

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

Факультет «Факультет инфокоммуникационных технологий»

Направление подготовки «Инфокоммуникационные технологии и системы
связи»

Отчет по дисциплине: «Web-программирование»

Практическая работа №3

Выполнила:

Колтунова Полина Владимировна

Группа №К3320

Проверила:

Марченко Елена Вадимовна

Санкт-Петербург

2024

ВВЕДЕНИЕ

Целью работы является изучение gulp, форм, php скриптов и Wordpress.

Задачи:

1. Настроить gulp: а) создать два таска – настроить на последовательное и параллельное выполнение; б) настроить отображение файлов проекта в браузере и автоматическую перезагрузку при изменении одного из контролируемых файлов проекта.
2. Создать форму для отправки информации по обратной связи от пользователя сайта – передает информацию о себе: имя, фамилия, электронная почта, поле с обратной связью, создать радиокнопки и чекбоксы. Разработать файл с формой и php скрипт по образцу.
3. Установить инструментарий для отладки проектов. Установить движок с портала wordpress. Настроить портал <http://test.site>, чтобы при вводе данного адреса, отвечал настроенный портал.

ГЛАВА 1. ХОД РАБОТЫ

Задание 1. Настройка gulp и создание тасков

Пункт а.

Для последовательного выполнения использована функция `series()`, которая параметрами принимает любое количество задач и выполняет их по очереди в порядке перечисления (рис.1).

```
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp> gulp --gulpfile gulpfile_series.js
[10:44:45] Using gulpfile ~\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp\gulpfile_series.js
[10:44:45] Starting 'default'...
[10:44:45] Starting 'task1'...
First task is being completed
[10:44:45] Finished 'task1' after 849 µs
[10:44:45] Starting 'task2'...
Second task is being completed
[10:44:45] Finished 'task2' after 739 µs
[10:44:45] Finished 'default' after 4.16 ms
```

Рисунок 1 - Последовательно выполнение

Для параллельного выполнения использована функция `parallel()`, которая параллельно выполняет задачи, передаваемые в качестве параметров (рис.2).

```
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp> gulp --gulpfile gulpfile_parallel.js
[10:44:57] Using gulpfile ~\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp\gulpfile_parallel.js
[10:44:57] Starting 'default'...
[10:44:57] Starting 'task1'...
[10:44:57] Starting 'task2'...
First task is being completed
[10:44:57] Finished 'task1' after 1.31 ms
Second task is being completed
[10:44:57] Finished 'task2' after 1.62 ms
[10:44:57] Finished 'default' after 4.04 ms
```

Рисунок 2 - Параллельное выполнение

Пункт б.

С помощью команды `npm i browser-sync` был установлен инструмент автоматизации BrowserSync, который синхронизирует изменения

в коде и автоматически обновляет страницу в браузере, что позволяет видеть результаты сразу после внесения изменений.

Код файла `gulpfile_reload.js` запускает сервер `BrowserSync`, который следит за изменениями в файлах проекта с помощью функции `watchFiles()`. Когда изменения происходят, вызывается функция `reload()`, которая перезагружает страницу в браузере. Задачи выполняются последовательно с помощью функции `series()`, которая последовательно запускает сервер, а затем начинается наблюдение за файлами.

Процесс работы `gulp` с `BrowserSync` на рис.3.

```
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp> gulp --gulpfile gulpfile_reload.js
[10:45:56] Using gulpfile ~\Desktop\учеба\5_сем\веб\лабы\lab3\task1_gulp\gulpfile_reload.js
[10:45:56] Starting 'default'...
[10:45:56] Starting 'browserSyncTask'...
[10:45:56] Finished 'browserSyncTask' after 16 ms
[10:45:56] Starting 'watchFiles'...
[Browsersync] Access URLs:
  -----
    Local: http://localhost:3000
    External: http://192.168.0.40:3000
  -----
    UI: http://localhost:3001
    UI External: http://192.168.0.40:3001
  -----
[Browsersync] Serving files from: ./
[10:46:35] Starting 'reload'...
[10:46:35] Finished 'reload' after 2.1 ms
[Browsersync] Reloading Browsers...
[10:46:47] Starting 'reload'...
[10:46:47] Finished 'reload' after 948 μs
[Browsersync] Reloading Browsers...
```

Рисунок 3 – Запуск задачи

Страница до изменений в файлах проекта (рис.4):

Страница для просмотра автоматической перезагрузки при изменении одного из контролируемых файлов

BrowserSync - утилита, которая автоматически перезагружает измененные файлы и страницы, синхронизирует навигацию между браузерами, что позволяет тестировать сайт сразу на нескольких устройствах

Рисунок 4 - Вид страницы до изменений

Страница после изменений в файлах проекта (рис.5):

Страница для просмотра автоматической перезагрузки при изменении одного из контролируемых файлов

BrowserSync - утилита, которая автоматически перезагружает измененные файлы и страницы, синхронизирует навигацию между браузерами, что позволяет тестировать сайт сразу на нескольких устройствах

Добавила текст и изменила цвет текста

Рисунок 5 - Вид страницы после изменений

Задание 2. Создание формы

Был создан файл `index.html` (рис.6), содержащий форму обратной связи, где пользователь вводит своё имя, фамилию, электронную почту, сообщение, выбирает варианты ответа на вопросы, а затем отправляет данные через метод POST на сервер для обработки скриптом `process.php` (рис.7). В файле `index.html` подключается внешний файл стилей CSS `styles.css`, который используется для оформления и стилизации элементов на странице.

```

<> index.html x process.php # styles.css
C: > Users > Polina > Desktop > учеба > 5_сем > веб > лабы > lab3 > task2_form > <> index.html > ...
1  <!DOCTYPE html>
2  <html lang="ru">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- для корректного отображения на мобильных устройствах -->
6      <title>Форма обратной связи</title>
7      <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10
11  <form action="process.php" method="POST">
12      <h1>Форма обратной связи</h1>
13
14      <label for="first_name">Имя:</label>
15      <input type="text" id="first_name" name="first_name" required><br><br>
16
17      <label for="last_name">Фамилия:</label>
18      <input type="text" id="last_name" name="last_name" required><br><br>
19
20      <label for="email">Электронная почта:</label>
21      <input type="email" id="email" name="email" required><br><br>
22
23      <label for="message">Сообщение обратной связи:</label><br>
24      <textarea id="message" name="message" rows="4" cols="50" required></textarea><br><br>
25
26      <label>Удовлетворены ли формой?</label><br>
27      <input type="radio" id="answer1" name="answer" value="да" required>
28      <label for="answer1">Да</label><br>
29      <input type="radio" id="answer2" name="answer" value="нет" required>
30      <label for="answer2">Нет</label><br><br>
31
32      <label>Что необходимо улучшить?</label><br>
33      <input type="checkbox" id="variant1" name="variants[]" value="Вариант1">
34      <label for="variant1">Вариант1</label><br>
35      <input type="checkbox" id="variant2" name="variants[]" value="Вариант2">
36      <label for="variant2">Вариант2</label><br>
37      <input type="checkbox" id="variant3" name="variants[]" value="Вариант3">
38      <label for="variant3">Вариант3</label><br><br>
39
40      <input type="submit" value="Отправить">
41  </form>
42
43  </body>
44  </html>

```

Рисунок 6 – Форма обратной связи

PHP предоставляет два основных метода для передачи данных между браузером и сервером: GET и POST.

Метод GET предназначен для запроса данных с сервера. Когда используется этот метод, параметры запроса передаются через URL в строке запроса, которая начинается с символа «?» и состоит из пар ключей и значений, разделенных знаком «&». Данные, передаваемые через него, отображаются в адресной строке браузера, а сам запрос может быть

закеширован и сохранен в закладках. С помощью данного метода можно совершать поиск информации. Длина URL ограничена (обычно до 2000 символов), что накладывает ограничения на объем передаваемых данных.

Метод POST используется для отправки данных на сервер. Данные передаются в теле HTTP-запроса, что делает их невидимыми для пользователя в адресной строке. POST не имеет строгих ограничений на объем передаваемых данных, что позволяет использовать его для отправки больших файлов. Запросы, выполненные с использованием POST, не кешируются и не сохраняются в закладках, т.к. они изменяют данные на сервере, а повторный отправленный запрос может привести к нежелательным последствиям (например, к созданию дубликатов)

```
<> index.html  process.php  # styles.css
C: > Users > Polina > Desktop > учеба > 5_сем > веб > лабы > lab3 > task2_form > process.php
1
2  <?php
3  if ($_SERVER["REQUEST_METHOD"] == "POST") {
4      // Получаем данные из формы
5      $first_name = htmlspecialchars($_POST['first_name']);
6      $last_name = htmlspecialchars($_POST['last_name']);
7      $email = htmlspecialchars($_POST['email']);
8      $message = htmlspecialchars($_POST['message']);
9      $answer = htmlspecialchars($_POST['answer']);
10     $variants = isset($_POST['variants']) ? $_POST['variants'] : [];
11
12     // Выводим полученные данные
13     echo "<h2>Полученные данные обратной связи:</h2>";
14     echo "<p><strong>Имя:</strong> $first_name</p>";
15     echo "<p><strong>Фамилия:</strong> $last_name</p>";
16     echo "<p><strong>Электронная почта:</strong> $email</p>";
17     echo "<p><strong>Сообщение:</strong> $message</p>";
18     echo "<p><strong>Удовлереы ли формой:</strong> $answer</p>";
19
20     if (!empty($variants)) {
21         echo "<p><strong>Ваши варианты:</strong></p>";
22         echo "<ul>";
23         foreach ($variants as $variants) {
24             echo "<li>" . ucfirst($variants) . "</li>";
25         }
26         echo "</ul>";
27     } else {
28         echo "<p><strong>Варианты не выбраны.</strong></p>";
29     }
30 } else {
31     echo "Форма не была отправлена.";
32 }
33 ?>
34
```

Рисунок 7 – Обработка формы

Для установки php с сайта <https://windows.php.net/download> был скачан интерпретатор php Thread Safe. После распаковки архива была проверена установка (рис.8).

```
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task2_form> php -v
PHP 8.4.1 (cli) (built: Nov 20 2024 11:13:29) (ZTS Visual C++ 2022 x64)
Copyright (c) The PHP Group
Zend Engine v4.4.1, Copyright (c) Zend Technologies
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task2_form> |
```

Рисунок 8 - Проверка установки php

Запуск встроенного php сервера с помощью команды `php -S localhost:8000` (рис.9):

```
PS C:\Users\Polina\Desktop\учеба\5_сем\веб\лабы\lab3\task2_form> php -S localhost:8000  
[Wed Nov 27 10:48:48 2024] PHP 8.4.1 Development Server (http://localhost:8000) started
```

Рисунок 9 - Запуск встроенного php сервера

Для проверки работы в строке браузера был введен адрес <http://localhost:8000>, после чего открылась форма (рис.10).

The screenshot shows a web browser window with the address bar displaying 'localhost:8000'. The page title is 'Форма обратной связи'. The form itself is centered and has a title 'Форма обратной связи'. It contains the following fields and controls:

- Имя:** A text input field containing 'Полина'.
- Фамилия:** A text input field containing 'Колтунова'.
- Электронная почта:** A text input field containing 'polinakoltunovapol@mail.ru'.
- Сообщение обратной связи:** A text area containing 'все супер!'.
- Удовлереыны ли формой?:** Two radio buttons, 'Да' (selected) and 'Нет'.
- Что необходимо улучшить?:** Three checkboxes, 'Вариант1' (unchecked), 'Вариант2' (checked), and 'Вариант3' (checked).
- Отправить:** A blue button at the bottom of the form.

Рисунок 10 - Форма обратной связи

После заполнения и отправки формы, php скрипт обрабатывает данные формы и выводится результат (рис.11)

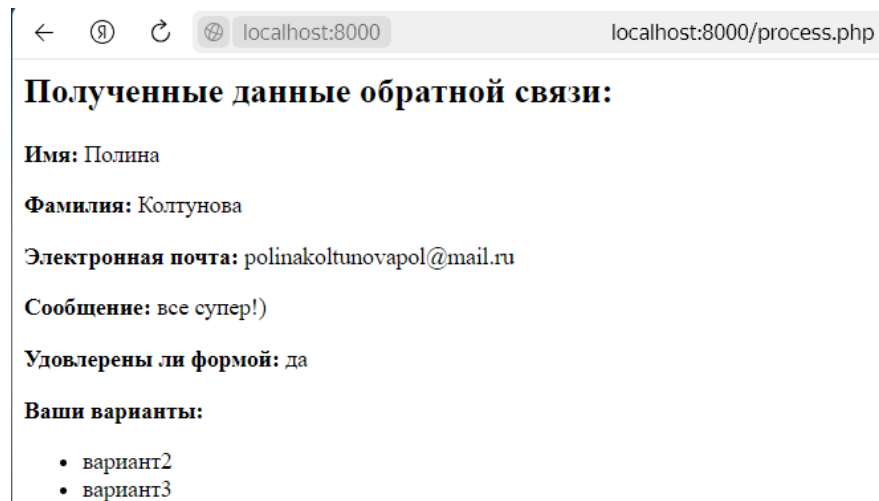


Рисунок 11 - Результат отправки формы

Задание 3. Работа с WordPress

1. Установка инструментария для локального сервера

Был установлен и настроен XAMPP для кроссплатформенной сборки локального веб-сервера с официального сайта <https://www.apachefriends.org/download.html> и скачана версия для Windows.

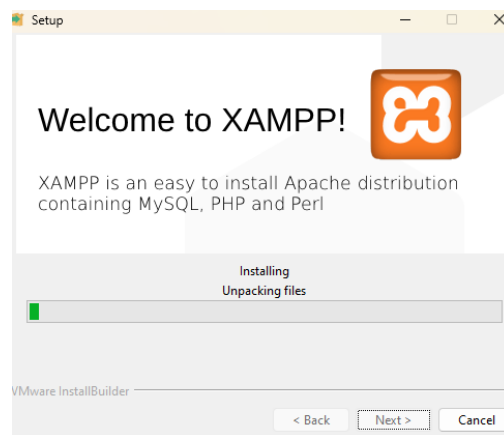


Рисунок 12 - Установка XAMPP

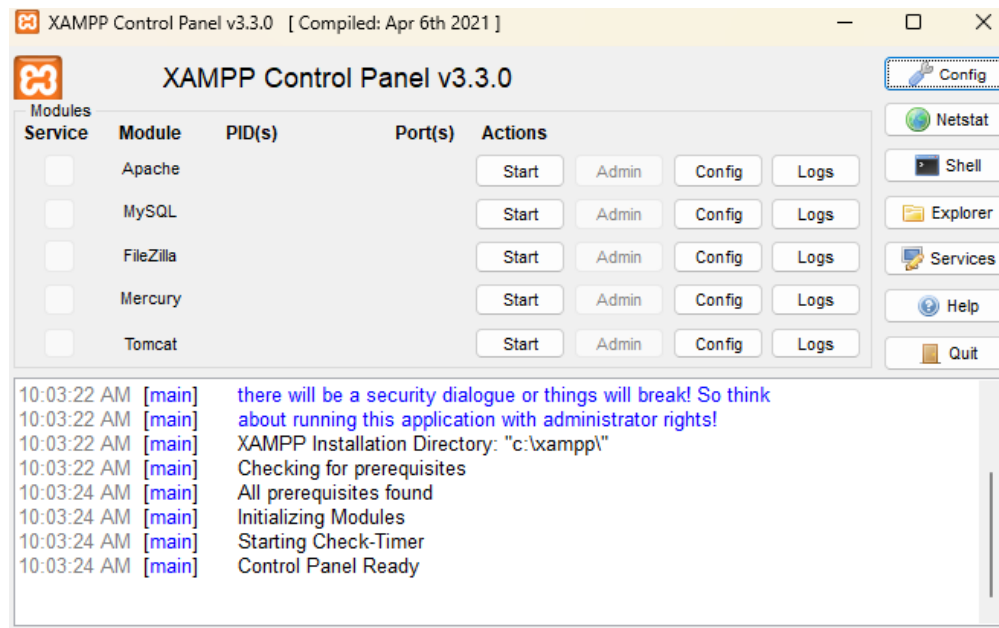


Рисунок 13 - Панель управления XAMPP



Welcome to XAMPP for Windows 8.2.12

Рисунок 14 - Проверка запуска XAMPP

2. Скачивание и установка WordPress

С официального сайта <https://ru.wordpress.org/> была скачана последняя версия. Скачанный файл был разархивирован в каталог `C:\xampp\htdocs` и переименован в папку `test.site`.

Затем в браузере, перейдя на `http://localhost/phpmyadmin`, была создана база данных `test_site` (рис.15).

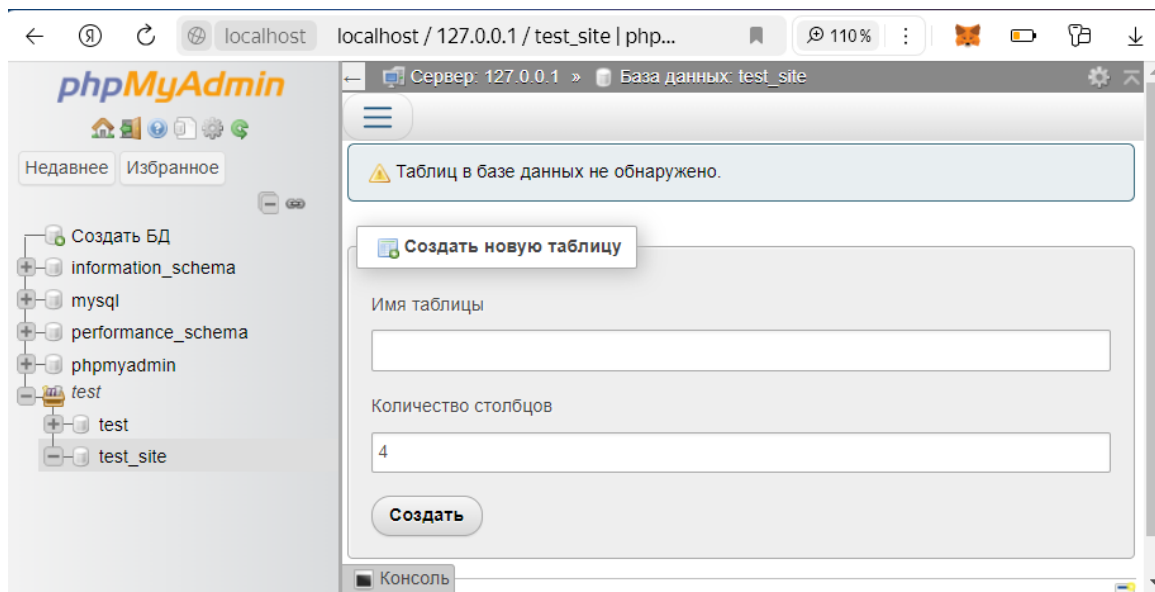


Рисунок 15 - Создание базы данных test_site

Далее на <http://localhost/test.site>, был настроен файл конфигурации с помощью внесения данных бд (название: test_site, имя пользователя: root, пароль: без, сервер бд: localhost) и закончена установка (рис. 16-18).

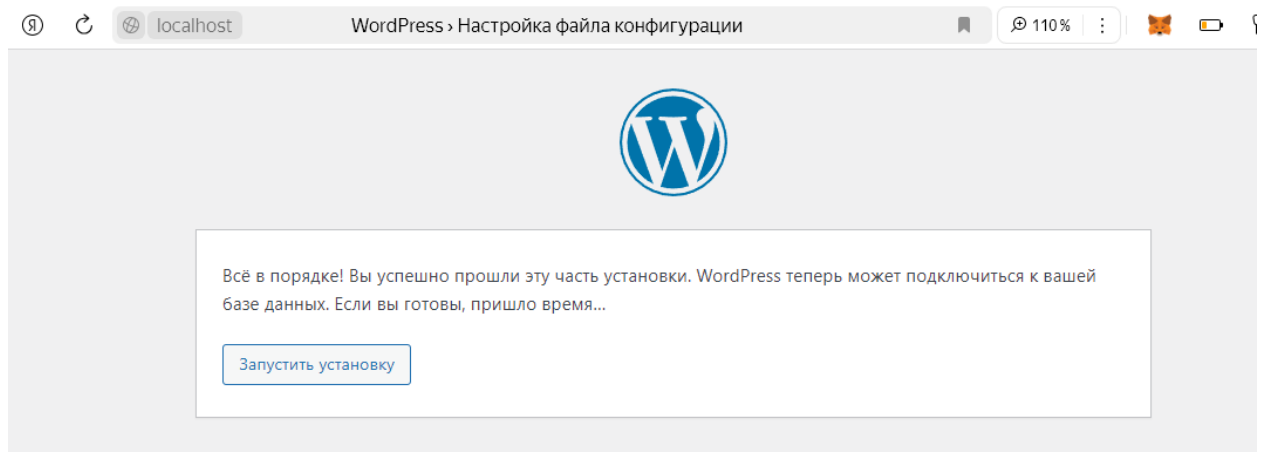


Рисунок 16 - настройка файла конфигурации

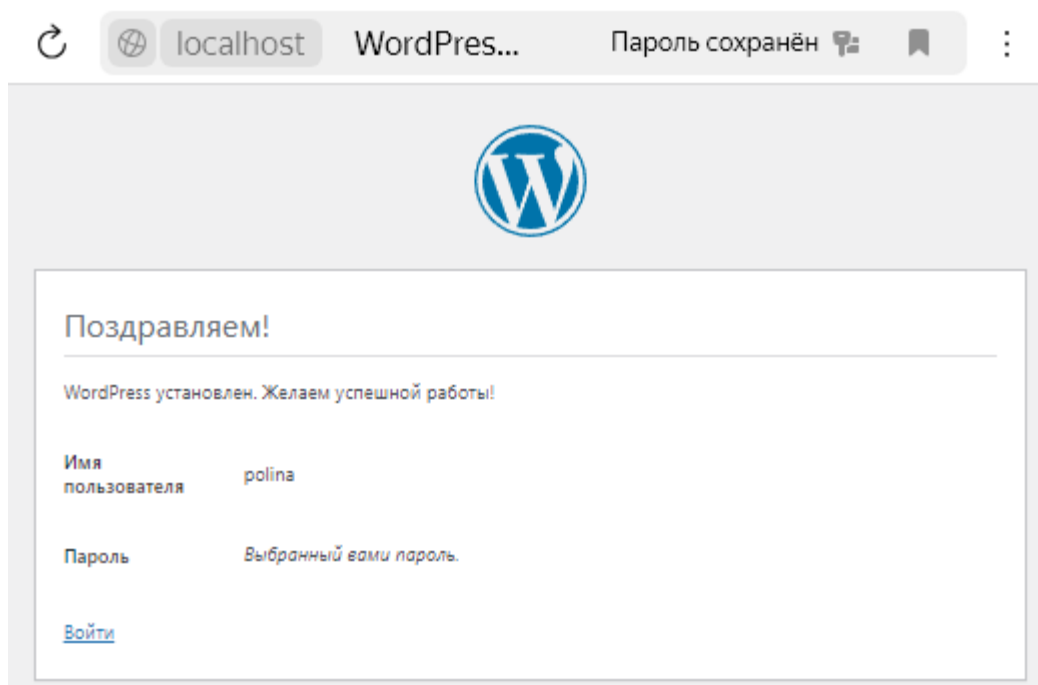


Рисунок 17 - Успешная установка WordPress

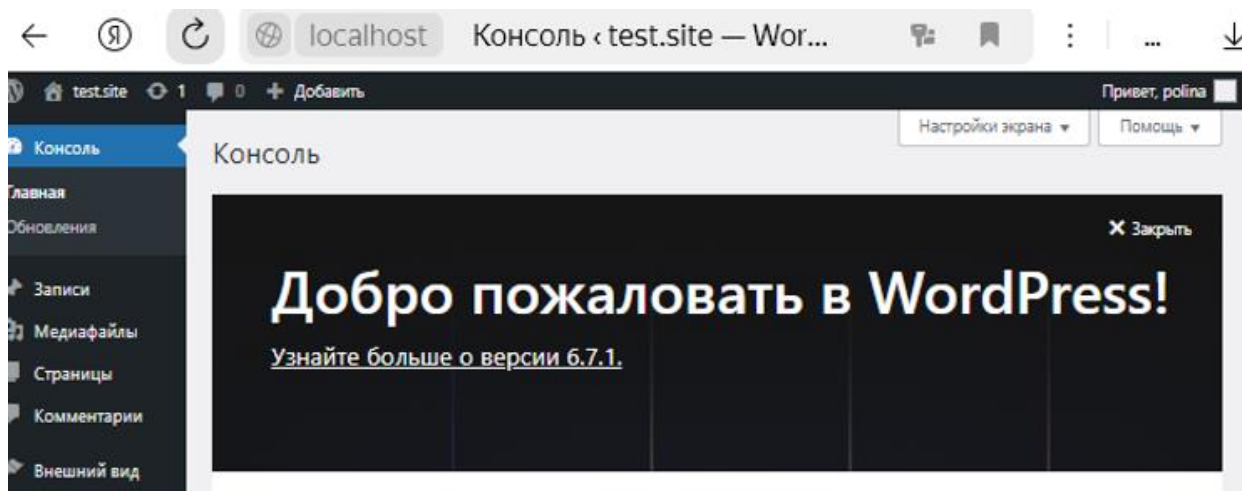
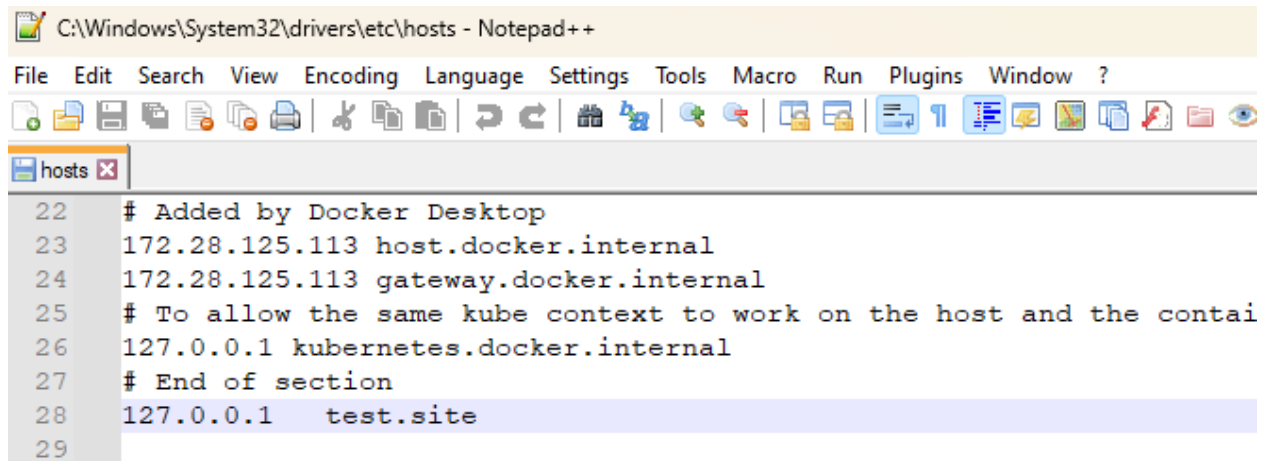


Рисунок 18 - Вход в WordPress

3. Настройка локального домена <http://test.site>

Был назначен IP-адрес 127.0.0.1 для домена test.site. Для это был отредактирован файл hosts, содержащий информацию о ip-адресах и их доменах с правами администратора (путь к файлу:

C:\Windows\System32\drivers\etc\hosts). В него добавлена строка 127.0.0.1 test.site (рис.19).

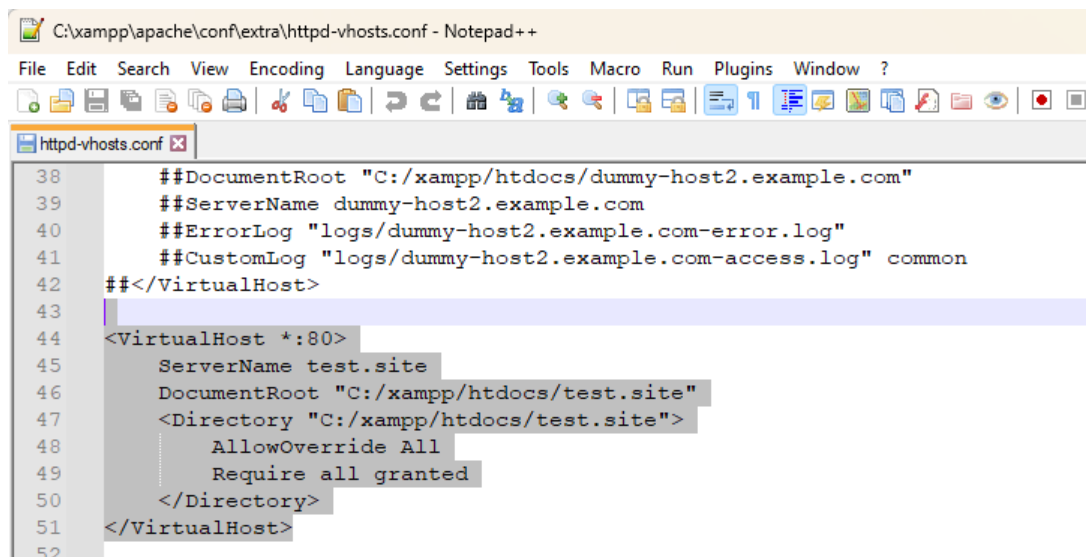
A screenshot of the Notepad++ application window. The title bar reads 'C:\Windows\System32\drivers\etc\hosts - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar contains various icons for file operations and editing. The 'hosts' file is open, showing the following content:

```
22 # Added by Docker Desktop
23 172.28.125.113 host.docker.internal
24 172.28.125.113 gateway.docker.internal
25 # To allow the same kube context to work on the host and the containe
26 127.0.0.1 kubernetes.docker.internal
27 # End of section
28 127.0.0.1 test.site
29
```

The line '127.0.0.1 test.site' is highlighted in blue.

Рисунок 19 - Редактирование файла hosts

Настройка Apache (рис.20). В файле конфигурации Apache (путь к файлу: C:\xampp\apache\conf\extra\httpd-vhosts.conf) был добавлен виртуальный хост

A screenshot of the Notepad++ application window. The title bar reads 'C:\xampp\apache\conf\extra\httpd-vhosts.conf - Notepad++'. The menu bar and toolbar are the same as in the previous screenshot. The 'httpd-vhosts.conf' file is open, showing the following content:

```
38 ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
39 ##ServerName dummy-host2.example.com
40 ##ErrorLog "logs/dummy-host2.example.com-error.log"
41 ##CustomLog "logs/dummy-host2.example.com-access.log" common
42 #</VirtualHost>
43
44 <VirtualHost *:80>
45     ServerName test.site
46     DocumentRoot "C:/xampp/htdocs/test.site"
47     <Directory "C:/xampp/htdocs/test.site">
48         AllowOverride All
49         Require all granted
50     </Directory>
51 </VirtualHost>
52
```

The lines from 44 to 51 are highlighted in blue.

Рисунок 20 - Добавление виртуального хоста

После перезапуска Apache, в браузере был введен локальный домен <http://test.site>, и открыт настроенный портал WordPress (рис.21).

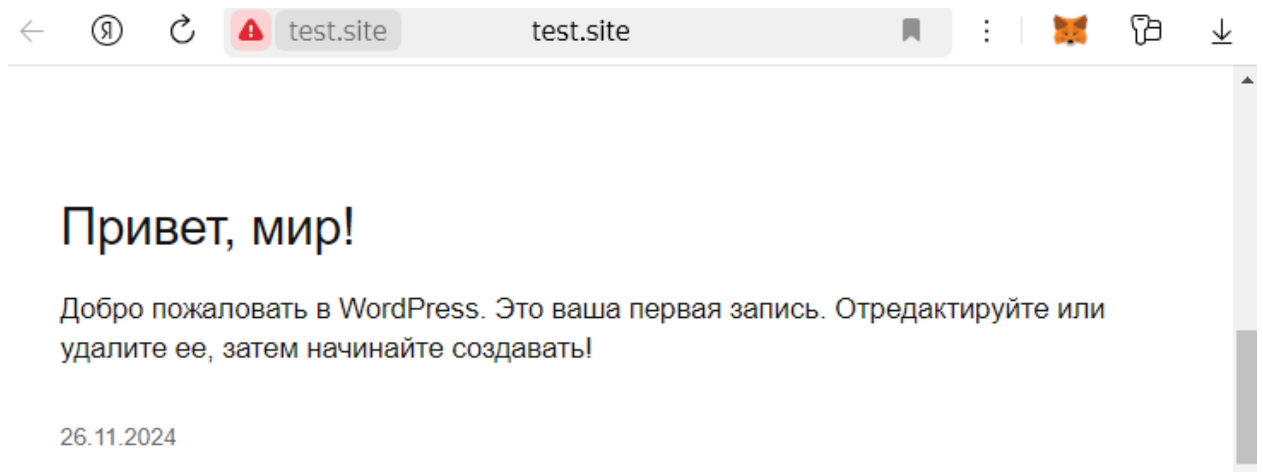


Рисунок 21 - Настроенный портал на WordPress

ВЫВОД

Цель была достигнута. Были получены навыки работы с gulp. Были созданы задачи для последовательного и параллельного выполнения. Был изучен сервер BrowserSync, с помощью которого настроена автоматическая перезагрузка при изменении одного из контролируемых файлов проекта. Была создана форма для отправки информации обратной связи, которая обрабатывается php скриптом. Установлен инструментальный XAMPP для локального сервера, скачан и установлен WordPress. Был настроен портал <http://test.site>, что при вводе данного адреса, отвечает настроенный портал.