Министерство образования и науки

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Отчет по лабораторной работе №3

Выполнил: Гусев Ярослав Александрович

Группа: К3320

Проверила: Марченко Е. В.

Санкт-Петербург

2024

Оглавление

ВВЕДЕНИЕ

Цель: ознакомление с основами работы с php и инструментами отладки проекта.

Задание 1. Настройка gulp

Инициализируем npm проект командой **npm init.** Устанавливаем зависимости (browser-sync, gulp-series) с помощью **npm install**. Создаём gulpfile.js, в котором создаём 3 таски: для очистки конечной директории, для переноса HTML файлов и для переноса CSS файлов. Код файла представлен на рисунке 1.

```
const { src, dest, series, parallel, watch } = require('gulp');
const browserSync = require('browser-sync').create();

async function cleanDist() {
    const { deleteAsync } = await import('del');
    await deleteAsync(['dist/**', '!dist']);
}

function copyHTML() {
    return src('src/**/*.html')
        .pipe(dest('dist/'))
        .pipe(browserSync.stream());
}

function copyCSS() {
    return src('src/styles/**/*.css')
        .pipe(dest('dist/styles/'))
        .pipe(browserSync.stream());
}

const buildParallel = series(cleanDist, parallel(copyHTML, copyCSS));
const buildSeries = series(cleanDist, series(copyHTML, copyCSS));

exports.cleanDist = cleanDist;
exports.copyHTML = copyHTML;
exports.copyCSS = copyCSS;
exports.buildParallel = buildParallel;
exports.buildSeries = buildSeries
exports.serve = series(buildParallel, serve);
```

Рисунок 1 – Таски для параллельного и последовательного выполнения в gulpfile.js

Результат последовательного выполнения показан на рисунке 2.

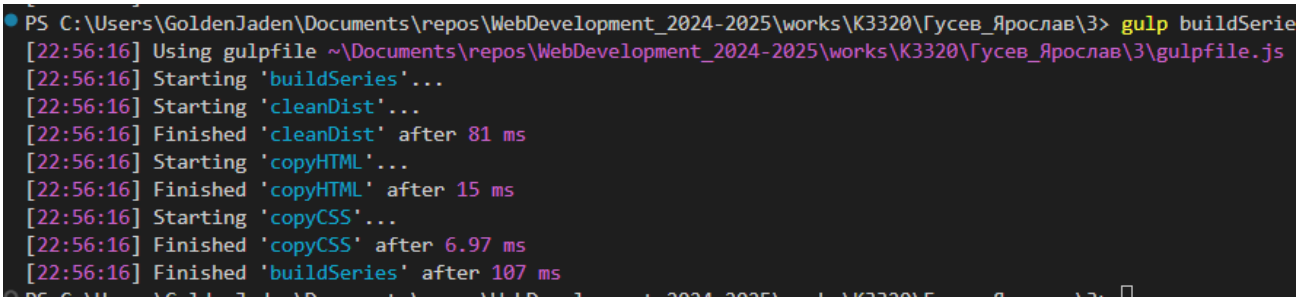

Рисунок 2 – Результат последовательного выполнения тасок

Результат параллельного выполнения показан на рисунке 3.

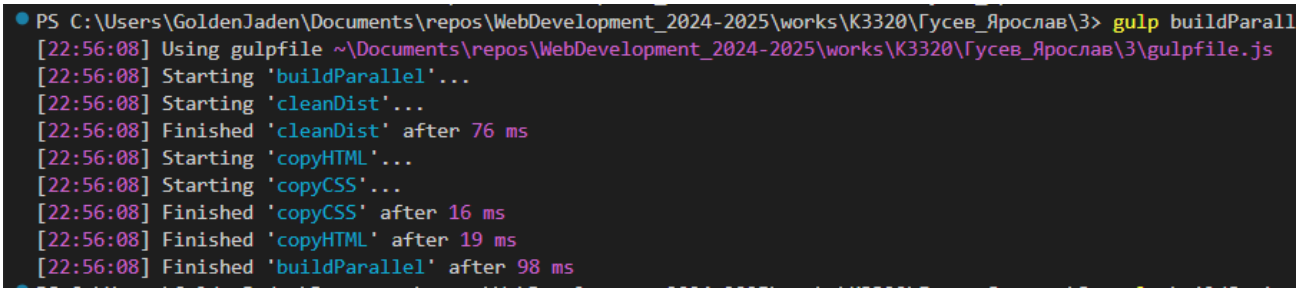

Рисунок 3 – Результат параллельного выполнения тасок

Для того, чтобы файлы при локальном изменении изменялись и в браузере, дополним наш gulpfile.js с помощью browser-sync следующим образом (рисунок 4):

```
const { src, dest, series, parallel, watch } = require('gulp');
const browserSync = require('browser-sync').create();

async function cleanDist() {
    const { deleteAsync } = await import('del');
    await deleteAsync(['dist/**', '!dist']);
}

function copyHTML() {
    return src('src/**/*.html')
        .pipe(dest('dist/'))
        .pipe(browserSync.stream());
}

function copyCSS() {
    return src('src/styles/**/*.css')
        .pipe(dest('dist/styles/'))
        .pipe(browserSync.stream());
}

function serve() {
    browserSync.init({
        server: {
            baseDir: "dist/"
        }
    });
    watch('src/**/*.html', copyHTML);
    watch('src/styles/**/*.css', copyCSS);
}

const build = series(cleanDist, parallel(copyHTML, copyCSS));

exports.cleanDist = cleanDist;
exports.copyHTML = copyHTML;
exports.copyCSS = copyCSS;
exports.build = build;
exports.serve = series(build, serve);
```

Рисунок 4 – Gulpfile.js с live-подгрузкой изменений

Результат выполнения **gulp serve** представлен на рисунке 5.

```
PS C:\Users\GoldenJaden\Documents\repos\WebDevelopment_2024-2025\works\K3320\Гусев_Ярослав\3> gulp serve
[23:02:52] Using gulpfile ~\Documents\repos\WebDevelopment_2024-2025\works\K3320\Гусев_Ярослав\3\gulpfile.js
[23:02:52] Starting 'serve'...
[23:02:52] Starting 'cleanDist'...
[23:02:52] Finished 'cleanDist' after 83 ms
[23:02:52] Starting 'copyHTML'...
[23:02:52] Starting 'copyCSS'...
[23:02:52] Finished 'copyCSS' after 19 ms
[23:02:52] Finished 'copyHTML' after 21 ms
[23:02:52] Starting 'serve'...
[Browsersync] Access URLs:
 --------------------------
 Local: http://localhost:3000
 --------------------------
    UI: http://localhost:3001
 --------------------------
[Browsersync] Serving files from: dist/
```

Рисунок 5 – Результат выполнения команды **gulp serve**

Задание 2. Форма обратной связи

Для создания формы обратной связи были созданы файлы feedback.html и feedback.php. Код скрипта feedback.php представлен на рисунке 6.

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $firstName = htmlspecialchars($_POST['first_name']);
    $lastName = htmlspecialchars($_POST['last_name']);
    $email = htmlspecialchars($_POST['email']);
    $feedback = htmlspecialchars($_POST['feedback']);
    $gender = htmlspecialchars($_POST['gender']);
    $categories = isset($_POST['categories']) ? $_POST['categories'] : [];

    echo "<h1>Ваши данные:</h1>";
    echo "<p>Имя: $firstName</p>";
    echo "<p>Фамилия: $lastName</p>";
    echo "<p>Email: $email</p>";
    echo "<p>Отзыв: $feedback</p>";
    echo "<p>Ваш выбор: $gender</p>";
    echo "<p>Категории: " . implode(', ', $categories) . "</p>";
}
?>
```

Рисунок 6 – Код скрипта feedback.php

Данный скрипт обрабатывает POST запросы, выводя полученную информацию. Метод GET используется для получения данных с сервера. Он отправляет данные через URL и подходит для запросов, не изменяющих состояние сервера. Метод POST отправляет данные в теле запроса и используется для отправки данных, например, форм, которые могут изменять состояние на сервере (например, создание записей). GET ограничен длиной данных, а POST может отправлять большие объемы информации.

Получившаяся страничка представлена на рисунке 7.

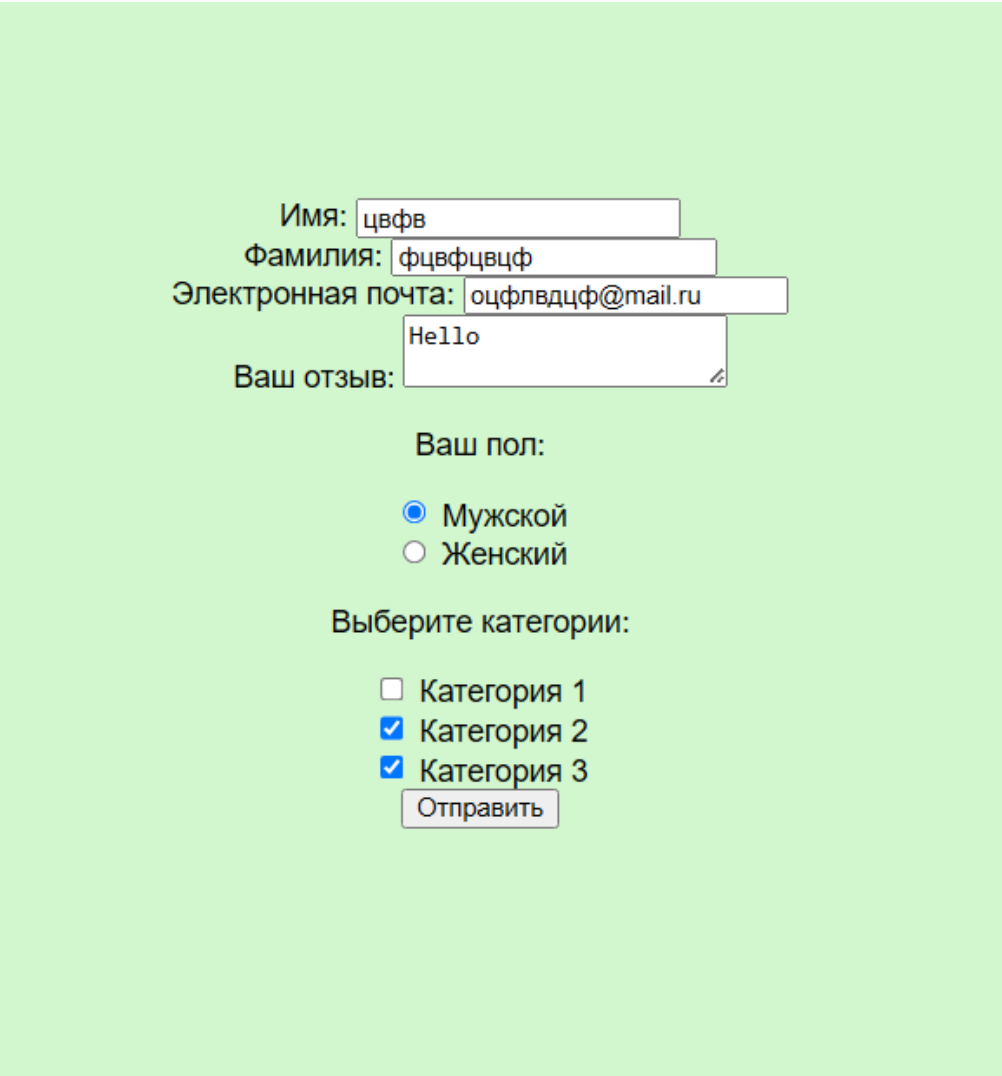

Рисунок 7 – Страница feedback.html

Так как browser-sync сам по себе не предназначен для работы с php скриптами, заодно поднимем php сервер на том же порте командой “**php -S localhost:3000**”.

Теперь наше приложение умеет обрабатывать POST-запросы. Результат нажатия на кнопку «отправить» представлен на рисунке 8.

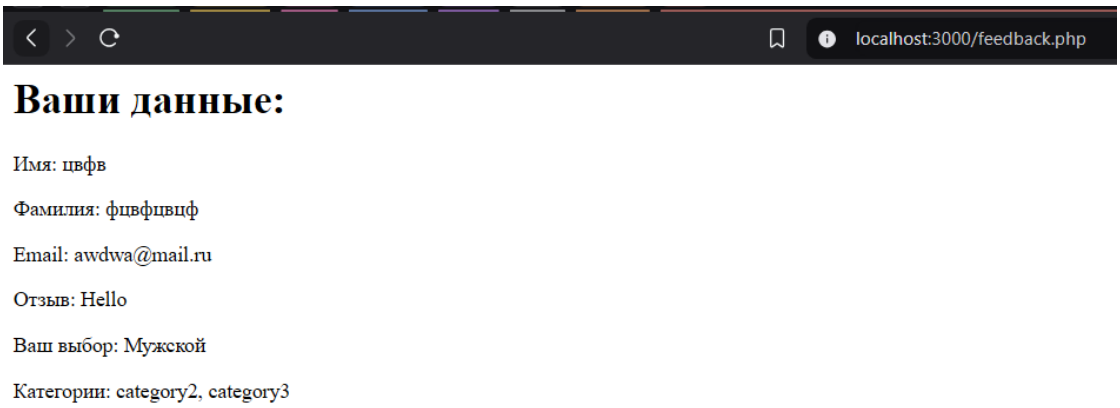

Рисунок 8 – Результат работы feedback.php на заданных данных

Задание 3. Деплой wordpress

Для установки wordpress вместе с mysql был использован docker compose (рисунок 9).

```yaml
version: '3.3'

services:
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     volumes:
       - wordpress_files:/var/www/html
     ports:
       - "80:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: my_wordpress_db_password
       WORDPRESS_DB_NAME: wordpress

   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: my_db_root_password
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: my_wordpress_db_password
volumes:
    wordpress_files:
    db_data:
```

Рисунок 9 – Файл docker-compose.yml

Для того, чтобы наш сервис был доступен по адресу http://test.site, добавляем в файл “C:\Windows\System32\drivers\etc\hosts” строку «127.0.0.1 test.site». Доступность сервиса по установленному адресу представлена на рисунке 10.

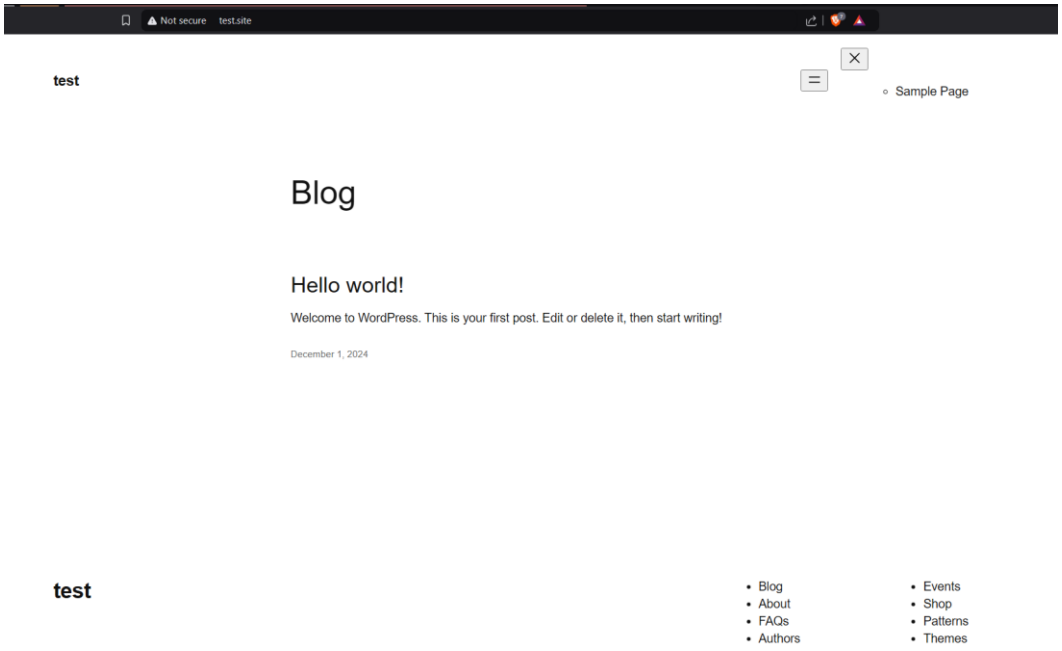

Рисунок 10 – Wordpress по адресу http://test.site

ЗАКЛЮЧЕНИЕ

В ходе данной работы был настроен gulp, создана форма и скрипт обрабатывающий POST запросы и установлен инструментарий для отладки проекта. Поставленная цель выполнена.