

**Санкт-Петербургский Национальный Исследовательский Университет
Информационных технологий, механики и оптики**

Отчет

Дисциплина: Web-программирование

Лабораторная работа 3.

Выполнил: Сарычев С.И.

Группа № K3321

Проверила: Марченко Е.В.

Санкт-Петербург

2024

Оглавление

Введение	3
Ход работы	4
1. Работа с Gulp.....	4
2. Создание формы.	7
3. Установка движка.	14
Вывод	18

Введение

Цель работы: изучить инструменты gulp, wordpress, познакомиться с php, MAMP.

Ход работы

1. Работа с Gulp.

В прошлой лабораторной работе уже было изучено как устанавливать gulp и инициализировать проект, поэтому просто повторим все это, но также установим и browser-sync, он поможет нам отслеживать изменения в файлах и автоматически отображать новые файлы в браузере (рисунок 1)

```
MacBook-Pro-Sergei:lab3 annafomicheva$ npm install --save-dev gulp browser-sync
npm WARN lab3@1.0.0 No description
npm WARN lab3@1.0.0 No repository field.

+ gulp@5.0.0
+ browser-sync@3.0.3
added 116 packages from 241 contributors, updated 5 packages and audited 260 packages in 9.708s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

MacBook-Pro-Sergei:lab3 annafomicheva$
```

Рисунок 1 – Установка browser-sync

Далее добавим задачи в Gulpfile, для работы с CSS, HTML, JS (рисунок 2).

```
// Задача для работы с HTML
function htmlTask() {
  console.log('задача 1')
  return gulp.src(paths.html).pipe(gulp.dest('dist')).pipe(browserSync.stream())
}

// Задача для работы с CSS
function cssTask() {
  console.log('задача2')
  return gulp
    .src(paths.css)
    .pipe(gulp.dest('dist/styles'))
    .pipe(browserSync.stream())
}

// Задача для работы с JS
function jsTask() {
  console.log('задача3')
  return gulp
    .src(paths.js)
    .pipe(gulp.dest('dist/scripts'))
    .pipe(browserSync.stream())
}
```

Рисунок 2 – Добавление задач

После этого протестируем последовательное и параллельное выполнение задач, для последовательного выполнения используем `gulp.series`, а для параллельного `gulp.parallel`, разницу в выполнении можно отследить в консоли (при выполнении задачи он выводит сообщение) результат представлен на рисунках 3 и 4.

```
[MacBook-Pro-Sergei:lab3 annafomicheva$ gulp
[02:24:07] Using gulpfile ~/Desktop/WEB/lab3/gulpfile.js
[02:24:07] Starting 'default'...
[02:24:07] Starting 'htmlTask'...
задача 1
[02:24:07] Finished 'htmlTask' after 26 ms
[02:24:07] Starting 'cssTask'...
задача2
[02:24:07] Finished 'cssTask' after 8.99 ms
[02:24:07] Starting 'jsTask'...
задача3
[02:24:07] Finished 'jsTask' after 7.27 ms
[02:24:07] Starting 'serve'...
```

Рисунок 3 – Последовательное выполнение

```
[02:27:05] Using gulpfile ~/Desktop/WEB/lab3/gulpfile.js
[02:27:05] Starting 'default'...
[02:27:05] Starting 'htmlTask'...
[02:27:05] Starting 'cssTask'...
[02:27:05] Starting 'jsTask'...
задача 1
задача2
задача3
[02:27:05] Finished 'htmlTask' after 36 ms
[02:27:05] Finished 'cssTask' after 40 ms
[02:27:05] Finished 'jsTask' after 40 ms
[02:27:05] Starting 'serve'...
```

Рисунок 4 – Параллельное выполнение

Далее по заданию реализуем task для автоматической перезагрузки при изменении одного из контролируемых файлов проекта, код taskа представлен на рисунке 5.

```
// Task для запуска сервера и автообновления
function serve() {
  browserSync.init({
    server: {
      baseDir: 'dist',
    },
  })

  gulp.watch(paths.html, htmlTask)
  gulp.watch(paths.css, cssTask)
  gulp.watch(paths.js, jsTask)
}
```

Рисунок 5 – Task для автоматической перезагрузки

После проверим что при изменении css файла, цвет шрифта автоматически меняется на веб странице (рисунок 6)

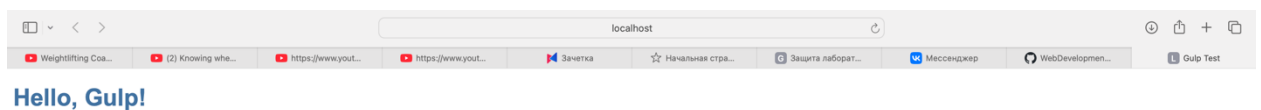
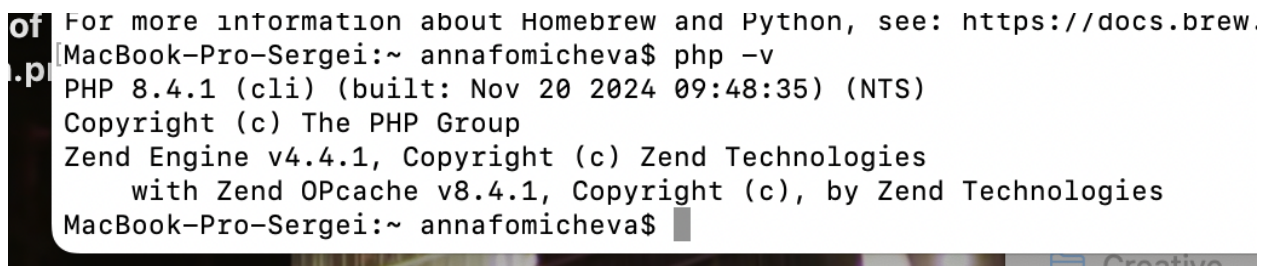


Рисунок 6 – Проверка автоматической перезагрузки

2. Создание формы.

В начале установим с помощью homebrew php (рисунок 7)

A terminal window screenshot showing the command 'php -v' being executed. The output displays the PHP version as 8.4.1 (cli) and provides copyright information for The PHP Group and Zend Technologies. The prompt indicates the user is 'anna' on a 'MacBook-Pro-Sergei' machine.

```
of For more information about Homebrew and Python, see: https://docs.brew.  
.p MacBook-Pro-Sergei:~ annafomicheva$ php -v  
PHP 8.4.1 (cli) (built: Nov 20 2024 09:48:35) (NTS)  
Copyright (c) The PHP Group  
Zend Engine v4.4.1, Copyright (c) Zend Technologies  
    with Zend OPcache v8.4.1, Copyright (c), by Zend Technologies  
MacBook-Pro-Sergei:~ annafomicheva$
```

Рисунок 7 – Проверка версии php

Далее создаем файл index.html и заполняем его как на рисунке 8.

```
<> index.html × 🐘 feedback_handler.php # styles.css
task2 > <> index.html > 📁 html > 📁 body > 📁 form#feedbackForm > 🏷 label
2   <html lang="en">
3   <head>
8   </head>
9   <body>
10  <h1>Форма обратной связи</h1>
11  <!-- Добавляем возможность выбора метода отправки -->
12  <form id="feedbackForm" action="feedback_handler.php" method="GET">
13    <h2>Основные данные</h2>
14    <label for="first_name">Имя:</label>
15    <input type="text" id="first_name" name="first_name" required>
16
17    <label for="last_name">Фамилия:</label>
18    <input type="text" id="last_name" name="last_name" required>
19
20    <label for="email">Электронная почта:</label>
21    <input type="email" id="email" name="email" required>
22
23    <label for="feedback">Ваше сообщение:</label>
24    <textarea id="feedback" name="feedback" rows="5" cols="30" required></textarea>
25
26    <h2>Капибара или бобёр:</h2>
27    <label for="capu_choise">
28    <input type="radio" id="capu_choise" name="capu_beaver" value="capu" required>
29    Капибара</label>
30
31    <label for="beaver_choise">
32    <input type="radio" id="beaver_choise" name="capu_beaver" value="beaver">
33    Бобёр</label>
34
35    <h2>Ваши интересы:</h2>
36    <label for="interest_1">
37    <input type="checkbox" id="interest_1" name="interests[]" value="sports">
38    Спорт</label>
39    <label for="interest_2">
40    <input type="checkbox" id="interest_2" name="interests[]" value="music">
41    Музыка</label>
42    <label for="interest_3">
43    <input type="checkbox" id="interest_3" name="interests[]" value="coding">
44    Программирование</label>
45
46    <h2>Выберите метод отправки:</h2>
47    <label for="method_get">
48    <input type="radio" id="method_get" name="send_method" value="GET" checked>
49    GET</label>
50    <label for="method_post">
51    <input type="radio" id="method_post" name="send_method" value="POST">
52    POST</label>
53
54    <button type="submit">Отправить</button>
55  </form>
56
57  <script>
58    // Скрипт для изменения метода отправки формы
59    const form = document.getElementById('feedbackForm');
60    const sendMethodRadios = document.querySelectorAll('input[name="send_method"]');
61
62    sendMethodRadios.forEach((radio) => {
63      radio.addEventListener('change', (event) => {
64        form.method = event.target.value;
65      });
66    });
67  </script>
68 </body>
69 </html>
```

Рисунок 8 – index.html

После создадим styles.css (рисунок 9)

task2 > css > # styles.css > ...

```
1  /* Общие стили для страницы */
2  body {
3      font-family: Arial, sans-serif;
4      background-color: #f4f4f9; /* Светлый фон страницы */
5      margin: 0;
6      padding: 0;
7      color: #333; /* Цвет текста */
8  }
9
10 /* Контейнер формы */
11 form {
12     width: 500px; /* Ширина формы */
13     margin: 50px auto; /* Центрируем форму */
14     padding: 30px; /* Внутренние отступы */
15     background-color: #ffffff; /* Белый фон */
16     border: 1px solid #ddd; /* Тонкая рамка */
17     border-radius: 10px; /* Сглаженные углы */
18     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Тень */
19 }
20
21 /* Заголовок формы */
22 h1 {
23     text-align: center; /* Центрируем заголовок */
24     font-size: 24px; /* Размер шрифта */
25     color: #007BFF; /* Яркий синий цвет */
26     margin-bottom: 20px; /* Отступ снизу */
27 }
28
29 /* Поля ввода */
30 input[type="text"],
31 input[type="email"],
32 textarea {
33     width: 100%; /* Поля растягиваются на всю ширину */
34     margin-bottom: 15px; /* Отступ между элементами */
35     padding: 10px; /* Внутренние отступы */
36     font-size: 16px; /* Размер текста */
37     border: 1px solid #ccc; /* Тонкая рамка */
38     border-radius: 5px; /* Сглаженные углы */
39     box-sizing: border-box; /* Учитываем padding */
40 }
41
42 /* Радиокнопки и чекбоксы */
43 input[type="radio"],
44 input[type="checkbox"] {
45     margin-right: 10px; /* Отступ справа от кнопки */
46 }
47
48 /* Подписи к элементам */
49 label {
50     font-size: 16px;
51     margin-bottom: 10px;
52     display: block; /* Каждая подпись на отдельной строке */
53 }
54
55 /* Кнопка отправки */
56 button {
57     width: 100%; /* Кнопка занимает всю ширину формы */
58     padding: 10px;
59     background-color: #007BFF; /* Яркий синий фон */
60     color: #ffffff; /* Белый текст */
61     border: none; /* Убираем рамку */
62     border-radius: 5px; /* Сглаженные углы */
63     font-size: 18px; /* Размер текста */
64     cursor: pointer; /* Указатель при наведении */
65 }
```

Рисунок 9 – styles.css

И наконец создадим feedback_handler.php (рисунок 10)

```
task2 > feedback_handler.php
1  <?php
2  // Проверяем метод запроса
3  if ($_SERVER["REQUEST_METHOD"] === "GET" || $_SERVER["REQUEST_METHOD"] === "POST") {
4      // Определяем, откуда поступили данные
5      $data = ($_SERVER["REQUEST_METHOD"] === "POST") ? $_POST : $_GET;
6
7      // Получение данных из формы
8      $first_name = htmlspecialchars($data["first_name"] ?? "Не указано");
9      $last_name = htmlspecialchars($data["last_name"] ?? "Не указано");
10     $email = htmlspecialchars($data["email"] ?? "Не указано");
11     $feedback = htmlspecialchars($data["feedback"] ?? "Не указано");
12     $capy_beaver = htmlspecialchars($data["capy_beaver"] ?? "Не указан");
13     $interests = isset($data["interests"]) ? $data["interests"] : [];
14
15     // Вывод данных
16     echo "<h1>Спасибо за обратную связь!</h1>";
17     echo "<p><strong>Имя:</strong> $first_name</p>";
18     echo "<p><strong>Фамилия:</strong> $last_name</p>";
19     echo "<p><strong>Электронная почта:</strong> $email</p>";
20     echo "<p><strong>Сообщение:</strong> $feedback</p>";
21     echo "<p><strong>Капибара или бобёр:</strong> $capy_beaver</p>";
22
23     if (!empty($interests)) {
24         echo "<p><strong>Ваши интересы:</strong> " . implode(", ", $interests) . "</p>";
25     } else {
26         echo "<p><strong>Ваши интересы:</strong> Не указаны</p>";
27     }
28
29     echo "<p><strong>Метод отправки:</strong> " . $_SERVER["REQUEST_METHOD"] . "</p>";
30 } else {
31     echo "<h1>Ошибка!</h1>";
32     echo "<p>Неверный метод запроса.</p>";
33 }
34 ?>
35
```

Рисунок 10 - feedback_handler.php

Проверим работу веб страницы, создадим локальный сервер с помощью php -S localhost:8000, и перейдем по советуемому адресу в браузере (рисунок 11)

Основные данные

Имя:

Фамилия:

Электронная почта:

Ваше сообщение:

Капибара или бобёр:

☒ Капибара
☐ Бобёр

Ваши интересы:

☒ Спорт
☒ Музыка
☐ Программирование

Выберите метод отправки:

☐ GET
☒ POST

Рисунок 11 – Веб страница

Протестируем методы GET и POST, на рисунке 12 мы используем метод GET, а на рисунке 13 метод POST, это также можно увидеть в инструментах разработчика.

Спасибо за обратную связь!

Имя: ваы
Фамилия: ваы
Электронная почта: sdfs@lol.ru
Сообщение: fsdf
Капибара или бобёр: capy
Ваши интересы: sports, music
Метод отправки: GET

Network Tab:

Name	Headers	Payload	Preview	Response	Initi
feedback_handler.php?first_name=ваы&last_name=ваы&email=sdfs%40lol.ru&feedback=fsdf&capy_beaver=capy&interests=sports,music&method=GET	General Request URL: http://localhost:8000/feedback_handler.php?first_name=ваы&last_name=ваы&email=sdfs%40lol.ru&feedback=fsdf&capy_beaver=capy&interests=sports,music&method=GET Request Method: GET Status Code: 200 OK Remote Address: [::1]:8000 Referrer Policy: strict-origin-when-cross-origin				

Рисунок 12 – Метод GET

Спасибо за обратную связь!

Имя: ваы

Фамилия: ваы

Электронная почта: sdf@lol.ru

Сообщение: fsdf

Капибара или бобёр: capy

Ваши интересы: sports, music

Метод отправки: POST

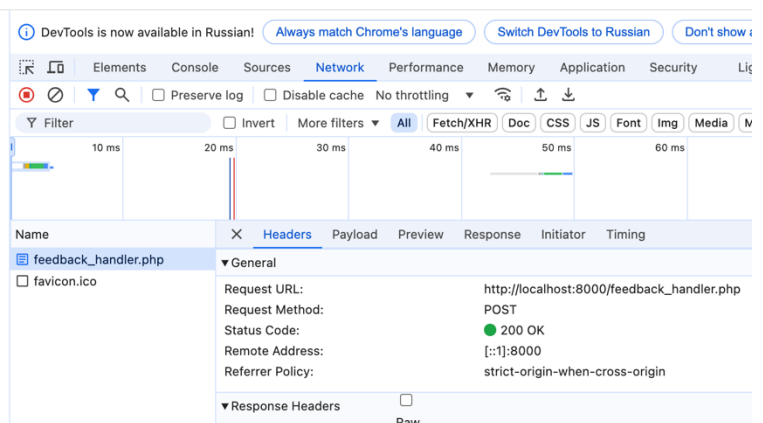


Рисунок 13 – Метод POST

Методы GET и POST используются в HTTP-протоколе для отправки данных с клиента на сервер. Оба метода являются основными средствами для взаимодействия между веб-страницами и сервером при отправке данных.

Метод GET отправляет данные в URL-строке запроса (в адресной строке браузера). Этот метод используется для получения информации с сервера, например, для запросов на поиск или при переходе по страницам.

Особенности метода GET:

- Данные в URL: Все данные передаются как часть URL в строке запроса (например: `https://example.com?name=John&email=john@example.com`).
- Ограничения на размер данных: Поскольку данные передаются через URL, существует ограничение на размер данных (обычно около 2000 символов, но это зависит от браузера и сервера).
- Безопасность: Данные, передаваемые через GET, видны в адресной строке и могут быть перехвачены, что делает метод менее безопасным для передачи конфиденциальной информации.
- Кэширование: Запросы, использующие GET, могут быть кэшированы браузером или прокси-серверами, что позволяет ускорить повторные запросы.

- **Идемпотентность:** GET-запросы считаются идемпотентными, что означает, что повторный запрос с теми же параметрами не должен изменять состояние на сервере.

Метод POST отправляет данные в теле запроса, что позволяет передавать более сложные и объемные данные. Этот метод чаще всего используется для отправки данных, например, при заполнении формы на сайте.

Особенности метода POST:

- **Данные в теле запроса:** Данные не отображаются в адресной строке браузера, что делает метод более безопасным для передачи конфиденциальной информации.
- **Отсутствие ограничений по размеру данных:** В отличие от GET, в POST-запросе нет таких строгих ограничений на размер передаваемых данных.
- **Безопасность:** Данные, передаваемые через POST, не видны в URL и передаются в теле запроса, что делает метод более безопасным для отправки чувствительной информации (например, паролей).
- **Не кэшируется:** Запросы POST не кэшируются браузером.
- **Не идемпотентность:** Запросы POST могут изменять состояние на сервере, например, добавлять новые записи в базу данных, поэтому их повторное выполнение может привести к различным результатам.

3. Установка движка.

Для начала необходимо скачать MAMP, так как лабораторная выполняется на MacOS, после этого необходимо также скачать wordpress с официального сайта и распаковать в папку MAMP/htdocs, далее запускаем MAMP и создаем локальный сервер (рисунок 14).

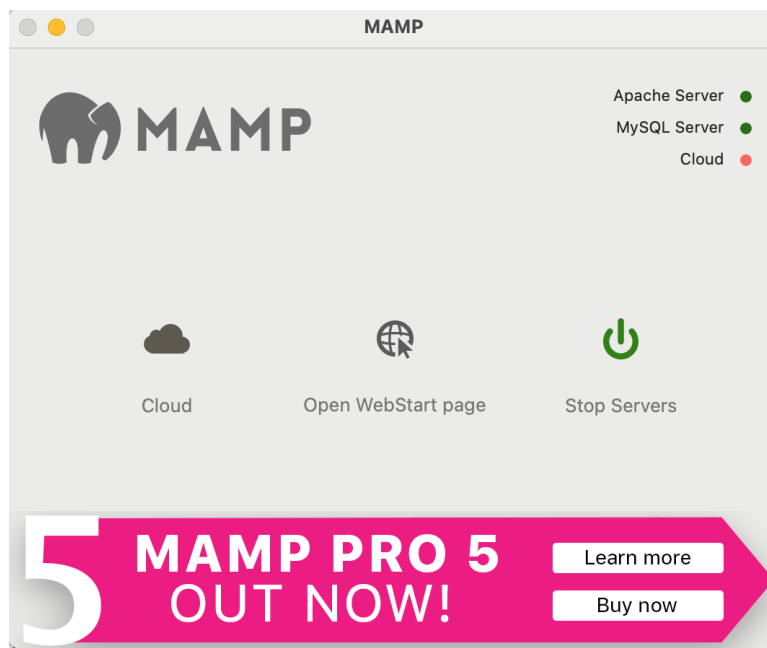


Рисунок 14 – MAMP

После этого создадим базу данных с помощью phpMyAdmin (рисунок 15).

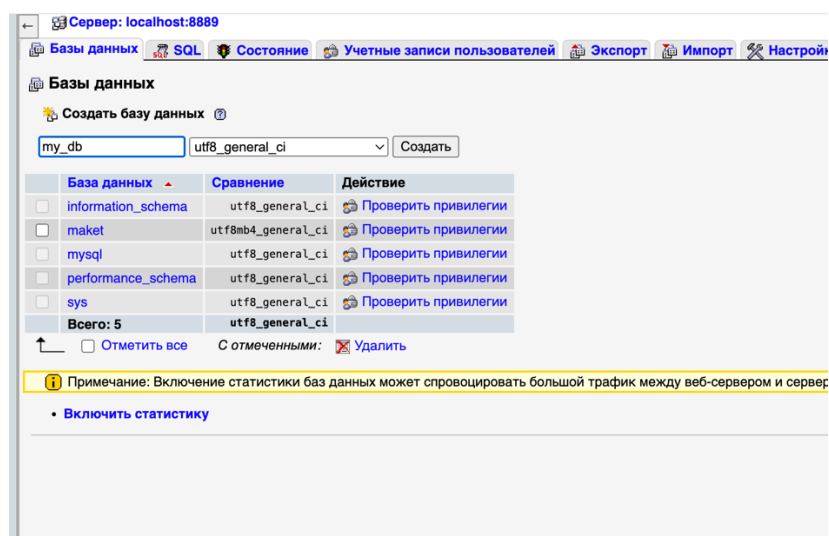
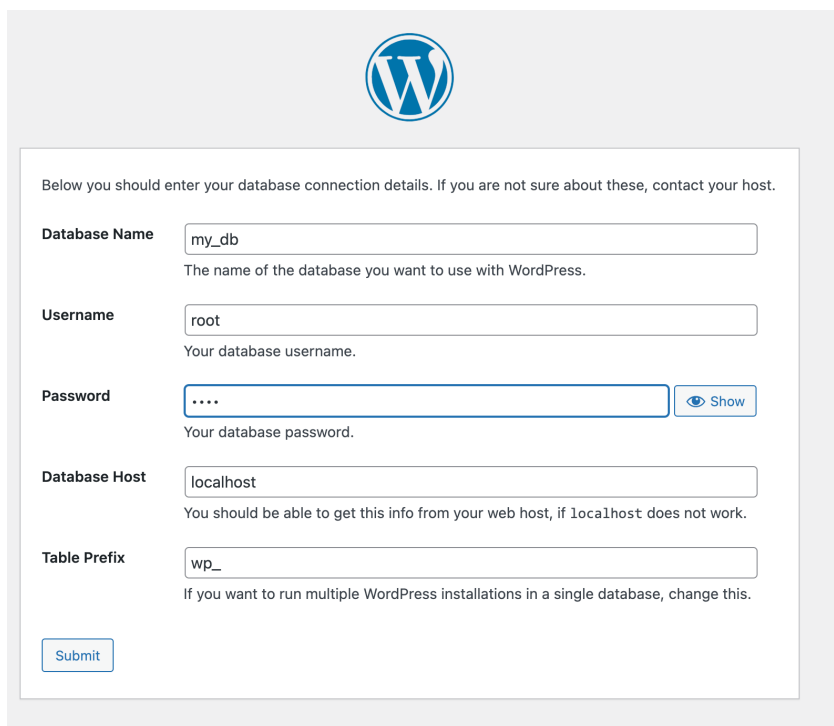


Рисунок 15 – Создание базы данных

Далее переходим по адресу <http://localhost:8888/wordpress>, и начинаем процесс установки WordPress (рисунок 16).



The screenshot shows the WordPress installation database configuration screen. At the top is the WordPress logo. Below it, a message states: "Below you should enter your database connection details. If you are not sure about these, contact your host." The form contains five input fields: "Database Name" with the value "my_db", "Username" with the value "root", "Password" with masked characters "...." and a "Show" button, "Database Host" with the value "localhost", and "Table Prefix" with the value "wp_". Each field has a descriptive label below it. At the bottom left is a "Submit" button.

Рисунок 16 – Установка WordPress

После заполнения еще одну форму и попадаем в панель управления (рисунок 17)

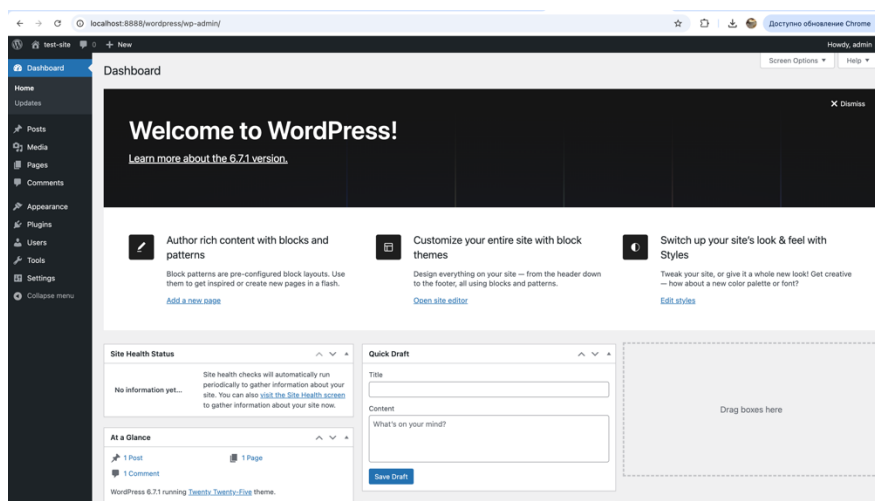


Рисунок 17 – Админ панель WordPress

Для того чтобы адрес <http://test.site> указывал на локальный сервер, необходимо отредактировать файл `hosts` на macOS. Этот файл связывает доменные имена с IP-адресами (рисунок 18).

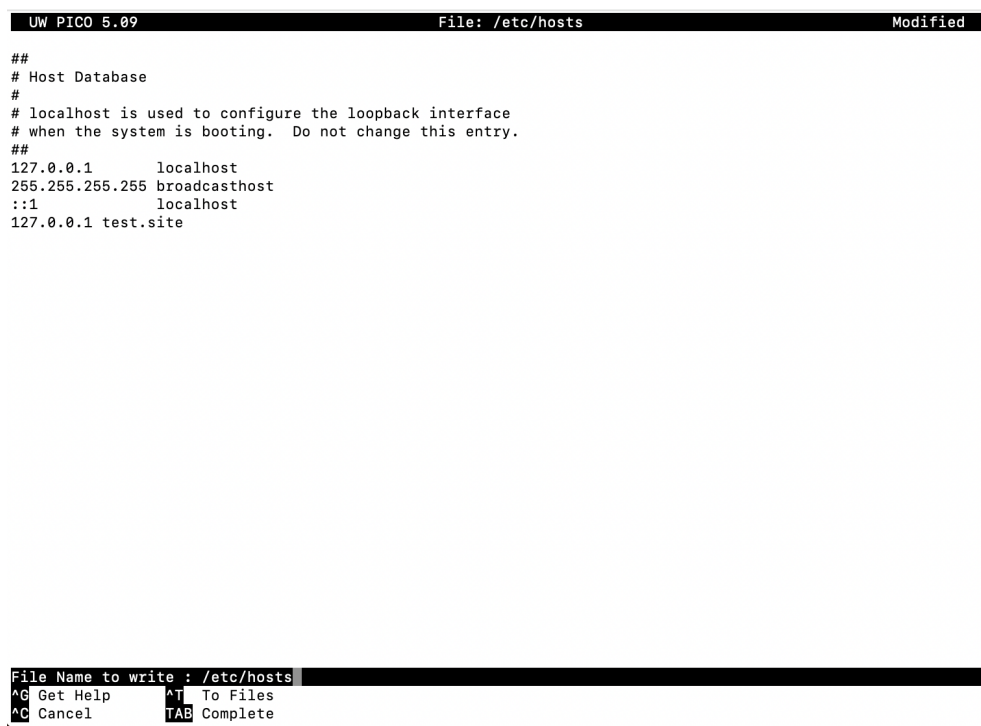


Рисунок 18 – Изменение файла `hosts`

Далее переходим по адресу <http://test.site>, и видим следующую страницу (рисунок 19).

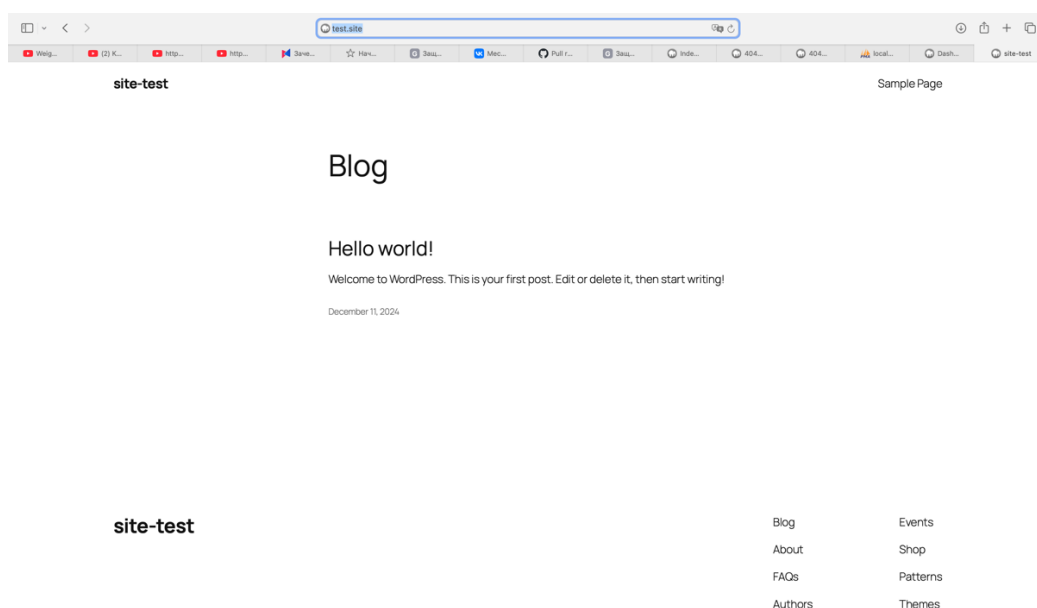


Рисунок 19 - <http://test.site>

После устанавливаем новую тему в настройках WordPress (рисунок 20).

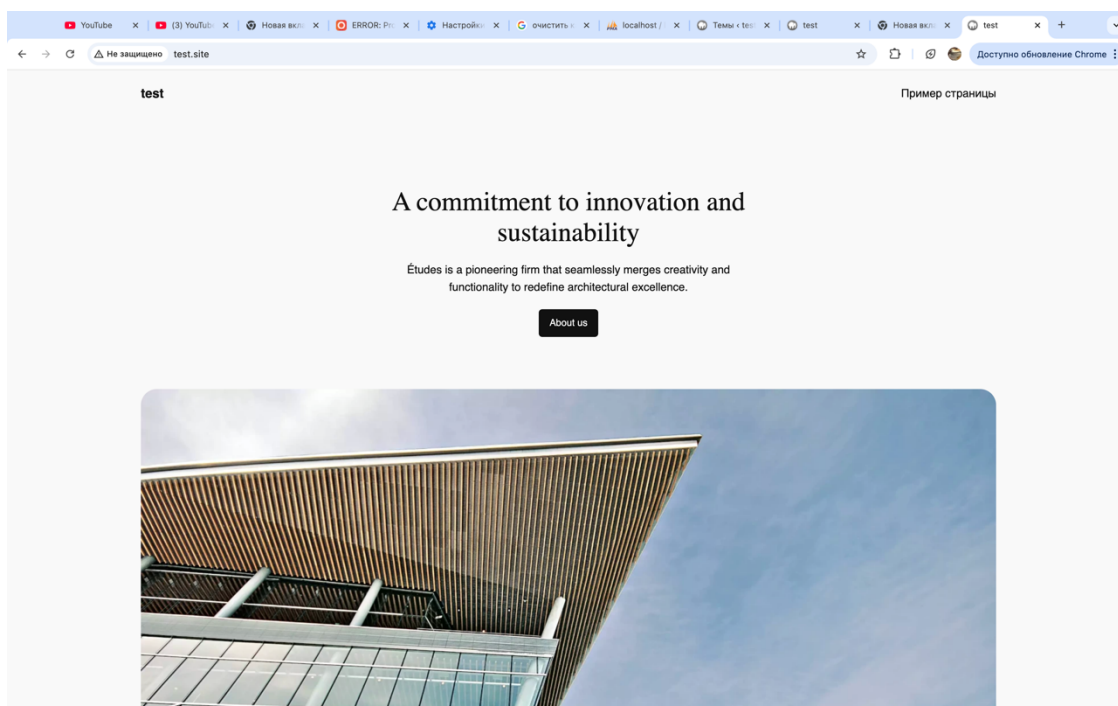


Рисунок 20 – Новая тема

Вывод

В рамках данной работы были изучен инструмент gulp, а также создан простейший сайт с использованием инструментов MAMP и Wordpress