

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 Основы работы с GULP и локальными серверами

Обучающийся Адрат Олеся Александровна

Факультет Факультет инфокоммуникационных технологий

Группа K3321

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	<u>23.11.2024</u> (дата)	<u> </u> (подпись)	<u>Адрат О.А.</u> (Ф.И.О.)
--------------------	-----------------------------	--	-------------------------------

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

Санкт-Петербург
2024 г.

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ	3
1 Задание 1	4
2 Задание 2	7
3 Задание 3	11
ЗАКЛЮЧЕНИЕ	14

ВВЕДЕНИЕ

Цель работы:

Изучить и настроить инструменты разработки для автоматизации задач, создания формы обратной связи и установки локального сервера для тестирования веб-проектов.

Задачи:

1. Настроить Gulp для автоматического выполнения задач и перезагрузки браузера при изменении файлов.
2. Разработать форму обратной связи с радиокнопками и чекбоксами и написать PHP-скрипт для отправки данных.
3. Описать использование методов GET и POST в отчёте.
4. Установить и настроить WordPress на локальном сервере для тестирования.

1 Задание 1

В первом задании следовало настроить GULP. Он был установлен при выполнении лабораторной работы 2, поэтому дополнительный шаг для установки не требовался.

В первую очередь было создано два таска - `job` и `cv` (рис. 1.1), настроенные на последовательное и параллельное выполнение с помощью метода `exports` (рис. 1.2)

```
function job(cb) {  
    console.log("i'm looking for a job right now");  
    cb();  
}  
  
function cv(cb) {  
    console.log('you can view my cv on github');  
    cb();  
}
```

Рисунок 1.1 — Gulp tasks

```
exports.serve = serve;  
exports.default = serve; //ставим функцию по дефолту  
  
exports.series = series(cv, job);  
exports.parallel = parallel(job, cv);
```

Рисунок 1.2 — Gulp tasks' settings

При запуске команды `gulp series`, вызывается функция `exports.series`, которая выполняет таски последовательно в указанном порядке (рис. 1.3)

```

● olesaadrat@MacBook-Air-Olesa-3 task2 % gulp series
[18:14:59] Using gulpfile ~/Desktop/itmo/5 semester/web/lab2/task2/gulpfile.js
[18:14:59] Starting 'series'...
[18:14:59] Starting 'cv'...
you can view my cv on github
[18:14:59] Finished 'cv' after 862 μs
[18:14:59] Starting 'job'...
i'm looking for a job right now
[18:14:59] Finished 'job' after 414 μs
[18:14:59] Finished 'series' after 2.85 ms
○ olesaadrat@MacBook-Air-Olesa-3 task2 % █

```

Рисунок 1.3 — Вызов gulp series

Аналогично с запуском команды `gulp parallel`, теперь можно заметить, что задачи выполняются параллельно (1.4)

```

● olesaadrat@MacBook-Air-Olesa-3 task2 % gulp parallel
[18:15:33] Using gulpfile ~/Desktop/itmo/5 semester/web/lab2/task2/gulpfile.js
[18:15:33] Starting 'parallel'...
[18:15:33] Starting 'job'...
[18:15:33] Starting 'cv'...
i'm looking for a job right now
[18:15:33] Finished 'job' after 879 μs
you can view my cv on github
[18:15:33] Finished 'cv' after 927 μs
[18:15:33] Finished 'parallel' after 2.26 ms
○ olesaadrat@MacBook-Air-Olesa-3 task2 % █

```

Рисунок 1.4 — Вызов gulp parallel

Теперь настроим инструмент `browserSync` для автоматической перезагрузки страницы браузера после внесения изменений.

Параметр `server` указывает корневую директорию, метод `watch('*.*html')` следит за изменениями всех html-файлов в указанной директории, а метод `reload` отвечает за автоматическую перезагрузку страницу после каждого сохраненного изменения в файлах (1.5)

```

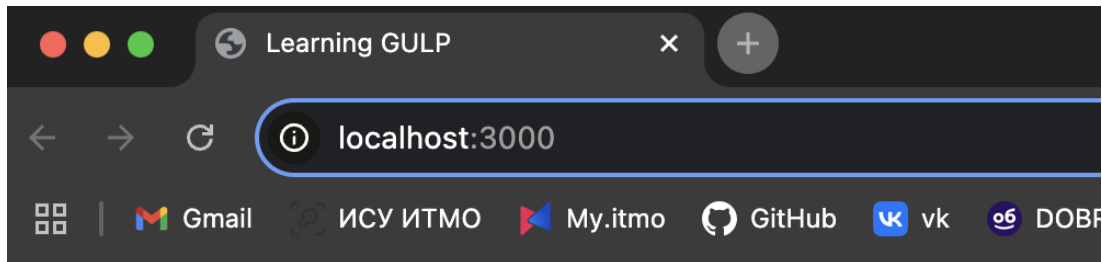
function serve() {
  browserSync.init({
    server: {
      baseDir: '/Users/olesaadrat/Desktop/itmo/5 semester/web/lab2/task2'
    }
  });

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

```

Рисунок 1.5 — Код задачи browserSync

Теперь проверим работу browserSync. Для начала запустим его с помощью команды `gulp`, так как он был выбран функцией по умолчанию. Можем видеть первоначальный текст страницы, расположенной на локальном сервере (1.6)

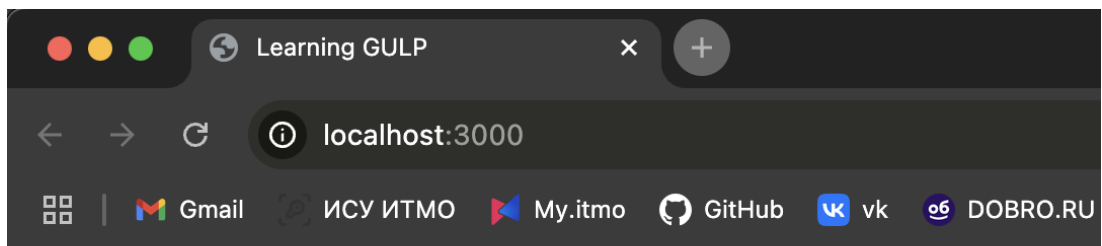


Here is my attempt to create a good GULP task

Hello my name is Olesya Adrat

Рисунок 1.6 — Первая версия страницы

Затем были внесены некоторые изменения в файл `index.html`, после этого сохранены, и мы можем наблюдать результат без перезагрузки страницы (2.3)



Here is my attempt to create a good GULP task

Hello my name is Olesya! Nice to meet you!

/_/\

(0.0)

> ^ <

Рисунок 1.7 — Измененная версия страницы

2 Задание 2

В следующем задании нужно было создать форму для отправки информации по обратной связи от пользователя сайта. Форма собирает данные от пользователя об их предпочтениях в автомобилях.

Далее был написан докерфайл со следующими контейнерами:

nginx - веб сервер для обработки http-запросов, для него был открыт 80 порт. Были подключены папки, где хранятся системные файлы wordpress и nginx, а также папка lab-2-form.

php-fpm - сервер для выполнения PHP-скриптов, сборка идет из настроенного php-образа папки ./php-fpm

mysql - база данных для хранения информации, задаются параметры через переменные окружения

wordpress - движок для создания сайтов, который соединяется с базой данных

```
version: "3.8"

services:
  nginx:
    image: nginx:alpine
    container_name: nginx2
    ports:
      - "80:80"
    volumes:
      - ./wordpress:/var/www/wordpress/html
      - ./lab2-2-form:/var/www/lab2-2-form/html
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - php-fpm
    networks:
      - wordpress_network

  php-fpm:
    build: ./php-fpm
    container_name: php-fpm2
    volumes:
      - ./wordpress:/var/www/wordpress/html
      - ./lab2-2-form:/var/www/lab2-2-form/html
    networks:
      - wordpress_network
```

Рисунок 2.1 — Контейнеры nginx и php-fpm

```

mysql:
  image: mysql:5.7
  container_name: mysql2
  platform: linux/x86_64
  environment:
    MYSQL_ROOT_PASSWORD: rootpassword
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpressuser
    MYSQL_PASSWORD: wordpresspassword
  volumes:
    - mysql_data:/var/lib/mysql
  networks:
    - wordpress_network

wordpress:
  image: wordpress:latest
  container_name: wordpress2
  environment:
    WORDPRESS_DB_HOST: mysql:3306
    WORDPRESS_DB_NAME: wordpress
    WORDPRESS_DB_USER: wordpressuser
    WORDPRESS_DB_PASSWORD: wordpresspassword
  volumes:
    - ./wordpress:/var/www/wordpress/html
  networks:
    - wordpress_network

volumes:
  mysql_data:

networks:
  wordpress_network:
    driver: bridge

```

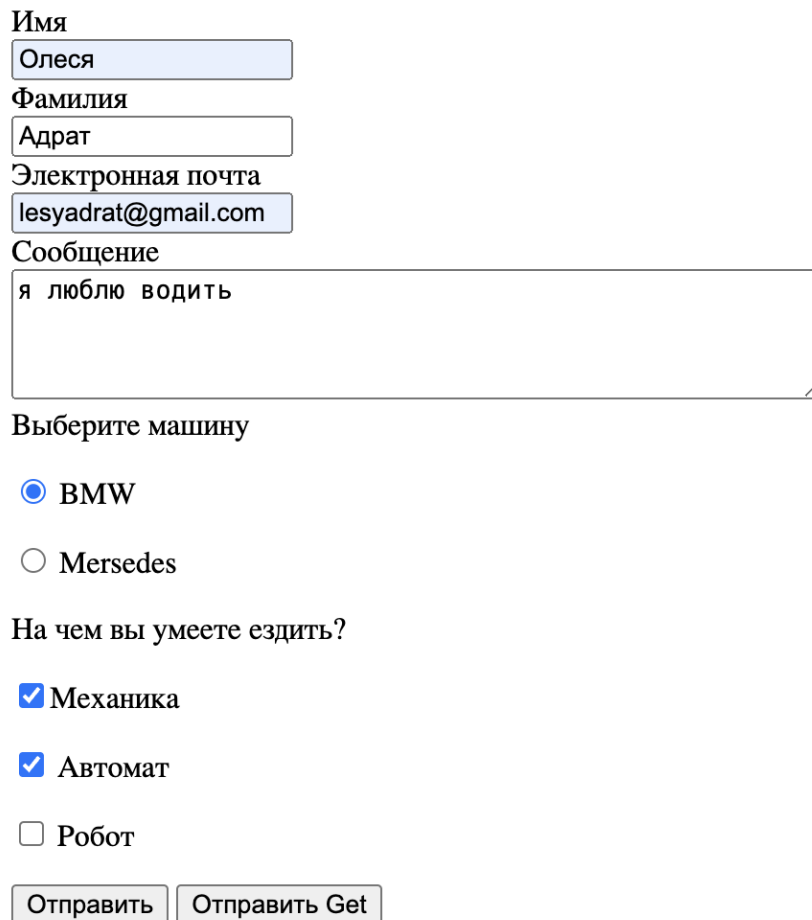
Рисунок 2.2 — Контейнеры mysql и wordress

Почему для развертывания сервера был выбран именно **docker**?

- Каждый сервис запускается в отдельном контейнере во избежание конфликтов между зависимостями
- Легкое масштабирование
- Универсальность окружения, код будет работать одинаково на всех платформах

После этого был написан простой php скрипт и код страницы в файле `index.html`. Было использовано 2 радиокнопки и 3 чекбокса. Также внизу страницы установлены две кнопки - для отправки **POST** и **GET** запросов. Командой `docker compose up` запускаем наш докерфайл, теперь можно попробовать открыть страницу с формой в браузере.

Опрос среди молодежи



Имя
Олеся

Фамилия
Адрат

Электронная почта
lesyadrat@gmail.com

Сообщение
я люблю водить

Выберите машину

☒ BMW

☐ Mercedes

На чем вы умеете ездить?

☒ Механика

☒ Автомат

☐ Робот

Отправить Отправить Get

Рисунок 2.3 — Страница с формой в браузере

Затем были отправлены оба запроса:

POST - метод для отправки данных на сайт, содержит тело запроса, в котором передается информация. Этот вид запроса чаще используется для отправки конфиденциальных данных(например, личные пароли, данные бан-

ковских карт и тд). Как мы можем видеть, заголовок после отправки формы не содержит в себе никакой информации(рис. 2.4)

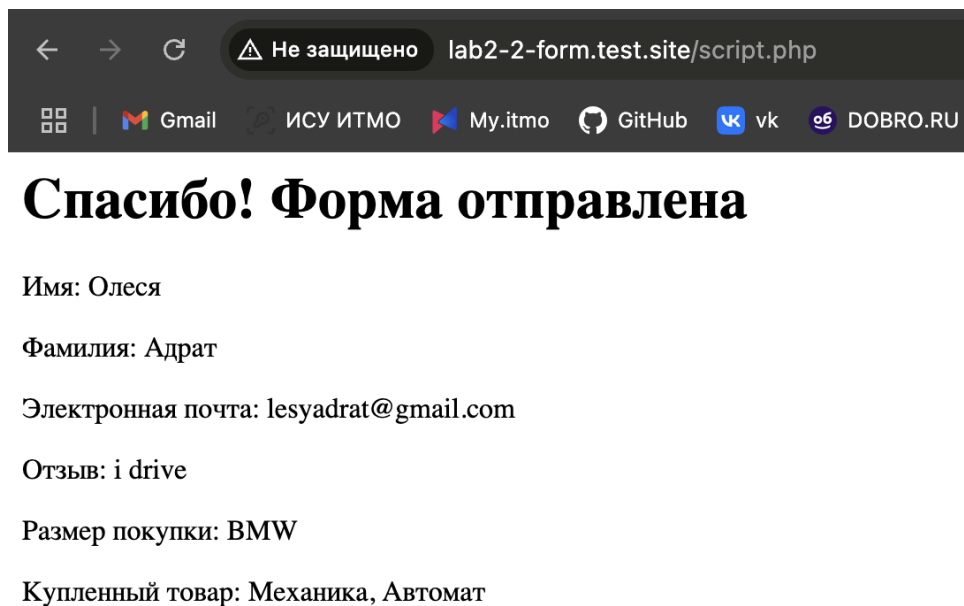


Рисунок 2.4 — Результат после POST запроса

GET - метод чтения данных с сайта, при котором происходит передача параметров через ссылку. Этот вид запроса чаще всего используется для фильтров, например, его используют интернет-магазины для просмотра определенного каталога товаров. Судя по ссылке на картинке, все данные, которые были введены в форме, оказались в заголовке(рис. 2.5)

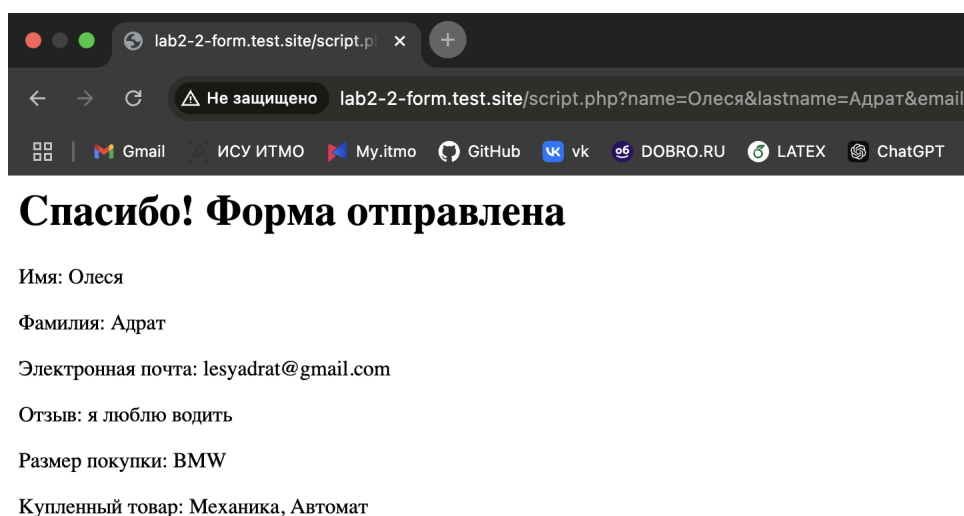


Рисунок 2.5 — Результат после GET запроса

3 Задание 3

В последнем задании требовалось установить движок wordpress и настроить портал `http://test.site`

Запуск локального сервера также был произведен с помощью докерфайла. Были прописаны все необходимые контейнеры, основной - wordpress, для настройки движка(код представлен во 2-м задании)

Командой `docker compose up` запускаем наш докерфайл. Локальный сервер запущен. Теперь перейдем по адресу `http://test.site`. Нам предлагают выбрать язык wordpress(рис. 3.1)

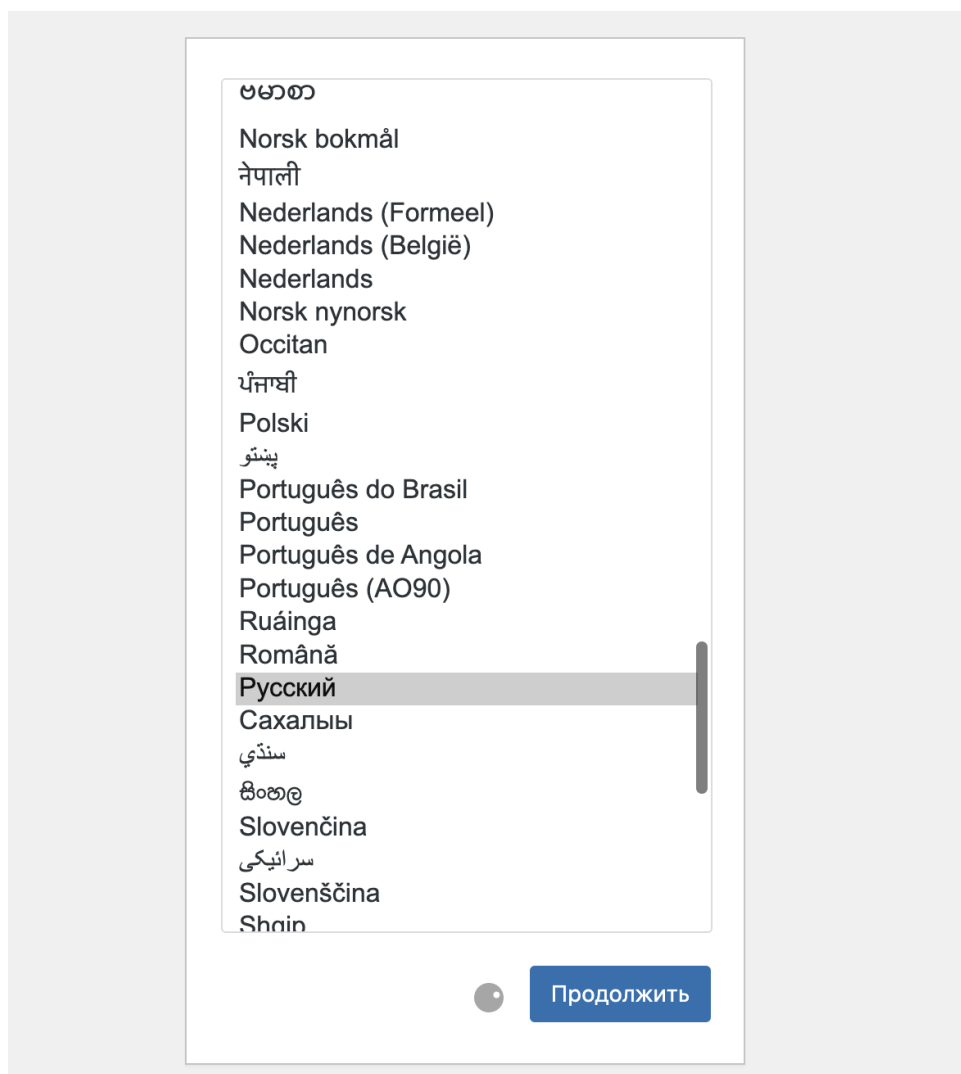


Рисунок 3.1 — Выбор языка

Далее открывается форма с регистрацией нового пользователя. Необходимо заполнить все поля(рис. 3.2)

Требуется информация

Пожалуйста, укажите следующую информацию. Не переживайте, потом вы всегда сможете изменить эти настройки.

Название сайта: test.site

Имя пользователя: olesia
Имя пользователя может содержать только латинские буквы, пробелы, подчёркивания, дефисы, точки и символ @.

Пароль: 123
Очень слабый
Важно: Этот пароль понадобится вам для входа. Сохраните его в надёжном месте.

Подтвердите пароль: ☒ Разрешить использование слабого пароля.

Ваш e-mail: lesyadrat@gmail.com
Внимательно проверьте адрес электронной почты, перед тем как продолжить.

Видимость для поисковых систем: ☐ Попросить поисковые системы не индексировать сайт
Будет ли учитываться этот запрос — зависит от поисковых систем.

[Установить WordPress](#)

Рисунок 3.2 — Регистрация пользователя

После регистрации, открывается окно входа в систему, необходимо ввести свои логин и пароль.

Имя пользователя или email: olesia

Пароль: [masked]

☐ Запомнить меня

[Войти](#)

Рисунок 3.3 — Авторизация пользователя

После авторизации, мы попадаем на **wp-admin**, после чего немного редактируем тему и текст страницы. В результате, после перехода на страницу **http://test.site**, видим измененную страницу, наслаждаемся результатом.

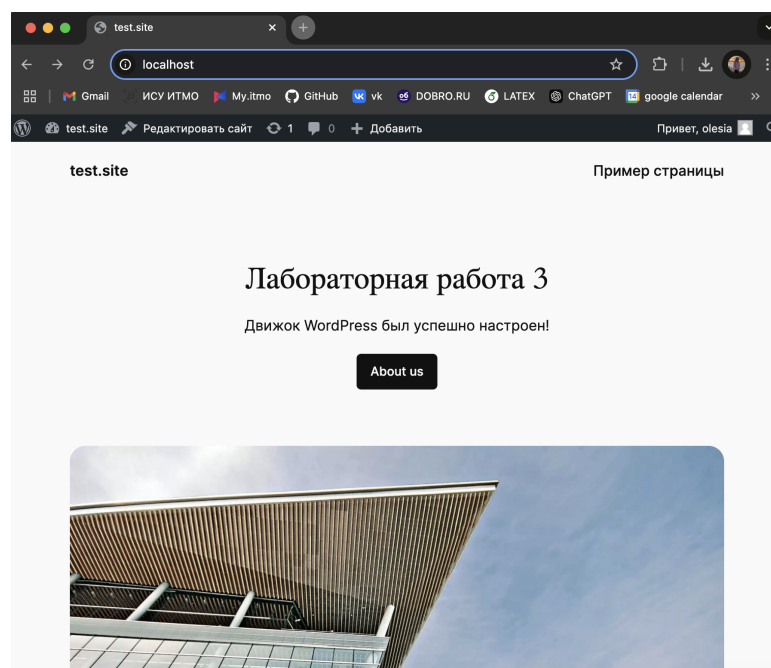


Рисунок 3.4 — Результат работы

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы был настроен веб-сервер Nginx для отображения разработанного портала и формы обратной связи. С помощью формы пользователи могут отправлять информацию, такую как имя, фамилия и электронная почта, с использованием методов GET и POST, что было протестировано и проанализировано в отчёте.

Настроенная виртуальная среда с использованием Nginx позволила обеспечить быстрый и эффективный доступ к сайту по адресу `http://test.site`, а автоматическая перезагрузка сервера при изменении файлов ускорила процесс разработки. Таким образом, цели лабораторной работы достигнуты: создана форма обратной связи, реализована обработка данных на PHP и обеспечена работа портала на локальном сервере.