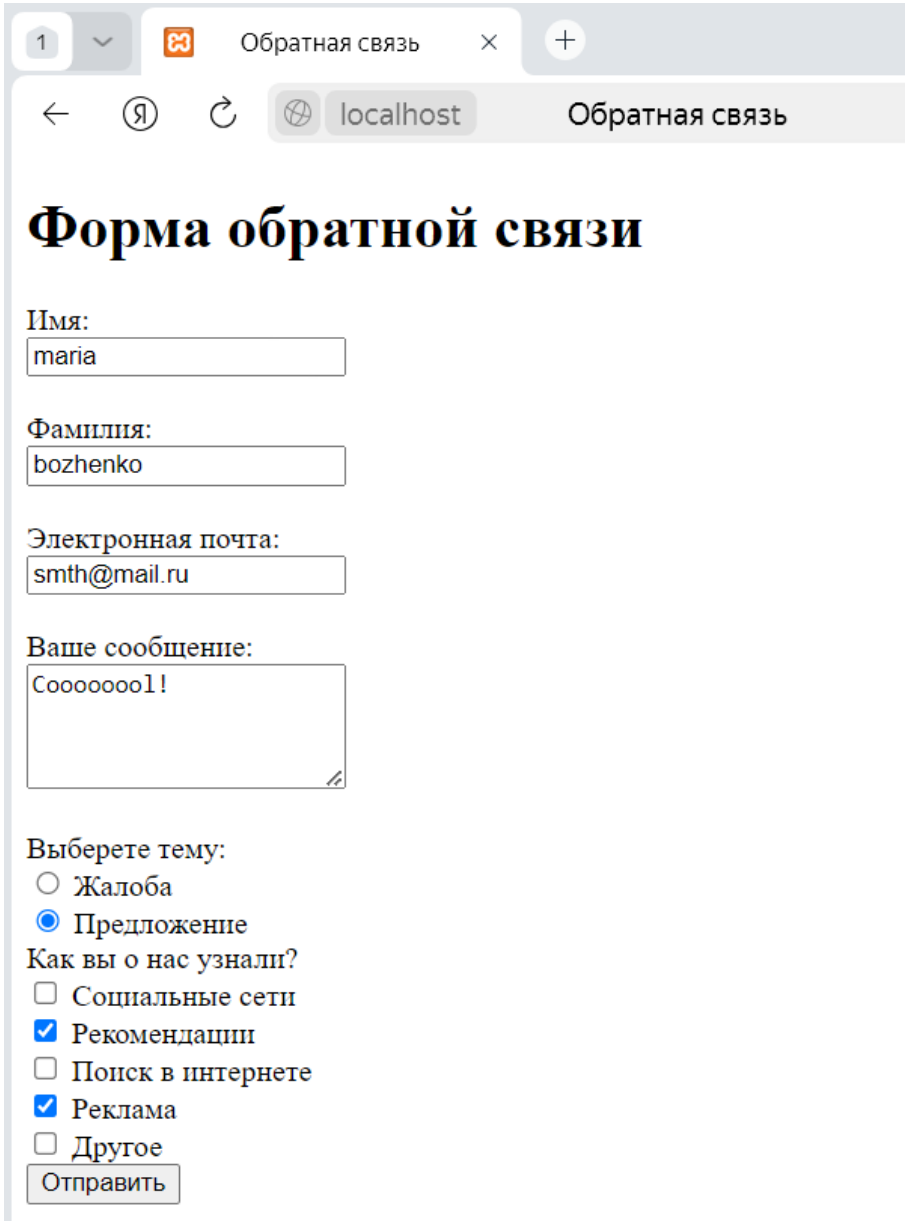

Рисунок 2.3 — Страничка формы

После отправки данных в браузере вывелось сообщение об удачной записи данных в файл.

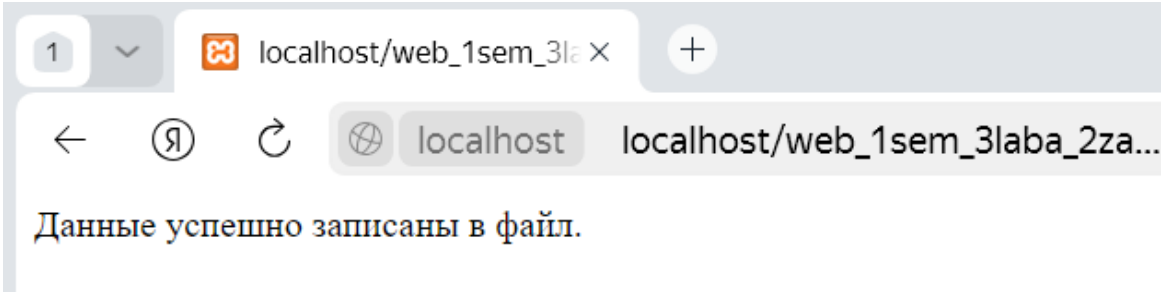

Рисунок 2.4 — Сообщение об успешной записи

Если открыть файл example.txt можно увидеть введенные пользователем данные. Таким образом форма работает корректно.

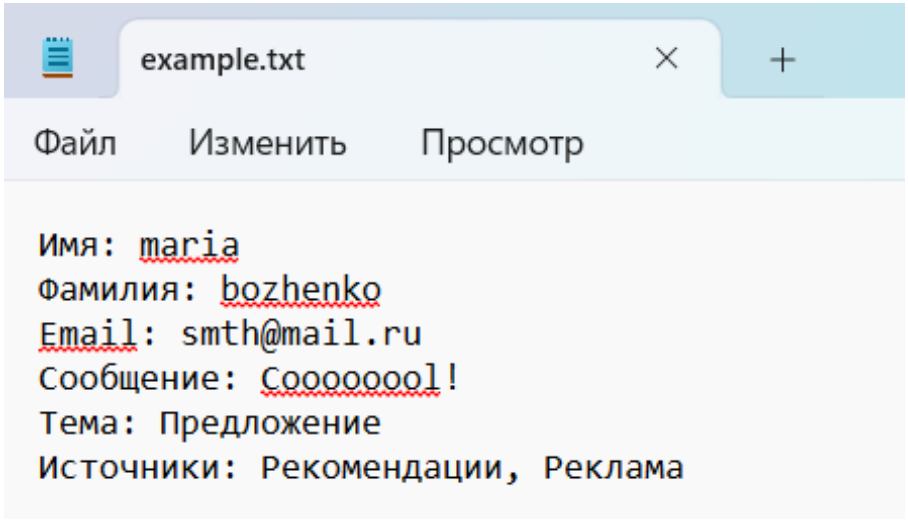

Рисунок 2.5 — Текстовый файл с данными

Метод POST это один из методов HTTP, который используется для отправки данных на сервер. Он часто применяется в веб-формах для передачи информации, как в данном примере.

Метод POST позволяет отправлять данные в теле HTTP-запроса. Это позволяет передавать больше информации по сравнению с методом GET, который включает данные в URL.

Соответсвенно метод «менее заметен» по сравнению с методом GET, поскольку данные не отображаются в адресной строке браузера. То есть метод является более безопасным, нежели GET.

3 Работа с WordPress

В данной части лабораторной нужно было установить инструментарий для отладки проектов (был выбран XAMPP), установить движок wordpress и настроить портал http://test.site

Прежде всего требовалось установить локальный сервер. Это было сделано в прошлой части лабораторной. В панели управления XAMPP были запущены Apache и MySQL.

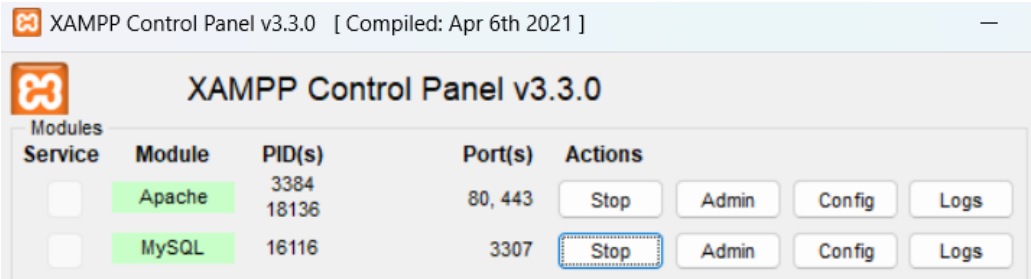

Рисунок 3.1 — Панель управления XAMPP

Далее с официального сайта был скачен WordPress. Zip-файл был разархивирован в папку локального сервера.

Была создана база данных. Перейдя по ссылке http://localhost/phpadmin, во вкладке Базы данных, была создана новая с именем new_test_site_db. В папке с установленным wordpress был изменен файл wp-config.php

```
/** The name of the database for WordPress */
define( 'DB_NAME', 'new_test_site_db' );

/** Database username */
define( 'DB_USER', 'root' );

/** Database password */
define( 'DB_PASSWORD', '' );

/** Database hostname */
define( 'DB_HOST', 'localhost:3307' );
```

Рисунок 3.2 — Изменения в файле wp-config.php

В браузере далее был введен адрес http://localhost/test.site, заполнены данные о будущем сайте и запущена установка wordpress.

Теперь перейдя по той же ссылке, открывается наш новый сайт. Зайдя в админку wordpress (по адресу http://localhost/test.site/wp-admin) мы имеем возможность поменять тему.

Итоговый сайт имеет следующий вид:

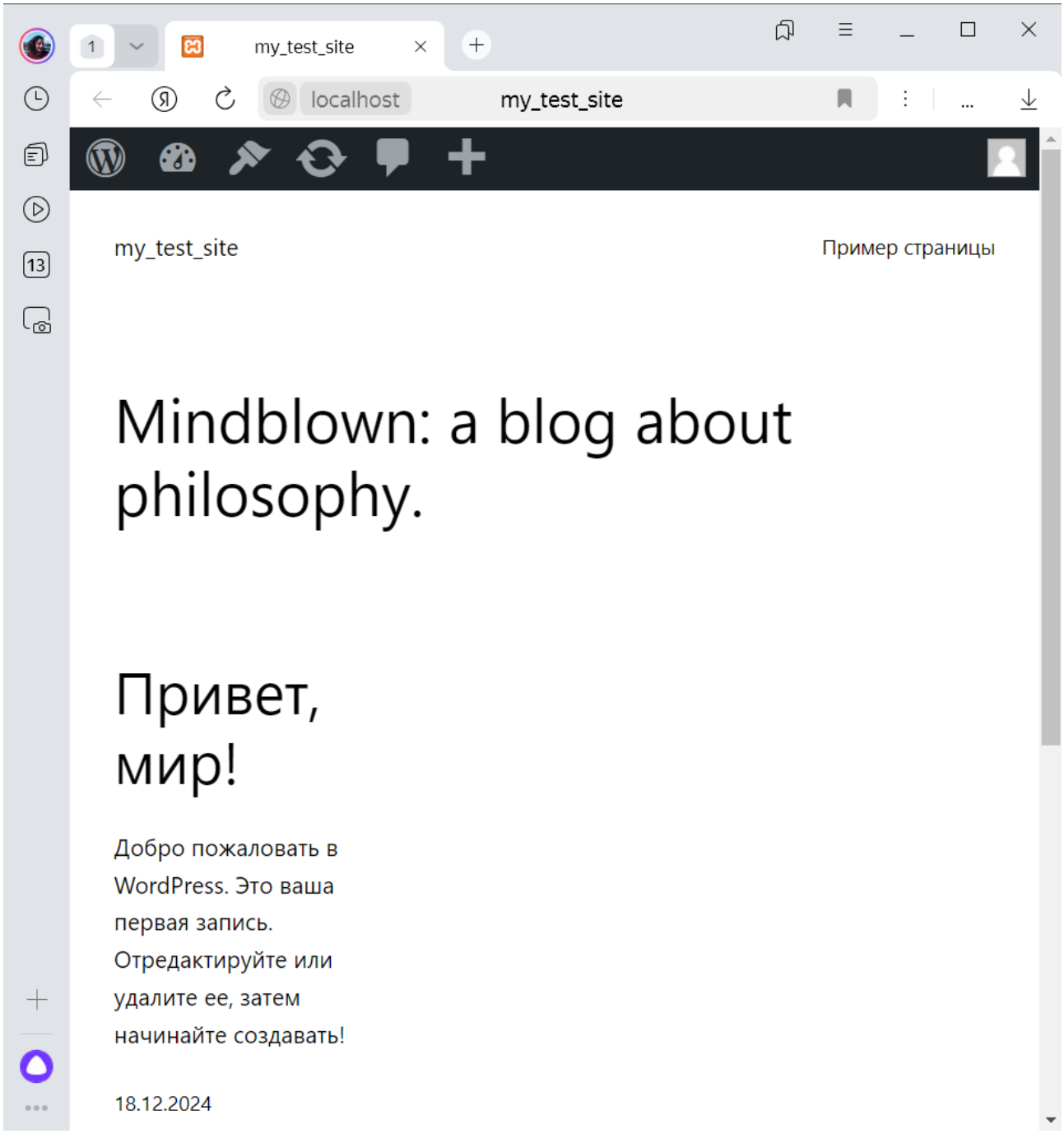

Рисунок 3.3 — Итоговый сайт

ЗАКЛЮЧЕНИЕ

В ходе проведения данной лабораторной работы были изучены основы работы с PHP и Wordpress. Были достигнуты все поставленные задачи, все файлы работают корректно.