

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

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ 3

По дисциплине Web-программирование

Обучающийся Надери Мариам

Факультет Факультет инфокоммуникационных технологий

Группа K3321

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	<u>11.12.2024</u> (дата)	<u> </u> (подпись)	<u>Надери М.Ш.</u> (Ф.И.О.)
Руководитель	<u> </u> (дата)	<u> </u> (подпись)	<u>Марченко Е.В.</u> (Ф.И.О.)

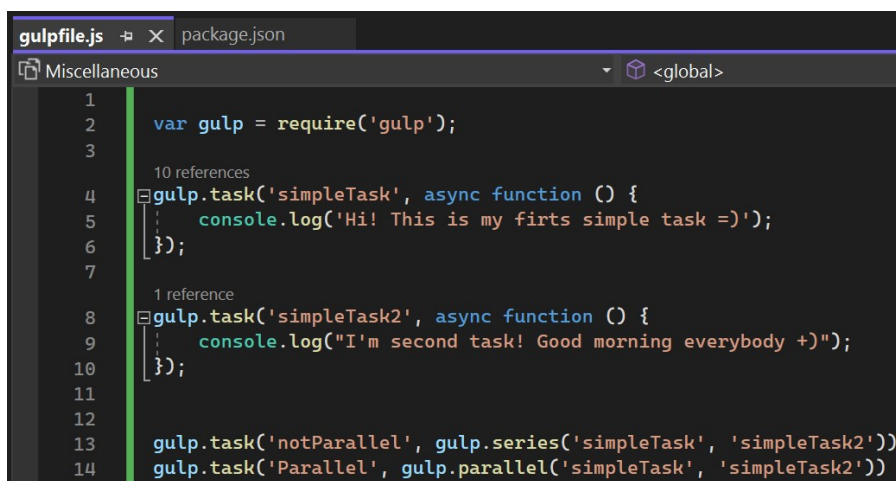
Санкт-Петербург
2024 г.

1 задание

1.1 Создать два таска – настроить на последовательное и параллельное выполнение

Перед началом было создано окружение (как в лабораторной работе 2).

Далее для этого задания было создано два таска, которые выводят в консоль текстовую строку (Рис. 1.1). С помощью `gulp.series` и `gulp.parallel` были созданы еще два таска для последовательного и параллельного запуска предыдущих двух тасков соответственно.



```
1
2 var gulp = require('gulp');
3
4 gulp.task('simpleTask', async function () {
5   console.log('Hi! This is my first simple task =)');
6 });
7
8 gulp.task('simpleTask2', async function () {
9   console.log("I'm second task! Good morning everybody +)");
10 });
11
12 gulp.task('notParallel', gulp.series('simpleTask', 'simpleTask2'))
13 gulp.task('Parallel', gulp.parallel('simpleTask', 'simpleTask2'))
```

Рисунок 1.1

После в `gulpfile.js` были добавлены команды «notP» и «P» для запуска тасков последовательно и параллельно соответственно (Рис. 1.2).

Результат запуска можно увидеть на рисунке 1.3, на нем видно, что при последовательном запуске второй таск начинается после завершения первого, а при параллельном второй таск начинается до завершения первого.

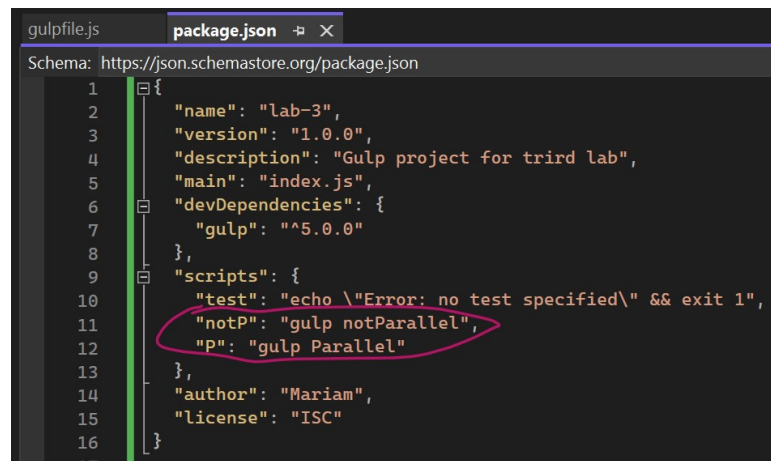


Рисунок 1.2

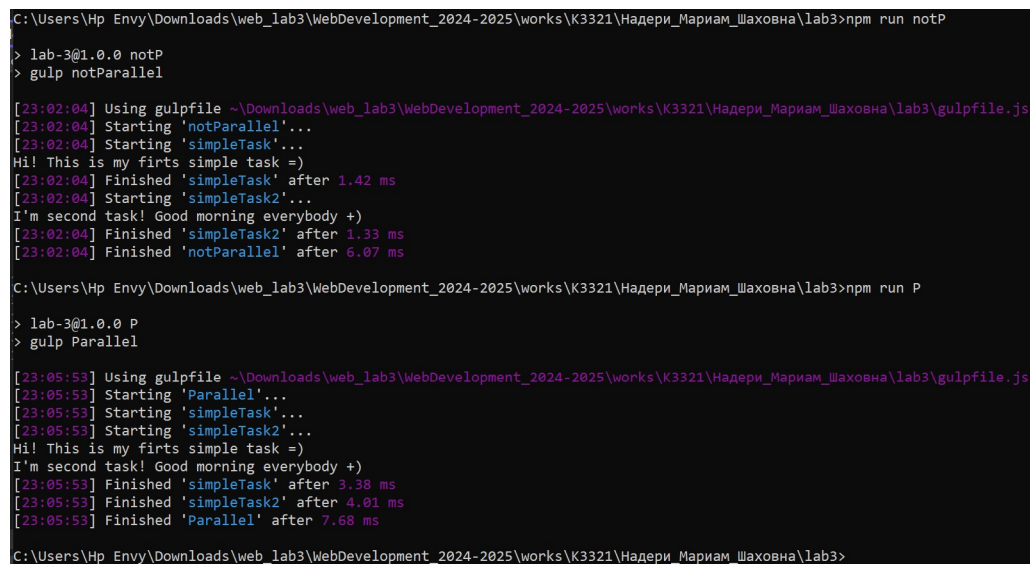


Рисунок 1.3

1.2 Настроить отображение файлов проекта в браузере и автоматическую перезагрузку при изменении одного из контролируемых файлов проекта

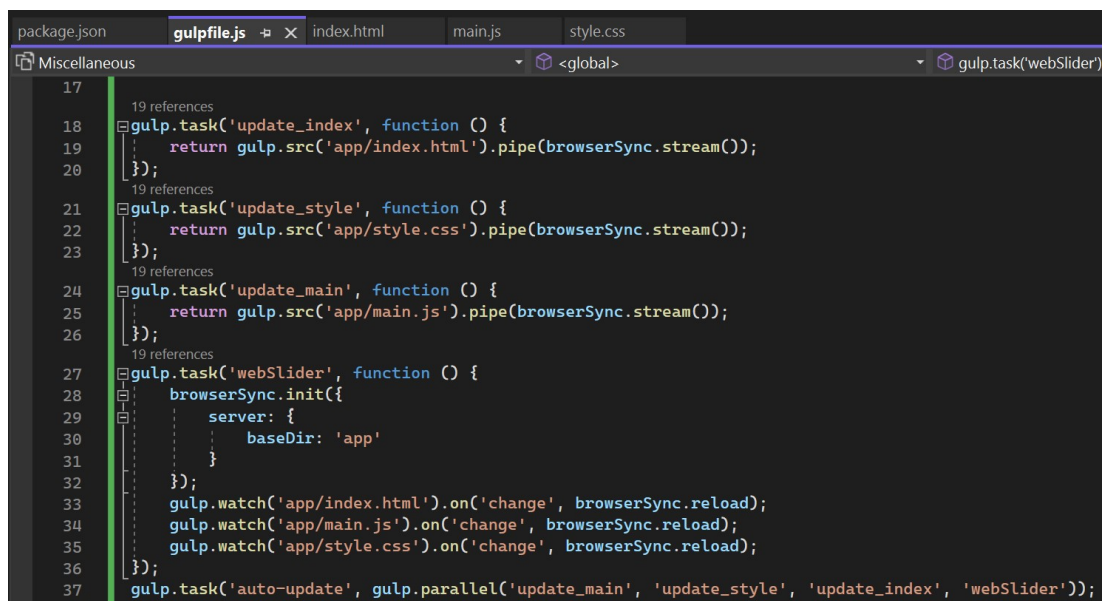
Для выполнения этого задания был использован слайдер веб страниц, реализованный в лабораторной работе №2.

После копирования нужных файлов из прошлой лабораторной, было написано три таска (Рис. 1.4), каждый из которых отвечает за обновление одно-

го из трех файлов (index.html, style.css, main.js). Таск для запуска слайдера остался с прошлой лабораторной.

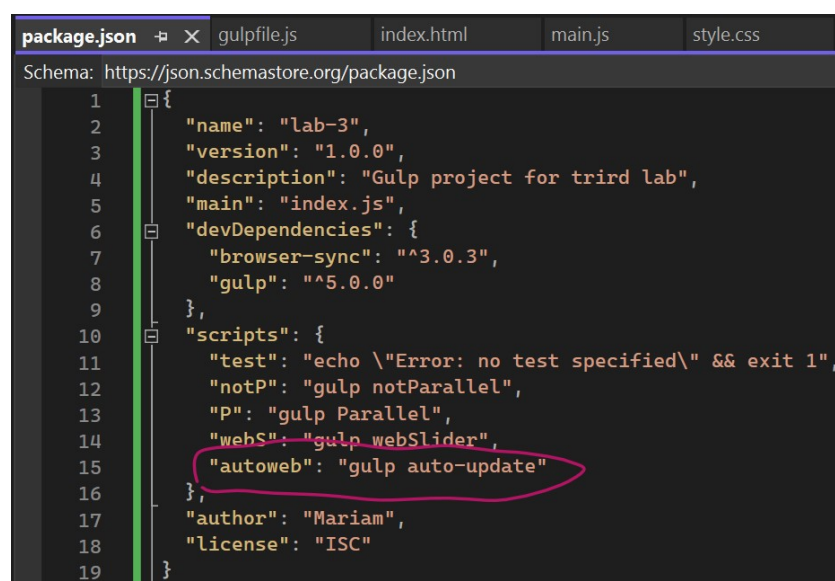
Самая главная строчка – последняя. Там создан таск, который запускает четыре вышеупомянутых таска параллельно. Это позволяет обновлениям в файлах сразу автоматически отображаться на веб страничке.

В gulpfile.js была добавлена команда autoweb для запуска нужного таска (Рис. 1.5).



```
17
18 19 references
19 gulp.task('update_index', function () {
20   return gulp.src('app/index.html').pipe(browserSync.stream());
21 });
22 19 references
23 gulp.task('update_style', function () {
24   return gulp.src('app/style.css').pipe(browserSync.stream());
25 });
26 19 references
27 gulp.task('update_main', function () {
28   return gulp.src('app/main.js').pipe(browserSync.stream());
29 });
30 19 references
31 gulp.task('webSlider', function () {
32   browserSync.init({
33     server: {
34       baseDir: 'app'
35     }
36   });
37   gulp.watch('app/index.html').on('change', browserSync.reload);
38   gulp.watch('app/main.js').on('change', browserSync.reload);
39   gulp.watch('app/style.css').on('change', browserSync.reload);
40 });
41 gulp.task('auto-update', gulp.parallel('update_main', 'update_style', 'update_index', 'webSlider'));
```

Рисунок 1.4



```
1 {
2   "name": "lab-3",
3   "version": "1.0.0",
4   "description": "Gulp project for trird lab",
5   "main": "index.js",
6   "devDependencies": {
7     "browser-sync": "^3.0.3",
8     "gulp": "^5.0.0"
9   },
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1",
12    "notP": "gulp notParallel",
13    "P": "gulp Parallel",
14    "webS": "gulp webSlider",
15    "autoweb": "gulp auto-update"
16  },
17  "author": "Mariam",
18  "license": "ISC"
19 }
```

Рисунок 1.5

На рисунках 1.6 и 1.7 можно увидеть результат. На рисунке 1.6 вид веб странички сразу после запуска, а на рисунке 1.7 после изменений (удаление зоны ввода интервала и изменение цвета фона), можно заметить, что в консоли отобразилось изменение css-файла и обновление веб странички.

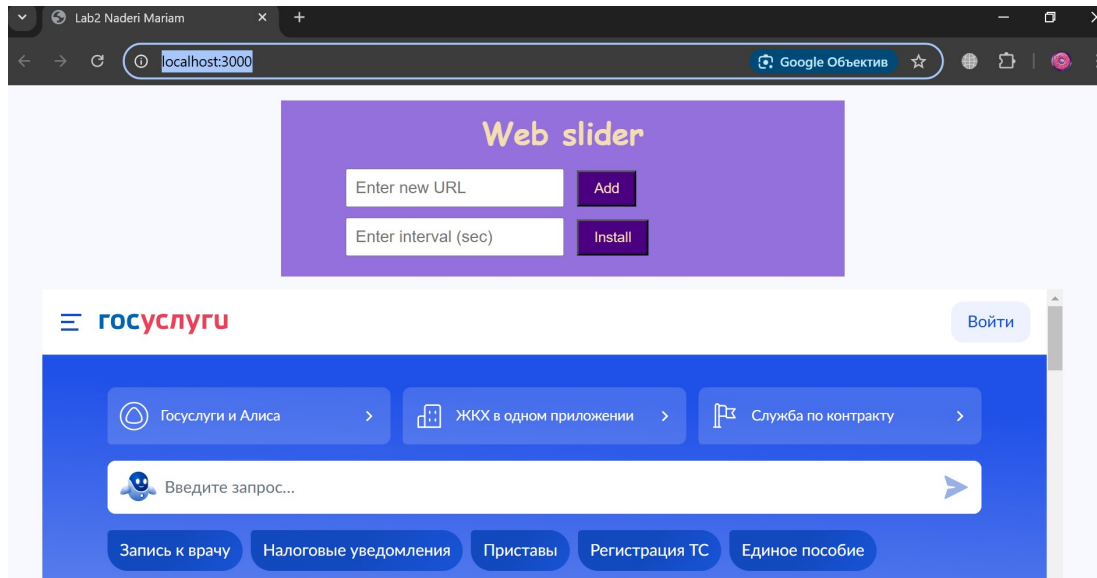


Рисунок 1.6

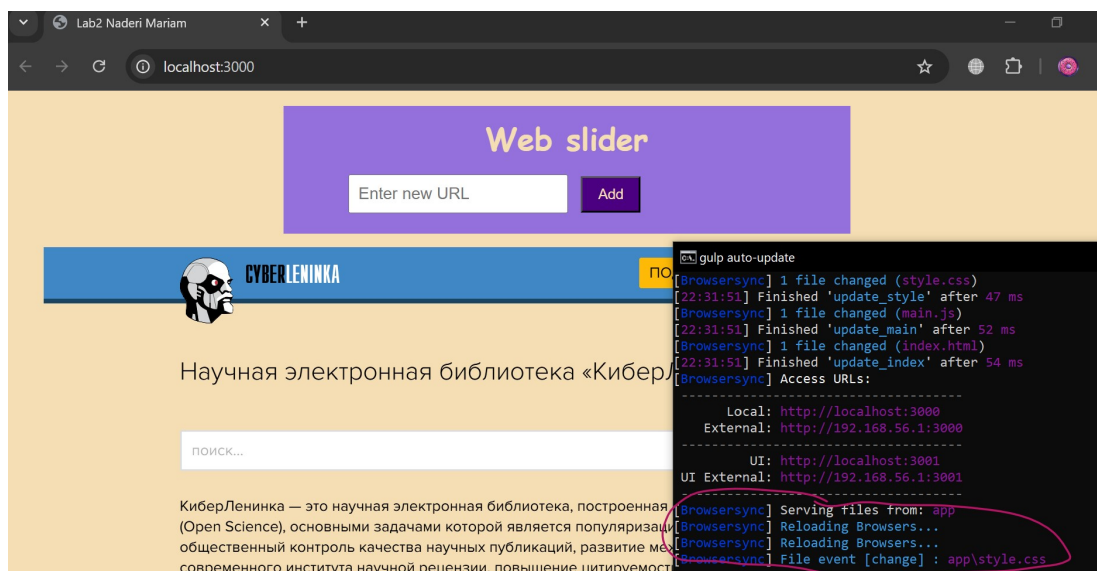


Рисунок 1.7

2 задание

Во втором задании нужно было создать форму для отправки информации по обратной связи от пользователя сайта используя php.

Для работы с php необходим веб-сервер, поэтому был установлен XAMPP. На рисунке 2.1 показан запуск сервера.

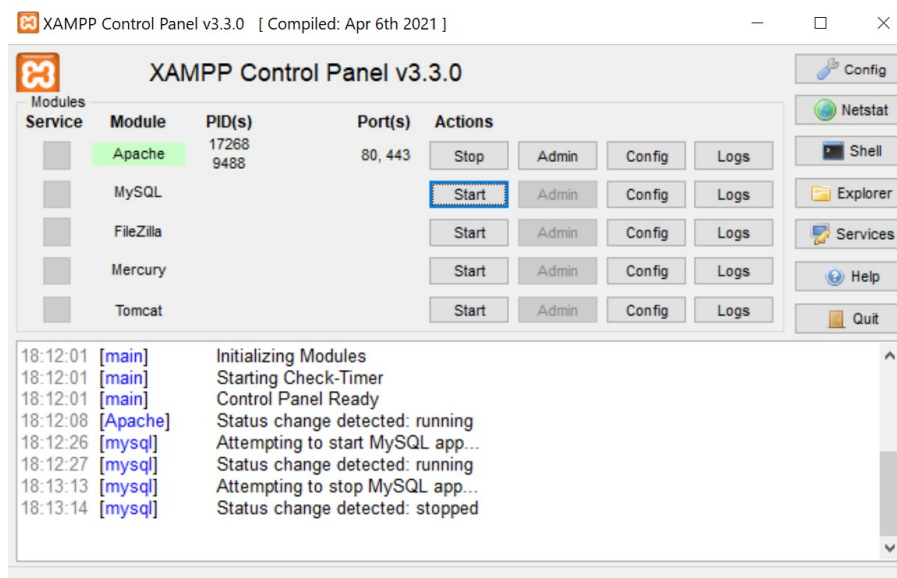


Рисунок 2.1

Чтобы все работало, скрипты должны находиться в папке C:\xampp\htdocs. В ней были созданы три файла: feedback.html, main.php и style.css.

В первом (Рис. 2.2-2.3) была реализована разметка веб-странички (намечены кнопки, поля для ввода, радиокнопки, чекбоксы). К нему были подключены два других файла. style.css нужен для задания стилизового оформления.


```

style.css  feedback.html  main.php
1  <!DOCTYPE html>
2
3  <html lang="en">
4  <head>
5    <meta charset="UTF-8">
6    <title>Lab3 task2 Naderi Mariam</title>
7    <link rel="stylesheet" href="style.css">
8  </head>
9
10 <body>
11 <header>
12 <h1>Feedback form</h1>
13 <form action="main.php" method="POST">
14   <label for="last_name">Your last name</label><br>
15   <input type="text" id="last_name" name="last_name" required><br><br>
16
17   <label for="first_name">Your first name</label><br>
18   <input type="text" id="first_name" name="first_name" required><br><br>
19
20   <label for="age">Your age</label><br>
21   <input type="number" id="age" name="age" required><br><br>
22
23   <label>Do you like a chocolate?</label><br>
24   <input type="radio" id="yes_radio" name="radio" value="Yes" required>
25   <label for="yes_radio">Yes</label><br>
26   <input type="radio" id="no_radio" name="radio" value="No">
27   <label for="no_radio">No</label><br><br>
28

```

Рисунок 2.2

```

style.css  feedback.html  main.php
28
29   <label>What kind of chocolate do you prefer:</label><br>
30   <input type="checkbox" id="dark" name="type_choc[]" value="Dark">
31   <label for="dark">Dark</label><br>
32   <input type="checkbox" id="milk" name="type_choc[]" value="Milk">
33   <label for="milk">Milk</label><br>
34   <input type="checkbox" id="white" name="type_choc[]" value="White">
35   <label for="white">White</label><br><br>
36
37   <label for="fav_choc">Write your favorite chocolate</label><br>
38   <textarea id="fav_choc" name="fav_choc" rows="4" cols="50" required></textarea><br><br>
39
40   <button id="submit">Submit Post</button>
41   <button id="get_method" formaction="main.php" formmethod="GET">Submit Get</button>
42
43 </form>
44 </header>
45 </body>
46 </html>
47

```

Рисунок 2.3

Последний же файл (Рис. 2.4-2.5) – main.php – осуществляет всю работу, в нем прописаны реализации с помощью методов post и get.

```

style.css  feedback.html  main.php
1  <?php
2
3  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
4      $first_name = htmlspecialchars($_POST['first_name']);
5      $last_name = htmlspecialchars($_POST['last_name']);
6      $age = htmlspecialchars($_POST['age']);
7      $fav_choc = htmlspecialchars($_POST['fav_choc']);
8      $radio = htmlspecialchars($_POST['radio']);
9      $type_choc = isset($_POST['type_choc']) ? $_POST['type_choc'] : [];
10
11      echo "<h1>Thanks for your feedback!</h1>";
12      echo "<p>First name: $first_name</p>";
13      echo "<p>Last name: $last_name</p>";
14      echo "<p>Age: $age</p>";
15      echo "<p>Favorite chocolate: $fav_choc</p>";
16      echo "<p>Like or not chocolate: $radio</p>";
17
18      if (!empty($type_choc)) {
19          echo "<p> You prefer: " . implode(" ", $type_choc) . "</p>";} else {
20          echo "<p> You didn't choose chocolate which you prefer";}
21  } elseif ($_SERVER['REQUEST_METHOD'] === 'GET') {
22      $first_name = htmlspecialchars($_GET['first_name']);
23      $last_name = htmlspecialchars($_GET['last_name']);
24      $age = htmlspecialchars($_GET['age']);
25      $fav_choc = htmlspecialchars($_GET['fav_choc']);
26      $radio = htmlspecialchars($_GET['radio']);
27      $type_choc = isset($_GET['type_choc']) ? $_GET['type_choc'] : [];
28

```

Рисунок 2.4

```

style.css  feedback.html  main.php
28
29      echo "<h1>Thanks for your feedback!</h1>";
30      echo "<p>First name: $first_name</p>";
31      echo "<p>Last name: $last_name</p>";
32      echo "<p>Age: $age</p>";
33      echo "<p>Favorite chocolate: $fav_choc</p>";
34      echo "<p>Like or not chocolate: $radio</p>";
35
36      if (!empty($type_choc)) {
37          echo "<p> You prefer: " . implode(" ", $type_choc) . "</p>";} else {
38          echo "<p> You didn't choose chocolate which you prefer";}
39  } else {
40      echo "<p>Error</p>";
41  }
42  ?>
43

```

Рисунок 2.5

На рисунке 2.6 показано как выглядит сама форма. У нее есть две кнопки: одна для применения метода post, а другая для get.

Результат post-метода показан на рисунке 2.7, а get-метода на рисунке 2.8. Можно увидеть, что при использовании get-метода в url можно увидеть значение переменных, при post-методе такого не происходит.

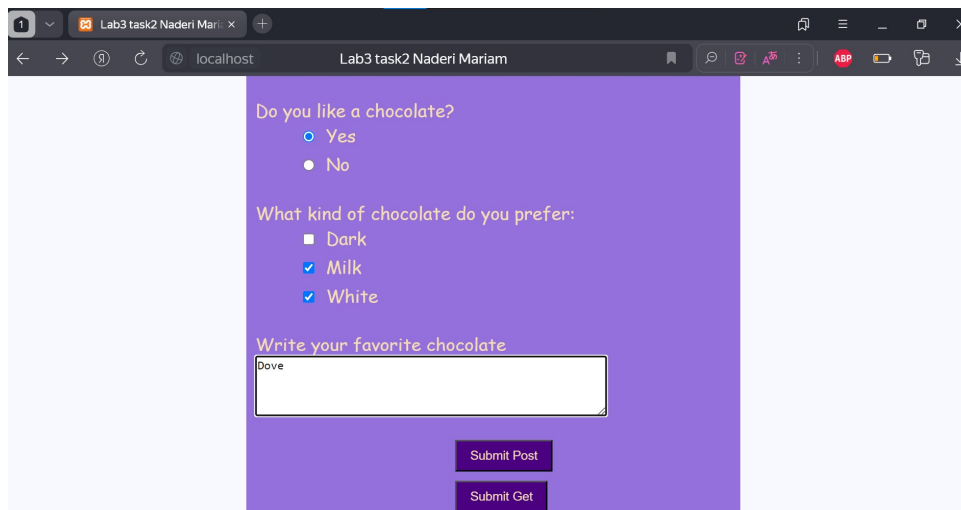
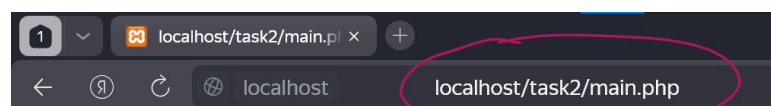


Рисунок 2.6



Thanks for your feedback!

First name: Мариам

Last name: Надери

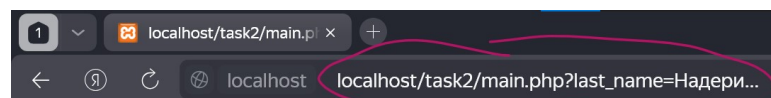
Age: 7

Favorite chocolate: Dove

Like or not chocolate: Yes

You prefer: Milk, White

Рисунок 2.7 — Результат post-метода



Thanks for your feedback!

First name: Мариам

Last name: Надери

Age: 7

Favorite chocolate: Dove

Like or not chocolate: Yes

You prefer: Milk, White

Рисунок 2.8 — Результат get-метода

3 задание

В третьем задании надо было установить инструментарий для отладки проектов, wordpress и настроить переход на портал `http://test.site`.

Для работы был использован установленный для второго задания ХАМРР. Снова запускаем модуль apacheи еще модуль mysql (Рис. 3.1).



Рисунок 3.1

Далее в папку `C:\xampp\htdocs\wordpress` был скачен движок wordpress.

После надо было настроить виртуальный хост для `http://test.site`. Для этого в файл `httpd-vhosts.conf` из папки `C:\xampp\apache\conf\extra` была добавлена следующая конфигурация (Рис. 3.2):

```
httpd-vhosts.conf  + X
43
44 <VirtualHost *:80>
45     ServerName test.site
46     DocumentRoot "C:/xampp/htdocs/wordpress"
47     <Directory "C:/xampp/htdocs/wordpress">
48         AllowOverride All
49         Require all granted
50     </Directory>
51 </VirtualHost>
```

Рисунок 3.2

В файл `hosts` из папки `C:\Windows\System32\drivers\etc` была добавлена следующая строка (Рис. 3.3):

```

C: > Windows > System32 > drivers > etc > hosts
22
23 172.28.64.102 host.docker.internal
24 172.28.64.102 gateway.docker.internal
25 # To allow the same kube context to work on the host
26 127.0.0.1 kubernetes.docker.internal
27 # End of section
28 127.0.0.1 test.site
29

```

Рисунок 3.3

Следующим этапом было создание базы данных. На портале <http://localhost/phpmyadmin/> была создана БД web_lab_3 (Рис. 3.4).

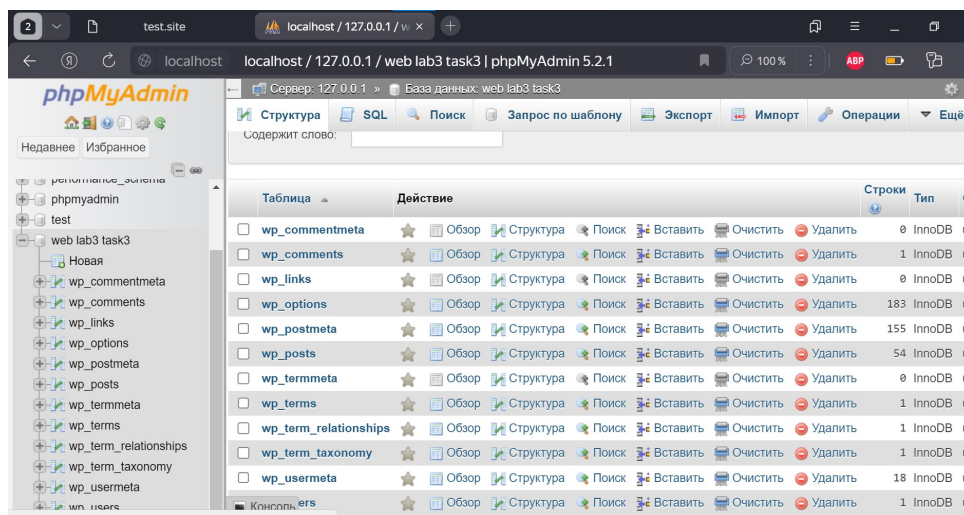


Рисунок 3.4

Далее переходим на сайт <http://test.site/>, регистрируемся, указывая название созданной выше БД (и другие параметры, имя, пароль и т.п.). После чего появляется консоль созданного сайта <http://test.site/wp-admin/> (Рис. 3.5).

После была установлена тема, поэтому сайт выглядит как на рисунке 3.6.

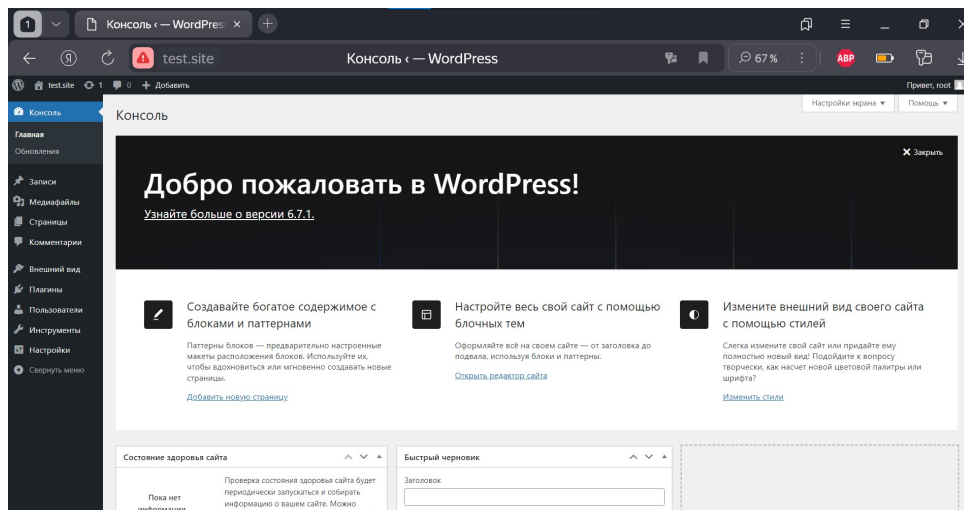


Рисунок 3.5

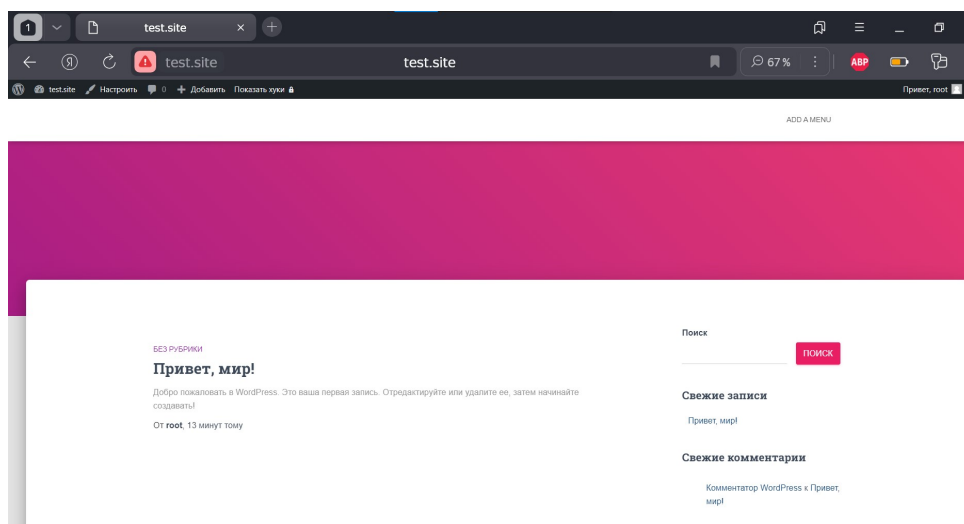


Рисунок 3.6

ЗАКЛЮЧЕНИЕ

В ходе лабораторной было осуществлено ознакомление с параллельными и последовательными задачами Gulp, реализована форма обратной связи с помощью php и настроен переход на сайт по ссылке test.site.