

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

**Отчёт по лабораторной работе 3**

**По дисциплине** Web-программирование

**Тема работы** Отчёт по лабораторной работе 3

**Обучающийся** Мартынюк Алексей Петрович

**Факультет** факультет инфокоммуникационных технологий

**Группа** K3320

**Направление подготовки** 11.03.02 Инфокоммуникационные технологии и системы связи

**Образовательная программа** Программирование в инфокоммуникационных системах

<b>Обучающийся</b>	_____	_____	<u>Мартынюк А.П.</u>
	(дата)	(подпись)	(Ф.И.О.)

<b>Руководитель</b>	_____	_____	<u>Марченко Е.В.</u>
	(дата)	(подпись)	(Ф.И.О.)

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Менеджер задач gulp.....	4
1.1 Последовательное и параллельное выполнение.....	4
1.2 Синхронизация изменений браузера.....	5
1.3 Запуск gulp.....	6
2 Форма обратной связи.....	7
2.1 Таблица в MySQL.....	7
2.2 Написание html формы и css стилей.....	7
2.3 Обработка формы с помощью php.....	8
2.4 Пример работы.....	8
2.4 Методы get и post.....	9
3 Wordpress.....	10
3.1 Написание docker-compose и nginx.conf.....	10
3.2 Создание сайта с WordPress.....	11
ЗАКЛЮЧЕНИЕ.....	12

## **ВВЕДЕНИЕ**

*Цель работы:*

изучить основы использования технологии gulp, познакомиться с языком программирования php, установить и изучить основы WordPress.

# 1 Менеджер задач gulp

## 1.1 Последовательное и параллельное выполнение

Создадим файл gulpfile.js, в котором опишем задачи для последовательного и параллельного вызова открытия url. Но для начала создадим функцию для чтения аргументов из командной строки и функцию открытия url:

```
function getArgs() {
  const args = process.argv.slice(2);
  const urls = [];
  let interval = 5000;

  args.forEach(arg => {
    if (arg.startsWith('--url=')) {
      urls.push(arg.split('=')[1]);
    } else if (arg.startsWith('--interval=')) {
      interval = parseInt(arg.split('=')[1], 10);
    }
  });

  return { urls, interval };
}
```

Рисунок 1 — Функция getArgs()

```
Codeium: Refactor | Explain | Generate JSDoc
function openUrl(url) {
  return function(done) {
    console.log(`Opening URL: ${url}`);
    open(url, { app: { name: 'chromium' } }).catch(err => {
      console.error(`Failed to open URL: ${url}`, err);
    });
    done();
  };
}
```

Рисунок 2 — Функция openUrl()

Далее было написано 2 task для последовательного и параллельного выполнения функций:

```

gulp.task('open-urls-sequential', function(done) {
  const { urls, interval } = getArgs();

  if (urls.length === 0) {
    console.error("No URLs provided. Use --url=<url1> --url=<url2> ...");
    return done();
  }

  const tasks = urls.map((url, index) => {
    return function(cb) {
      setTimeout(() => {
        openUrl(url)(cb);
      }, index * interval);
    };
  });

  gulp.series(...tasks)(done);
});

```

Рисунок 3 — gulp.series()

```

gulp.task('open-urls-parallel', function(done) {
  const { urls } = getArgs();

  if (urls.length === 0) {
    console.error("No URLs provided. Use --url=<url1> --url=<url2> ...");
    return done();
  }

  const tasks = urls.map(url => openUrl(url));
  gulp.parallel(...tasks)(done);
});

```

Рисунок 4 — gulp.parallel()

## 1.2 Синхронизация изменений браузера

Был также добавлен task «serve», который проверяет изменения в файлах .html и .css в текущей директории и подгружает их в браузер.

```

gulp.task('serve', function() {
  browser.init({
    server: {
      baseDir: '.'
    }
  });

  gulp.watch('/*.html').on('change', browser.reload);
  gulp.watch('./css/*.css').on('change', browser.reload);
});

```

Рисунок 5 — browser-sync

## 1.3 Запуск gulp

Запустим наши tasks с помощью команды: `gulp --url=yandex.ru --url=google.com --interval=3000`

```
[15:42:30] Using gulpfile ~/code/reports_web_lesha/WebDevelopment_2024-2025/work/K3320/Мартынюк_Алексей/3/part_1/gulpfile.js
[15:42:30] Starting 'default'...
[15:42:30] Starting 'open-urls-sequential'...
[15:42:30] Starting '<anonymous>'...
Opening URL: yandex.ru
[15:42:30] Finished '<anonymous>' after 12 ms
[15:42:30] Starting '<anonymous>'...
Opening URL: google.com
[15:42:33] Finished '<anonymous>' after 3.01 s
[15:42:33] Finished 'open-urls-sequential' after 3.02 s
[15:42:33] Starting 'open-urls-parallel'...
[15:42:33] Starting '<anonymous>'...
[15:42:33] Starting '<anonymous>'...
Opening URL: yandex.ru
[15:42:33] Finished '<anonymous>' after 13 ms
Opening URL: google.com
[15:42:33] Finished '<anonymous>' after 30 ms
[15:42:33] Finished 'open-urls-parallel' after 35 ms
[15:42:33] Starting 'serve'...
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
  External: http://192.168.46.143:3000
-----
    UI: http://localhost:3001
  UI External: http://192.168.46.143:3001
-----
[Browsersync] Serving files from: ./
[Browsersync] Reloading Browsers...
[Browsersync] Reloading Browsers...
[Browsersync] Reloading Browsers...
[15:42:33]
```

Рисунок 6 — Запуск

## 2 Форма обратной связи

### 2.1 Таблица в MySQL

С помощью миграций была создана таблица для сохранения обратной связи пользователей

```
SET GLOBAL character_set_server = utf8mb4;

CREATE DATABASE IF NOT EXISTS `lab3` CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;


USE `lab3`;

CREATE TABLE feedback (
  id INT(11) AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  email VARCHAR(100) NOT NULL,
  feedback TEXT NOT NULL,
  feedback_type ENUM('complaint', 'suggestion') NOT NULL,
  services TEXT
);
```

Рисунок 7 — Таблица feedback

### 2.2 Написание html формы и css стилей

Далее была написана форма обратной связи с различными типами полей:



← → ↻ localhost/index.html

### Feedback Form

Имя:

Фамилия:

Электронная почта:

Обратная связь:

Выберите тип обратной связи:

☐ Жалоба

☐ Предложение

Дополнительные услуги:

☐ Подписка на новости

☐ Получение обновлений

☐ Специальные предложения

Рисунок 8 — Feedback Form

## 2.3 Обработка формы с помощью php

Был написан php-скрипт для обработки ответов формы:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $firstName = htmlspecialchars($_POST['first_name']);
    $lastName = htmlspecialchars($_POST['last_name']);
    $email = htmlspecialchars($_POST['email']);
    $feedback = htmlspecialchars($_POST['feedback']);
    $feedbackType = htmlspecialchars($_POST['feedback_type']);
    $services = $_POST['services'] ?? [];

    $conn = new mysqli("mysql", "root", "password", "lab3");

    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $stmt = $conn->prepare("INSERT INTO feedback (first_name, last_name, email, feedback, feedback_type, services) VALUES (?, ?, ?, ?, ?, ?)");
    $servicesString = implode(" ", $services);

    $stmt->bind_param("ssssss", $firstName, $lastName, $email, $feedback, $feedbackType, $servicesString);

    if ($stmt->execute()) {
        echo "Feedback submitted successfully.";
    } else {
        echo "Error: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
} else {
    echo "Invalid request method.";
}
```

Рисунок 9 — submit.php

## 2.4 Пример работы

Feedback Form

Имя:

Фамилия:

Электронная почта:

Обратная связь:

Выберите тип обратной связи:

☐ Жалоба

☒ Предложение

Дополнительные услуги:

☒ Подписка на новости

☒ Получение обновлений

☒ Специальные предложения

Рисунок 10 — Отправка формы





Рисунок 11 — Результат отправки

## 2.4 Методы get и post

Данные передаются в URL-адресе в виде пар "имя=значение". Это делает их видимыми и доступными для всех, кто может видеть URL. Кроме того такой вид запроса поддерживает только строковые данные

При отправки данных через post все параметры передаются в теле запроса. Что обеспечивает большую конфиденциальность данных. Кроме того можно передовать различные типы данных, включая, например, файлы.

## 3 Wordpress

### 3.1 Написание docker-compose и nginx.conf

Для запуска wordpress были созданы два файла. Один из них docker-compose.yaml со следующими сервисами: db с MySQL, nginx — веб-сервер, wordpress. И создан файл с конфигурацией nginx

```
services:
  wordpress:
    image: wordpress:latest
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress_user
      WORDPRESS_DB_PASSWORD: wordpress_password
      WORDPRESS_DB_NAME: wordpress_db
    volumes:
      - wordpress_data:/var/www/html
    networks:
      - wp_network

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: wordpress_db
      MYSQL_USER: wordpress_user
      MYSQL_PASSWORD: wordpress_password
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - wp_network

  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - wordpress
    networks:
      - wp_network

volumes:
  wordpress_data:
  db_data:

networks:
  wp_network:
```

Рисунок 12 – Файл docker-compose.yaml

## 3.2 Создание сайта с WordPress

После запуска докер контейнеров с помощью wordpress был получен следующий простой сайт по адресу test.site.

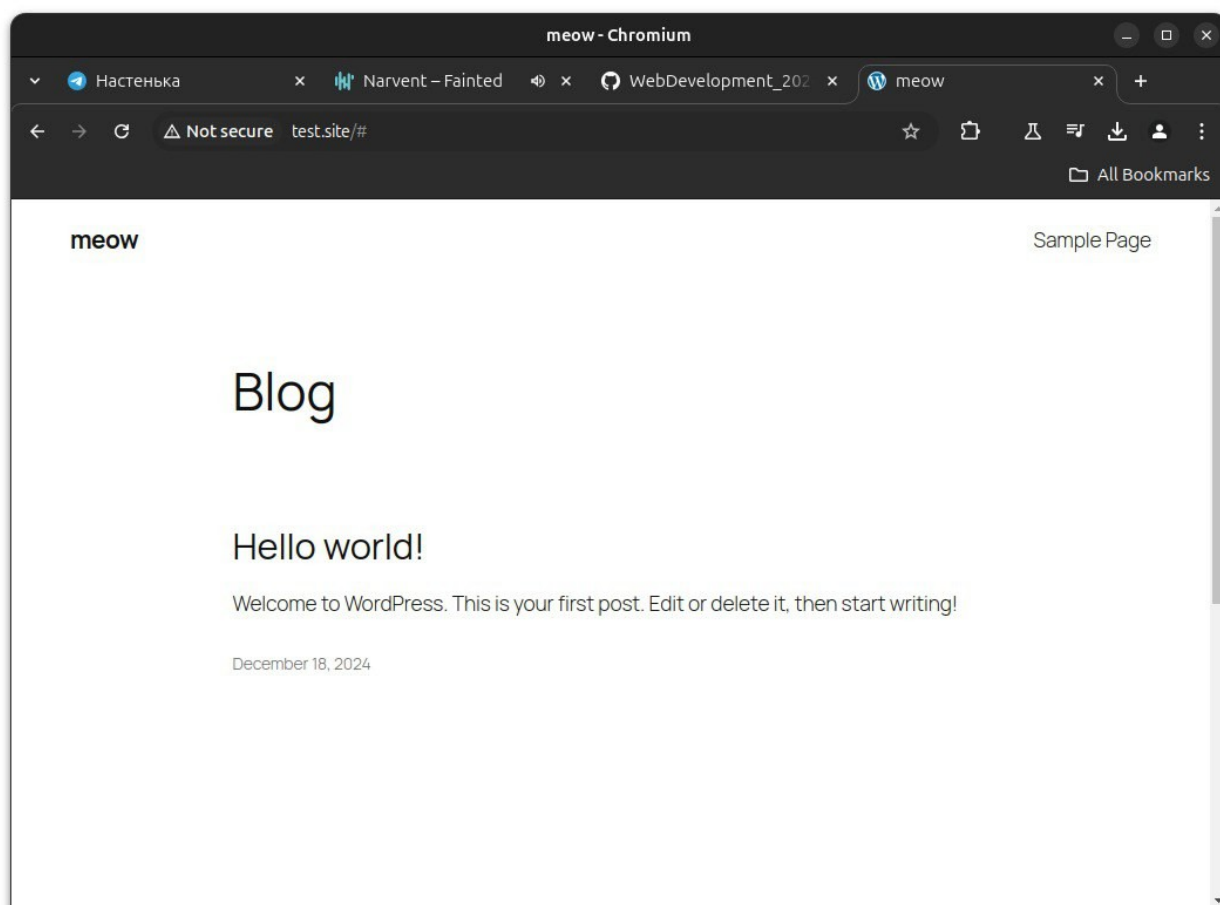


Рисунок 13 — test.site

## **ЗАКЛЮЧЕНИЕ**

При выполнении практических заданий лабораторной получены базовые навыки программирования на php, изучены get и post запросы. Были изучены менеджер задач gulp и движок wordpress.